Applying Experientialism to HCI Methods

by

Manuel Imaz

A thesis submitted in partial fulfillment of the
requirements of Napier University for
the degree of Doctor of Philosophy

SCHOOL OF COMPUTING
NAPIER UNIVERSITY
EDINBURGH, UK

Committee in charge:
Professor David Benyon, Chair
Professor Chris Johnson
Professor John Waterworth

January 2001

Acknowledgements

I would like to thank Dr. David Benyon, who first discovered the possibility of a PhD thesis in the seeds of a MSc thesis done at the Open University, and also because he supported me along the entire source-path-goal trajectory and to Michael Smyth for his revision and valuable comments. I would also like to thank Bonnie Nardi for her exchange of comments regarding my original ideas about Activity Theory, and John Waterworth for his contributions regarding the role of metaphor in HCI design. Their contribution has been very important in the development of my own ideas.

Table of Contents

Table of Figures

Abstract

The aim of this thesis is to incorporate the results of Experientialism in the domain of Human-Computer Interaction. The purpose is twofold: on the one hand it shows how some concepts of Experientialism like metaphor, image-schema, stories or conceptual integration may be used to explain where some concepts of HCI come from. On the other hand it uses the same conceptual background to support the design activity: the same concepts of Experientialism may be employed to build new conceptual artifacts in order to design User Interfaces and application software in general. One of the most fruitful ideas Experientislim may offer is *conceptual integration* as the basis upon which to construct new design solutions.

Notwithstanding the pervasive use of metaphor in everyday language and even in HCI texts, there is a considerable amount of criticism regarding the use of metaphor in designing user interfaces based on the assumption that this practice may be the origin of troubles when using such software products. That is why one of the chapters is aimed at showing that not only the use of metaphor is pervasive in HCI but even the use of figurative language as well. Not only figurative language is usually employed but it is even one of the main tools for conceptualising new ideas and concepts required in the activity of software development.

The Thesis proposes a framework aimed at designing User Interfaces based on the concepts of Experientialism. The proposal integrates two phases (analysis and design) the same way as most of software development methods do, trying to profit on the broad scope of the cognitive processes such as image-schema, metaphor and conceptual integration. These general concepts may be well suited to build conceptual models upon which to elaborate the user interfaces and the *optimality principles* proposed to study the suitability of conceptual integration may be also used as validity criteria to evaluate such design artifacts.

In order to validate such a proposal, the Thesis shows how to use the framework in two different situations: i) to explain why a problem such as the Mac trashcan -used to eject diskettes- is not a problem of using metaphors but an unfortunate design decision, and ii) to be applied in the design of a new User Interface.

Other concepts of Experientialism are proposed in capturing user requirements. The concept of *story* is the ground on which to build *scenarios* or *use cases*, as stories are a more general cognitive process and a form of telling things at a more general level. That is why the *user stories* may be mapped to *use cases*, as both are essentially different type of stories and the capture of requirements is a way of specifying one type of stories (use cases) based on the original stories (user stories).

Chapter 1

Introduction

This thesis is motivated by the observation that most of the phenomena studied in relation with cognitive semantics are also applicable to HCI constructs and methods. One of them is the pervasive use of metaphor in language that motivated our original observation that objects –as programming constructs- are considered as people (Imaz, 1995). As we will see, the projection of human capacities to unanimated things in real life is an ongoing process in natural language expressions. It is interesting to observe the contradiction between the results of cognitive science and what we usually read in the literature about computing science: a widely opposed position regarding the use of metaphor. Moreover, it is usual to see remarks about the unfortunate metaphor use in designing user interfaces, or the dangers of anthropomorphizing some concepts of computer science.

This is the original reason that has led us to study other aspects bound to HCI methods. Among the main issues we have studied in this thesis are the following:

1 Where most of HCI concepts come from and how these concepts are derived from some generic cognitive schemas by means of specific mechanisms applied to such schemas.

2 How metaphors are a pervasive mechanism used anywhere in the field of Human-Computer Interaction. Metaphors are a general projection mechanism between two conceptual domains, which is frequently used from the field of requirements engineering through modelling elements and even to the design of specific user interfaces.

3 How the use of figurative language is fundamental –and more frequent that we are aware of- in computer sciences in general. The traditional stand tries to separate both types of languages in such a way that some discourses appear as referring to the 'real world' while others -more figurative or analogical- seem as intuitive approaches or even poetic ways of speech.

4 How conceptual integration (*blend*) is a basic mechanism for producing new design solutions, and how it may be described as patterns for designing computer-based signs.

5 How to derive a pragmatics to use most of the concepts of cognitive science for designing user interfaces.

6 How the concept of story may be used in requirement elicitation and the design of interactions.

This thesis describes a project that provides examples to illustrate the cases where some constructions are explained in function of cognitive mechanisms while showing at the same time how other mechanisms may be useful in the task of designing new application software. In any case, the fact of being aware of unconscious operations employed in the design task will always give us an advantageous position over a classical approach, where most of decisions are taken using methods or techniques, which we are not conscious of.

1.1 Experientialism

The main theoretical approach used in this thesis is Experientialism. The reason for using this framework is its capacity to explain numerous cognitive processes in terms of embodied schemas, metaphors and conceptual integration. These concepts refer to basic cognitive processes, participating in all domains of our mental life, so its pervasiveness entitles them as candidates to be used in the context of Human-Computer Interactions as well.

Lakoff (1987, 1988) has introduced the term Experientialism as an easy way of including a group of scholars and works:

Cognitive grammar (Langacker)

Cognitive Semantic (Lakoff and Johnson)

Mental Spaces (Fauconnier)

Conceptual Integration (Fauconnier and Turner)

Parabolic Projection and Stories (Turner)

The main idea is that cognitive processes mainly ground on physical experience as well as on social and cultural fields. Our bodily and social activity generates image-schemas through which, and by means of projections to more abstract domains, we are able to categorise and think. All the cognitive building is erected on such bodily foundations. This idea differentiates Experientialism from other current theories and, in particular, from those based on an objectivist philosophy as –for example- Cognitive Psychology. Experientialism tries to overcome the contradiction between objective-subjective that has prevailed as the analysis of knowledge for many centuries.

One of the basic mechanisms used by Experientialism is metaphoric projection (from concrete domains to more abstract ones) and there is a large amount of examples that show how natural language uses metaphors based on bodily experience. Nevertheless, the most controversial claim about metaphor in the last 20 years is that this concept is essentially a figure of speech. Experientialism, on the contrary, states that metaphor is a *mental* mapping that significantly influences how people think, reason, and imagine in everyday life.

In opposition with the general idea that language is a general container for thoughts, Fauconnier points out that "language...is but the tip of the iceberg of cognitive construction. As discourse unfolds, much is going on behind the scenes: internal structure emerges and spreads, viewpoint and focus keep shifting. Everyday talk and commonsense reasoning are supported by invisible, highly abstract, mental creations, which grammar helps to guide, but does not by itself define" (1994, p. xxiii). These highly abstract, mental creations are *mental spaces* and *blends*, built upon them. Most of the time, we are not aware of such mental creations, and they are only deducible from a linguistic analysis.

Many cognitive psychologists are critical of the claim that such examples of metaphor in conventional language really indicate that people ordinarily think about many concepts in terms of metaphors. One of the main objections raised by cognitive psychologists is that data per se does not prove that metaphorical mappings are composed automatically and effortlessly

during metaphor understanding. They add that "trying to infer aspects of conceptual knowledge from an analysis of systematic patterns of linguistic structure results in theories that seem post hoc" (Gibbs, 1998, p. 91).

This last argument -apparently quite convincing- would imply that our endless list of examples in favour of metaphor as a basic cognitive mechanism would not account as proof for our arguments. For those who use the above argument, our examples would merely be a list of nice literary cases. Fortunately, we count on the results offered recently by Snirivas Narayanan (1997).

In Linguistics, *aspect* –the general structure of events- has a conceptual structure and logic. What Narayanan discovered was that exactly the same neural structure that can perform motor control also characterizes the conceptual structure of linguistic aspect, and the same neural mechanism that can control bodily movements can perform logical inferences about the structure of action in general (Lakoff and Johnson, 1999, p. 42).

Narayanan, using an extremely ingenious theoretical device, has shown that a simulation model employed for simulating bodily actions is able to simulate the inferences involved in a metaphorical expression. He constructed a neural model of conceptual metaphor and then found cases in which body-based metaphors were used in an abstract domain, in this case, international economics. "Narayanan then showed that models of the motor schemas for physical actions can –under metaphoric projection- perform the appropriate abstract inferences about international economics" (op. cit. p. 42).

> "In our theory, metaphors such as the Event Structure Metaphor (Lakoff, 1994) use the dense and familiar causal structure of spatial motions and object manipulations to permit reflex (fast, unconscious) inferences. Conventionalised projections from motion features allows the speaker to use terms that rely on the highly familiar structure of spatial motion and object manipulation to communicate complex scenarios, dynamically changing goals, resources, monitoring conditions, and communicative intent" (Narayanan, op. cit. p.9).

These recent works make the theory of Experientialism more robust, with solid basements in order for us to use it in our purpose of methodological goals, as *methodology* means –in its pure semantic origin- the *study of methods*.

1.2 Relations with Other Approaches

In the last 15 years, there have been some other projects for establishing an applied science of Human-Computer Interaction grounded on different approaches, more or less close to cognitive sciences. One of them has been published in Carroll (1987, 1991) and other are represented by the Ecological Approach (Gibson, 1986) and by Activity Theory (Nardi (1996a, 1996b, 1996c), Kaptelinin (1996a, 1996b), Engeström (1996, 1999), Kuuti (1996), Bødker (1991, 1996)).

In fact, what differentiates Experientialism is its explanation of cognitive processes as derived from bodily –physical- processes and not from an inner mental capacity of people to capture concepts and elaborate theories. The theories based on such innate capacity of human beings derive from rationalistic grounds, whereby the division between body and mind, objective and subjective, is the common and general framework.

The whole Western rationalistic tradition is based on such an assumption, the Cartesian idea that the proof of our existence is our capacity to think. What Experientialism question is this rationalistic assumption whereby thought has a primacy over the body, pointing out that it is the opposite that is valid. One book's title is significantly graphic in its expression, "Descartes' Error" (Damasio, 1994), and some of the main ideas of the author are in line with those of Experientialism:

> "The lower levels in the neural edifice of reason are the same ones that regulate the processing of emotions and feelings, along with the body functions necessary for an organism's survival. In turn, these lower levels maintain direct and mutual relationships with virtually every bodily organ, thus placing the body directly within the chain of operations that generate the highest reaches of reasoning, decision making, and, by extension, social behavior and creativity" (op. cit. p. xiii).

What is important in Experientialism, in our viewpoint, is the amount of evidence that such an approach brings to show how the mind is mainly a bodily construction or, in other words, that there is a continuous chain of operations between bodily organs and reasoning. This important idea is underlined by some of its proponents in titles such as "The Body in the Mind" (Johnson, 1987) or "Philosophy in the Flesh" (Lakoff & Johnson, 1999).

There is no something like a pure mental essence, which is pre-existent to any bodily interaction with the real world. Just the opposite is true: mental processes derive from our interaction with the external reality in all its aspects: material, cultural and social.

1.2.1 Cognitive Psychology or Cognitivism

The term *cognitive* usually refers to cognitive sciences in general -new scientific discipline based on a group of other disciplines such as linguistics, philosophy, psychology, computing, etc. But cognitive –in its classical sense- also refers to an isolated individual perspective, as a consequence of the main role that one of its constituents, psychology, has played in cognitive sciences. That is why it is usual to speak of cognitivism, which imposes an individual point of view, where we try to understand how an individual subject is able to conceptualise the external world. When used in this way, cognitive is usually understood as *cognitive psychology*.

The result of trying to integrate the classical cognitive approach into the field of Human-Computer Interaction (HCI) produced a solution based on the assumption that both human beings and computers can be considered as information processing units (Kaptelinin, 1996a). Both cognitive psychology and computer sciences offer a mirror image to each other: they reaffirm their status when considered together, in a tautological process.

This is the main reason why we have not employed the classical cognitive psychology approach to study the phenomena we consider in this thesis.

What traditionally has been used in cognitive psychology is the concept of mental models, a mental equivalent –which is isomorphic, in the idea of its supporters- to the objective reality. So the cognitive processes taken into account when trying to analyse a HCI task are the reflexes of the already existing –physical- phenomena. In this extended approach mental models are ready-made artifacts, or clones of objective tasks we are going to perform. This approach is a variant of the Objectivism folk philosophy.

As opposed to Experientialism, Objectivism offers us a vision of the real world as populated by objects with attributes that are objectively existents, with the only need for human beings to observe and detect them. Unlike Objectivism, Experientialism explains that object attributes are the consequence of interactions between objects and human beings; attributes that are fundamentally *interactional.*

In this thesis we use the word *cognitive* in a broader sense. It is the same sense used, for example, as in cognitive semantics. In cognitive semantics, as we will see in Chapter 2, it is mainly linguistics that takes the central role, and linguistics is the interface between the individual and the social world. The concept of *cognitive*, unlike "cognitivism", becomes a description of general processes that apply to any individual as a consequence of being incorporated in a social world through language as well as social and cultural experience. This is the reason why cognitive semantics can also be described as Experientialism.

One of the main contributions of cognitive psychology to HCI is the concept of mental models, whose underlying conceptual framework is based on the contemporary Anglo-American philosophy of language. When considering the concept of mental model, we have a wide spectrum of ideas, ranging from a classical functionalist approach to another closer to Experientialism.

For example, in terms of Williams and others (1983), "an autonomous object is a mental object with an explicit representation of state, an explicit representation of its topological connections to other objects, and a set of internal parameters...A mental model is a collection of "connected" autonomous objects. Running a mental model corresponds to modifying the parameters of the model by propagating information using the internal rules and specified topology" (p. 133).

This is a prototypical approach based on the metaphors *Thought as Object Manipulation* and *Thought as Language. Thinking*, according to these metaphors, is object manipulation, the same as *communicating* is sending or the structure of a thought is the structure of an object. Another assumption of this approach is that the structure of thought is accurately representable as a linear sequence of written symbols of the sort that constitute a written language. (Lakoff and Johnson, 1999, p. 249)

An intermediate position in the spectrum is Norman's (1983) point of view. According to this author, mental models are:

"1) incomplete

2) People's abilities to "run" their models are severely limited.

3) Mental models are unstable: People forget the details of the system they are using, especially when those details (or the whole system) have not been used for some period.

4) Mental models do not have firm boundaries: similar devices and operations get confused with one another.

5) Mental models are "unscientific": People maintain "superstitious" behaviour patterns even when they know they are unneeded because they cost little in physical effort and save mental effort" (p. 8).

Norman offers a criticism of mental models, which is based –at the same time- on a more complex conception of mind: mental models are not recipes or procedures written in the mind.

There is also an underlying idea (point 5 above) that mental models are some external conceptual artifacts –instead of internal objects- that users may adopt or refuse.

When he says that 'people maintain superstitious behaviour patterns even when they know they are unneeded', it is clear that people have consciously adopted such behaviour as something external and convenient as it saves physical and mental effort. In this case, it is suggested that mental models are not the result of the interaction between the user and a system, as usually HCI suggests, but of a given external representation of the system.

The last example of mental model is offered by Gentner and Gentner (1983) and is close to Experientialism, as a consequence of authors wanting to explore the conceptual role of analogy. "If analogies are to be taken seriously as part of the apparatus used in scientific reasoning, it must be shown that they have real conceptual effects" (p. 99). They point out that people sometimes use implicit analogies –this is a clear case of using metaphor- in which "the person seems to borrow structure from the base domain without noticing it. Phrases like 'current being routed along a conductor,' or 'stopping the flow' of electricity are examples" (op. cit. p. 99).

The concept of mental model therefore –as most of concepts used in different domains- is not homogenous but embraces a set of multiple theories. Each theory is based on different assumptions and varying from a classical approach (Williams, 1983) based on the metaphors of *Thought as Object Manipulation* or *Thought as Language*, to an approach (Gentner and Gentner, 1983) that recognises the important role of metaphor in mental models.

That means that our understanding of what mental acts are is fashioned metaphorically in terms of physical acts like moving, seeing, manipulating objects and so on. The problem is that "such metaphors hide what is perhaps the most central property of mind, its embodied character...The world *mental* picks out those bodily capacities and performances that constitute our awareness and determine our creative and constructive responses to the situations we encounter. Mind isn't some mysterious abstract entity that we bring to bear on our experience. Rather, mind is part of the very structure and fabric of our interactions with our world" (Lakoff and Johnson, 1999, p. 266).

1.2.2 Ecological Reality

It is interesting to observe the coincidence between some –basic- aspects of *Experientialism* and Gibson's (1986) *Ecological Approach.*

In Gibson's terms, the world of physical reality does not consist in meaningful things. We are in face of a meaningful world when we consider the world of *Ecological Reality*. If an environment refers to the surroundings of those organisms that perceive and behave, the set of environments -or Ecological Reality- refers to the physical world with all the organisms living in it.

As can be seen, the concept of 'Ecological Reality' implies an interaction between human beings –and other organisms- and the external physical reality. A meaningful world, is a world where human beings have already marked it with a whole set of signs, derived from the set of activities performed by human beings. It is rather an external physical reality modified –transformed- by human actions that counts for us, and it is this transformation that bring us a *meaningful* world.

This external physical reality is the equivalent to what Lakoff calls an external, mind-free reality. Such a reality is the construction of the objectivist philosophy whereby an objective –pre-existent- world is assigned meaning through arbitrary symbols.

Neither Gibson nor Lakoff agree with the objectivist approach, as meaning is generated during the interaction between bodily human beings and such an external reality. Moreover, the concept of physical reality would be a construct derived from our more concrete conceptualisations about the real world. It is not prior information to be taken into account in order to know what meaning and knowledge about the world are. "If what we perceived were the entities of physics and mathematics, meaning would have to be imposed on them" (Gibson, 1986, p. 33). This last statement is equivalent to what experientialism considers an objectivist philosophy: objective world is assigned meaning through arbitrary symbols.

In a similar way, Lakoff consider that «what is needed to replace the objectivist view of meaning is an irreducibly cognitive semantics, one that accounts for what meaning is to human beings, rather than trying to replace humanly meaningful thought by reference to a metaphysical account of a reality external to human experience» (1988, p.120)

Gibson considers that internal representations are not static, localised entities but are fluid, distributed structures in brains that are derived by 'in-forming'- That is, their contents are derived by actions of the body as the agent of the mind, which extract information from objects in the world through their relations to the agent in the form of affordances.

> "One sees the environment not with the eyes but with the eyes-in-the-head-on-the-body-resting-on-the-ground. Vision does not have a seat in the body in the way that the mind has been thought to be seated in the brain. The perceptual capacities of the organism do not lie in discrete anatomical parts of the body but lies in systems with nested functions" (Gibson, 1986, p. 205)

This point of view is also totally consistent with Experientialism, which focuses on two aspects that were ignored by the objectivist approach: 1) the role of the body in characterising meaningful concepts, and 2) the human imaginative capacity for creating meaningful concepts and modes of rationality that go well beyond any mind-free, external reality. (Lakoff, 1988).

The Ecological approach considers that properties belonging to objects afford an agent the opportunities to use them to achieve its individual goals, through which the agent establishes relations with those facets of the world with which it can interact and from which it can learn. "By virtue of these relations, the mind is not restricted to the brain or body but extends into the world and the mind is a seamless fabric of inner and outer experience" (Freeman and Núñez, 1999, xiv)

A very similar idea to that expressed by Lakoff and Johnson (1999, p. 266) when they say that mind is part of the very structure and fabric of our interactions with our world.

These ideas constitute what Gibson calls the theory of *affordances*. «The fact that a stone is a missile does not imply that it cannot be other things as well. It can be a paperweight, a hammer, or a pendulum bob...These affordances are all consistent with one another...The theory of affordances rescues us from the philosophical muddle of assuming fixed classes of objects, each defined by its common features and then given a name. As Ludwig Wittgenstein knew, you

cannot specify the necessary and sufficient features of the class of things to which a name is given. They have only a "family resemblance"» (Gibson, 1986, p.134)

The theory of affordances rescues us from objectivism, the same way as Experientialism does. For example, *basic level categories* defined by Lakoff –classes of objects- are derived from the level at which human beings interact with their environment most effectively and process and store and communicate information most efficiently. An important component of Experientialism is aimed at analysing how categories are built in general activities and are not coincident with the classical idea of fixed class objects with shared characteristics.

There are other two similar concepts: affordances of the environment and experience.

> "It is a mistake to separate the natural from the artificial as if there were two environments: artifacts have to be manufactured from natural substances. It is also a mistake to separate the cultural environment from the natural environment, as if there were a world of mental products distinct from the world of material products. There is only one world..." (Gibson, 1986, p.130).

The Ecological approach considers that the most elaborate affordances of the environment are provided by other people. In terms of affordances, behaviour affords behaviour, and «the whole subject matter of psychology and of the social sciences can be thought of as elaboration of this basic fact» (op. cit. P.135)

Experientialism considers that "experiential" is to be taken in the broad sense, including basic sensory-motor, emotional, social, and other experiences of a sort available to all normal human beings –and especially including innate capacities that shape such experience and make it possible. «The term "experience" does not primarily refer to incidental experiences of a sort that individuals happen to have had by virtue of their unique histories. We are focusing rather on that aspect of experience that we have simply by virtue of being human and living on earth in a human society» (Lakoff, 1988, p.120)

1.2.3 Activity Theory

Activity Theory has been presented as a potential framework for HCI research but, in our understanding, it has some limitations for offering a detailed method for User Interface design. Both Experientialism and Activity have some points in common. Both consider that cognitive processes derive from the interactions between human beings and material elements of reality, and between human beings and each other in social and cultural relationships.

The main difference between both approaches is the emphasis on a social or individual aspect of human cognition. Activity Theory emphasises the social character of any human activity but recognising at same time the need to include some *subjective* aspects to the theory, as in the open problem of externalization/internalization (the interface between external and internal –mental- processes).

Experientialism focuses on a subjective experience and tries to explain it in terms of conceptual metaphor, something that is pervasive in both thought and language. For example, we apply a conceptual metaphor when we conceptualise *understanding* an idea (subjective experience) in terms of *grasping* an object (sensorimotor action).

As these conceptual metaphors are encapsulated in everyday language –as when we say that somebody *has a good grasp of mathematics*- there is –in addition to the subjective experience- a social –shared- experience that gives the conceptual metaphor an *objective* dimension. We must take into account that *subjective* experience –in terms of Experientialism- is an easy and useful term to refer to a quite complex process; the criticism stated by Experientialism regarding the opposition between *subjective* and *objective* shows clearly that *subjective* experience may not be considered in a traditional mode, the same way that we think of mind *as part of the very structure and fabric of our interactions with our world.*

Activity Theory considers that instead of considering only a single individual, it has been recognized that features of cooperation, communication, and coordination are often vital in the successful performance of tasks. "Thus HCI research seeking practical relevance cannot restrict itself to the study of individual acts" (Kuutti ,1996, p. 21).

The problem with Activity Theory –considered as the Soviet originated cultural and historical research tradition- is that such approach in neither interested in activities in general nor is it a theory, that is, a fixed body of accurately defined statements.

For example, one of the issues considered by Activity Theory is the transformation from *operations* -routinely unconscious acts, like in driving- to *actions*, which are movements we are aware of in the context of an activity.

This useful distinction may be applied to our Design Activity in order to move from a classical design approach of *operations* to a new one of *actions*. That means that we may access a higher degree of conscious decisions by using additional knowledge about our own cognitive capacities: trying to integrate the main ideas of Experientialism (metaphor, mental spaces, conceptual integration and stories) into HCI methods. This is an important issue as one of the basic results of Experientialism is that most of our cognitive processes are unconscious, not in the Freudian sense of repressed, but in the sense of processes laying under our consciousness level.

In the frame of activity theory both processes, concept generation on the one hand, and software engineering on the other, are activities in their own. The former activity creates the *cognitive artifacts* to be used in the latter, but both are intertwined. Cognitive artifacts have to be changed as problems derived from their use emerge, as the history of most software engineering methods –and technology, in general- shows.

Carroll (Carroll et al, 1991) stated a similar idea in terms of a task-artifact cycle: "The evolution of HCI technology is a coevolution of HCI tasks and HCI artifacts". What Activity Theory proposes is a larger framework, in which context is included so that "HCI should not be a closed system of 'user-computer' but should include the meaningful context of the user's goals, environment, available tools, and interactions with other people". (Kaptelinin, 1996a, p. 49).

There is another issue that Activity Theory tries to solve but has not successfully explained yet: the internalisation/externalisation mechanism that people experience while performing activities. Regarding this issue, we have to take into account the fact that even Activity Theory suffers the consequences of a traditional approach when referring to the *externalisation/internalisation* process.

It is evident that the terms *external* and *internal* refer to a subject's mind, to a transformation process going *from the activity to the mind of a person* or team and vice versa. This traditional approach is based on the metaphors of mind that conflict with what cognitive science has discovered.

> "We conceptualize the mind metaphorically in terms of a container image schema defining a space that is inside the body and separate from it. Via metaphor, the mind is given an inside and an outside. Ideas and concepts are internal, existing somewhere in the inner space of our minds, while what they refer to are things in the external, physical world. This metaphor is so deeply ingrained that it is hard to think about mind in any other way" (Lakoff and Johnson, 1999, p. 266).

As usually occurs in any approach, other authors have different ideas about internalisation. For instance, Kaptelinin points out that the "most fundamental principle of activity theory is that of the unity of consciousness and activity. 'Consciousness' in this expression means the human mind as a whole, and 'activity' means human interaction with the objective reality" (1996b, p. 107). Comparing Kaptelinin's quotation with Lakoff and Johnson's (*mind is part of the very structure and fabric of our interactions with our world*) we deduce that both approaches share similar underlying assumptions: Kaptelinin's *consciousness* and *objective reality* is equivalent to Lakoff and Johnson's *mind* and *our world*, respectively.

The main difference is on focus: while Activity Theory tries to study a unit, called an activity, Experientialism intends to study the consequences of very general human activities (like a baby dragging himself on the ground, or grasping a ball, or entering into a room, etc) which determine mental patterns and processes like image schemas, metaphor and more complex conceptual integration.

Unlike Activity Theory, Experientialism can offer a base for building some *cognitive artifacts*: concepts like image-schemas, metaphor, blends, and stories are the raw material we may use to construct such cognitive artifacts. These new concepts may be used in HCI methods (Chapter 3), as computer based signs employed in the elaboration of user interfaces (Chapter 5 and 6) or the specification of requirements in terms of user and abstract stories (Chapter 7). As Hutchins (1999) explains, "cognitive artifacts are always embedded in larger sociocultural systems that organize the practices in which they are used. The utility of a cognitive artifact depends on other processes that create the conditions and exploit the consequences of its use".

Finally, it is worth saying that what we are proposing about creating new cognitive artifacts is possible because HCI –and cognitive sciences- have got a minimum degree of maturity. Historically, things happen like this: an accumulation of new facts and problems associated to an older theory (artifact) determine the production of new theories that explain such phenomena. Carroll (op. cit. p. 79) points out that "the original direct manipulation interfaces owed little or nothing to scientific deduction, since in fact the impact went the other way round. Analogous points could be made regarding other innovations in HCI, for example, spreadsheet systems and object-oriented programming. Indeed, it seems typical in HCI for new ideas to be first codified in exemplary artifacts and only later abstracted into discursive descriptions and principles".

What we propose is the elaboration of cognitive artifacts aimed at building new *exemplary artifacts*. Note that the word artifact has a large meaning, but here it would be enough to state that Carroll's *exemplary artifacts* are, in our terms, *computer-based signs*. We try to detect

such facts in HCI in order to build the basis of a theoretical framework that could enlighten our perception of the design practice.

1.2.4 Postmodernism

One of the main criticisms about Experientialism is provided by Postmodernism. This post-structuralist philosophy -based on some French scholars like Derrida and Lyotard- has some points of view specifically regarding metaphor. Some Derrida's general ideas may agree with Experientialism, such as his focus on language: while Derrida is logocentric in the ordinary use of the word -he focus on words- Experientialism is also *cognitive semantics* or *cognitive linguistics*.

When considering logocentrism in a strong sense, Derrida aims at criticising it, as it may be considered as an equivalent of *reason* –it is the old sense of the word *logos* (word)- which also suggests the presence of a grounding principle (Coyne, 1995).

Logocentrism is supposed to be –in this approach- the quest for the ground, the origin, the substrate on which all knowledge, understanding, and being is built. That is why Derrida's project is to go beyond logocentric argumentation, not simply denying that there is a ground –to deny would also be a metaphysical position, implying the need for a nonground- but attempting to deconstruct its own rethoric.

So, as Coyne explains, there is a need to work out a nonmetaphysical view of the topic at hand –language, literature, art, science, etc- and, in particular, in relation to design and technology.

In order to criticise the topic at hand, postmodernism proposes to use the strategy of *deconstruction.* Deconstruction is an argumentative strategy to unsettle and challenge metaphysics or logocentrism. The strategy is similar to an old one –dialectic- used by pre-Socratics as well as by modern philosophers like Hegel, Heidegger or Marx, but is has been transformed using a metaphor: *to argument is to play.* Other authors –like Caputo (1987)- have called this strategy *radical hermeneutics*, as hermeneutics is the study of interpretation or a search for meaning.

There are some similarities with Experientialism. Postmodernism recognises that in language there has never been a close coupling between signifier and signified: as images and data, signifiers appear to refer to other signifiers (other images and data) in an endless chain. This endless chain of signification is not an unfortunate accident of language but "a constitutive element of its structure" (Culler, 1985).

Experientialism, on the other hand, explains meaning in terms of a network of mental spaces, each one referring to other. It is the unfolding of a set of mental spaces that constitutes the meaning and not something contained in words as suggests the idea of a signifier (container) and signified (contents). The difference with Postmodernism is that the chain of mental spaces is not an *endless one*, but grounds on the bodily –and social and cultural- experience.

What is important to our aim is the application of deconstruction to the ideas of Experientialism. In Coyle terms: What does Derrida have to say about the bodily basis of

thought? The main problem with Lakoff and Johnson's ideas –according to Coyle- is that they privilege the body without apparently acknowledging the contradictions and reversals this entails.

> "To trace language and metaphor use back to the immediate realm of bodily experience is a metaphysical exercise" (Coyle, 1995, p. 273).

Fortunately, it is recognised in the same text that Lakoff and Johnson are writing in a scientific and analytical context rather than a philosophical or literary one and that "their thesis about the bodily basis of metaphor opens up avenues for exploration, and the ultimate arbitrator of the validity of these ideas is their usefulness" (op. cit., p. 274).

As Coyle's book was published in 1995, he could not anticipate some later results like Narayanan's –his thesis was presented in 1997- showing that the same neural mechanism that can control bodily movements can perform logical inferences about the structure of action in general (Lakoff and Johnson, 1999, p. 42).

Even if Lakoff and Johnson enterprise is a scientific one, we can never avoid some philosophical assumptions or implications, as when Experientialism criticises the objectivist philosophy. The second argument we may use is that Lakoff (1988, p. 123) recognises himself as adhering to a basic realism –a basic metaphysical assumption-, but have we to take this issue as a contradiction or –on the contrary- the premises for any type of scientific work?

Finally, I think that we cannot completely eliminate any kind of contradiction or metaphysical assumption in our scientific work and this fact is, the same way as Postmodernism posted it, *a constitutive element of its structure*. Moreover, *the ultimate arbitrator of the validity of these ideas is their usefulness*, but this does not imply that the truth is based merely on internal coherence.

Basic realism means a commitment to the existence of a real world, which is twofold: a world external to human beings and the reality of the human experience. But it means also the existence of a link of some sort between human conceptual systems and other aspects of reality. Even if Postmodernists try to eliminate any type of metaphysics in their arguments, in fact when they argue they are assuming a metaphysical existence of a link between their conceptual system and other aspects of reality; on the contrary, arguing is not worthwhile.

1.3 Experientialism as Cognitive Foundations of HCI

This delimitation of meaning is a first application of the ideas we are going to introduce in this thesis: cognitive semantics –or Experientialism- is concerned with human concepts as the basis to meaning, rather than with truth-condition as the basis of meaning (Turner, 1994). In order for us to understand an expression, we successively unfold mental spaces where each of them have their own frames and structure. It is this unfolding of successive mental spaces that allows us to comprehend the semantics of the given expression, so we are going to unfold the concepts included in the title.

When we think about foundations of a given domain (normally a scientific discipline), we usually think of some type of axiomatic concepts, whose validity is absolute, or the equivalent of Kantian *a priori* concepts. *A priori* concepts are those that are not empirically determined, they are formal concepts with a universal validity. Indeed, it is the primary sense of the word's

foundations, the parts of a building that form its base, which are, in this case, *metaphorically* extended to theories.

In mathematics, the search for complete theories -in this case theories that would integrate their foundations by means of the demonstration of its own non-contradiction- has been given up after the famous Gödel's theorem about incompleteness.

The equivalent of this incompleteness appears when analysing the foundations of a given domain. It becomes clear that there are no privileged domains within a collection of priori concepts, which could be used as a general foundation. The situation, instead, is represented by the following figure:

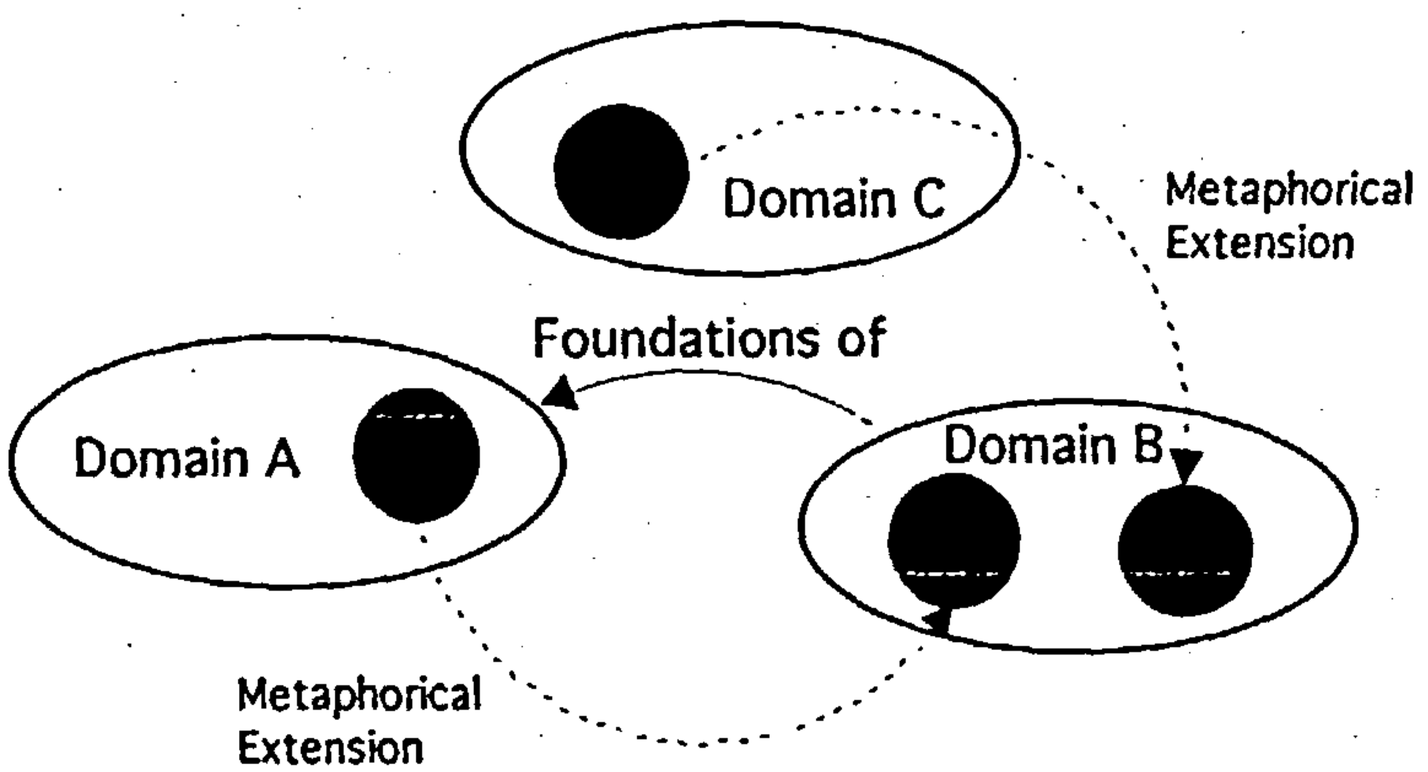


Figure 1 Foundations are not unidirectional

In this oversimplified example —, Domain B is used as a foundation of Domain A. But Domain B has a sub-domain which is a metaphorical extension of sub-domain of Domain A. At the same time, Domain B uses other metaphorical extensions of other domains, in this case, Domain C. For example, if we decided to use Cognitive Grammar (Langacker, 1987 and 1991) as foundations of Object-Oriented Methods (Computing Domain), in the former we would detect some metaphorical extensions of the latter like *computation* , *cognitive routine* , etc. At the same time, Cognitive Grammar has other metaphorical extensions from other domains (*valence relation* from chemistry; *scanning* and *composition* from editing, *focusing* from optics, and so on) or Mental Spaces referring to an spatial –physical- metaphor.

That shows that even if we are to use Cognitive Grammar as a tool for constructing Human-Computer Interaction foundations, Cognitive Grammar is by no means a *pure*, uncontaminated domain where other -less literal- conceptual domains may be founded. As we have to evaluate the explicative power of a given domain –in this case Experientialism- we do not care about its possible contamination. This, on the contrary, is a demonstration of the validity of some results of Experientialism: our conceptual categories are networks of mappings between different spaces.

Another interesting example is offered by Turner (1994): "In computing, a start-up program gets us going. This book is like a start-up program, designed to get us going in the activity of

reconsidering an extremely influential group of default concepts, which is to say, a group of concepts we fall back on automatically when we do not engage in the arduous work of considering the fundamental ground of our thought and action" (p. 25). While we will use Turner's work as a basis for Human-Computer Interaction foundations, Turner *in turn* uses computing –the start-up program metaphor- as the basis for his study of mind, language and literature.

One consequence from this discussion is that we have only relative foundations, metaphorical extensions and projections *from* other domains and *to* another domains. The usefulness of this search for foundations can be found in the extensions and projections themselves: when we are aware of the type of operations underlying our thoughts, we can try to apply a different focus in the same extensions or to try other extensions in order to get a deeper knowledge of the target domain. The exercise of working on the extensions and projections -or whatever the cognitive processes may be- becomes more systematic when such foundations are clearly shown.

As can be deduced, foundations means an interplay between many domains in which we try to detect connections that could reveal new aspects of our activity as designers. It is not the search for a proto-domain, which could be used to explain all other disciplines. Foundations is the analysis of relationships in order to detect new meaning, or, in other words new viewpoints of analysed domains.

As a last remark, it is important to note that the use of metaphors like *the Mind as Body*, *the Mind as Machine*, *Thought as Motion*, *Thought as Object Manipulation*, *Thought as Language* etc, and philosophies based on such metaphors is so pervasive that it is near impossible to prevent us for using them. Indeed this thesis may have some arguments originated on such classical metaphors, so they have to be detected and interpreted in the light of Experientialism.

1.4 Thesis' Contributions

Among the main contributions of the thesis, we can emphasise the following:

1.4.1 To show where main HCI and Software Engineering concepts come from

We constantly use in other domains of our everyday activities basic cognitive processes –image-schemas, metaphors, blends, metonimies, etc-. If we analyse the history of software engineering concepts, we can observe that the ground of them is always the same type of schemas –image schemas-: containers, source-path-target, links, etc. Even the original computing concepts like records are conceived as containers: they *contain* 'fields', and fields *contain* data. The reason is given by Experientialism when posits that concepts are of two sorts that are meaningful because of their roles in bodily experience (especially movement and perception):

- Basic level concepts
- Image-schemas

We use basic level concepts to categorise things for which we have mental images and general sensorimotor programs, things we usually call *concrete* things. In the case of abstract concepts (like entities, relationships, etc) we have to use image-schemas.

1.4.2 To show that most of the language used in HCI is figurative

The idea that our scientific or technical discourse must be 'literal' obeys an objectivist assumption, whereby we use language to refer to 'things' of the 'real world'. *Literal*, in the case of computing –an abstract discipline- might be considered as a language that is precise, linear and logical as opposed to a diagrammatic or verbal reasoning (Goguen, 1993b). Even supposing that such a *referential* idea –language *refers* to things of the real world- would be correct, when designing a new software system we are not referring to things that already exist. We are trying to depict some features, which do not exist yet, so the use of a figurative language appears not only as logical or useful, but as indispensable. After refining a design model we are in position of using a more precise, linear and logical reasoning.

The complex subject of literal language grounds on the traditional view of literal meaning based on Frege's principle of compositionality. Frege's position was that a large number of sentences in a natural language can be understood by a competent speaker/hearer without knowing who said the sentence, where it was said, or when or why. In other words, the interpretation of many sentences is independent of knowledge of extralinguistic context (Gibbs, R. Jr, 1994). In terms of Gibbs,

> "the literal meaning cannot be *uniquely* determined, since our understanding of situations will always influence our understanding of sentences. To speak of a sentence's literal meaning is already to have read it in the light of some purpose, to have engaged in an interpretation. What often appears to be the literal meaning of a sentence is just an occasion-specific meaning where the context is so widely shared that there doesn't seem to be a context at all" (op. cit, p. 71).

1.4.3 To highlight the importance of blends in HCI and to promote its use in a disciplined way (in the form of patterns) in order to build computer-based signs

In Chapter 5, we propose to apply the concept of blend as a structured pattern, and the purpose is to make conscious a process that normally is performed unconsciously. Blends are source of creativity in literature and in science (complex numbers theory or the desktop interface show this important feature of blends), although Turner has shown to what extent the 'scientific' thought is also 'literary'- but we need some additional cognitive devices for using blends in a disciplined way.

1.4.4 To show the limitations of Experientialism when applied to User Interface design

We consider that the main limitation of experientialism is the high level of abstraction of the concepts it provides. Even if this issue may be considered as a limitation –we would have liked to make use of an approach that leads us through the complete lifecycle of the User Interface design- at the same time this apparent weakness might be considered as its strengths. The positive aspect of figurative language or metaphors in general is that they are a source of creativity: new ideas and solutions are offered to our design problems. This is an important trade-off between high level languages, that only offer a solution at a conceptual level, and more precise languages that are usually driven by technological –or domain specific- aspects. That is why we propose to divide the process of development of the User Interface (Chapter 5) into phases the same way we proceed for developing software systems in general. This division

has the advantage of considering the developing process for the interface in a way that is consistent with the process of developing system software: an *analysis* where it is built a conceptual model and a *design* where the conceptual model is refined to build a logical model.

1.4.5 To use stories as the raw material for constructing requirements

The statements we use about a given domain are a set of stories that make an account of such a domain. At the most general –'abstract'- levels, are a set of stories that allow us to think about a given mental space in a more detailed manner

1.5 Thesis layout

Chapter 2 reviews the background to experientialism and introduces the key concepts. A brief discussion of where experientialism has been applied is provided. This chapter does not aim to present a detailed critique of experientialism, rather it explores the concepts in an 'axiomatic' fashion. The thesis seeks to demonstrate the utility of the concepts arising from experientialism, not to engage in a debate about their general validity. Despite this fact, this chapter points out the main limitations of the approach regarding the design of User Interfaces, indicating that it is mainly in the analysis level that Experientialism may be successfully used.

In Chapter 3 we show how most of elements of the HCI methods notation are derived from the main concepts proposed by Experientialism, mainly image-schemas, metaphoric projection and blends. Realising where notations and concepts have come from enables designers to see the strengths and weaknesses or each one and enables designers to see where changes in the underlying assumption leads to changes in the way that HCI is viewed. In a similar way the top-down design approach is reviewed along with the various ways in which system requirements may be seen. Different ways of looking at HCI are considered

Chapter 4 explores the concept of metaphor in HCI. This has long been a controversial aspect of HCI design with strong feelings on both sides of the argument as to whether metaphor should or should not be employed at the human-computer interface. The answer turns out to be that the question is meaningless. It is not possible to avoid metaphorical interfaces; we think through metaphor and through using metaphors in blends. The chapter looks at metaphors at different levels of description and at how metaphors related to a semiotic view of HCI, in order to be used in the next chapter. An important outcome of this analysis is that HCI is inherently figurative; designers cannot rely on a 'literal' interpretation of the signs that they create. Some examples of figurative language in HCI are offered in order to show the extent to which such expressions are usual in technical language.

Chapter 5 presents a method for design based on an experientialist view. The method is presented as a generic 'pattern' that illustrates how knowledge of the underlying metaphors and image schemata can be utilised in the analysis and design process. The process of 'conceptual integration' is described and applied at two different levels: analysis (or conceptual modelling) and design (or computer-based signs design). The difference between analysis and design is also used to explain why the criticism about the famous trash can icon on the Macintosh is, in fact, a criticism to a design decision and not a way of questioning the important role that metaphor plays in User Interface design.

Chapter 6 provides an example of design based on the pattern presented in Chapter 5. Whilst avoiding the prescriptive approach of some methods, it lays out the important criteria

that need to be considered in the design of an interface. The approach is illustrated with an example from the design of a home information centre device. This example is used to show how the limitation of the experientialist approach –suitable to be applied mainly at a conceptual level- is at the same time source of new ideas, of creativity.

Chapter 7 extends the experientialist analysis to the concept of stories. Turner has argued that stories are an essential cognitive process. In HCI they appear in a variety of guises such as use cases, 'context of use' and scenarios. The basic story concept is important within the HCI context because stories are fundamentally about interactions. The utility of the story concept is illustrated with an example

Chapter 8 provides the conclusions to the thesis. It is the utility of the ideas that is important. The main contribution of the thesis is to make the abstract concepts of an important new view of cognition available to HCI designers. The thesis puts an end to the futile debate about the role of metaphor at the interface by demonstrating that designers have to make use of metaphors. The important thing is that they are aware of how to make use of metaphor and related concepts such as conceptual integration and idealised cognitive models. Designers create interactions, classify functions and design computer-based signs. The methods and notations they use allow them to focus upon and express ideas within the constraints of the underlying image schemata. The ways in which they classify things are derived from how they project from the perceptual into the cognitive. The design 'pattern' presented shows how the major features of the spaces that feed into the interface can be considered in a rational fashion and where the 'raw material' for design comes from.

Chapter 2

Cognitive Background

2.1 Objectivist Cognition

Cognitivism employs the metaphor of computing by a machine and applies it to the human mind. Both domains, those of the mind and the computer serve, as foundations to each other. This way, we think of the mind as if it is computer and the reverse is also valid: a computer works as the human mind; each one reinforces the other. In fact, the rational thought is conceived as an algorithmic manipulation of arbitrary abstract symbols. These symbols have not meaning by themselves, but get their meaning by being associated with things in the world; this is the *referential* side of the approach. Cognitivism is a consequence of a more extended framework regarding cognition: *objective cognition.*

This idea of objective cognition is deeply rooted in our conceptualisation of cognition. As Mark Turner explains, it should be referred to some classical Greek philosophers' ideas:

> "The distinction between objective and subjective meaning, which the updated premise of Protagoras asks us to cross off our list, seems to derive from a conceptual metaphor in which we understand mind in terms of a container. In this basic metaphor, meaning is an object. Objective meaning is, metaphorically, located outside the mind-container. Metaphorically, we can have inside the mind-container a copy of this external objective meaning. If the copy is good, then we know objective meanings. We can all have copies, connected by virtue of their all having an external referent. In this metaphor, these internal copies cannot be connected to each other directly. They can be connected only through the intermediary of the objective referent, which is located outside the brain, and which is independent of the copies, and which has priority over the copies." (Turner, 1994)

Once symbols have been created -in a historical development or as an arbitrary construction- they are manipulated in the processes that take place in the mind, in order to obtain new symbols as results.

This algorithmic manipulation of symbols has been called a *language of thought* or *mentalese*. As symbols do not have meaning by themselves, the rules that manipulate them do not need to use such meaning: so rules are universal and apply to any type of symbols. The twofold aspect of the objectivist theory implies that mental processes are algorithmic in the mathematical sense and that arbitrary symbols can be meaningful only by being associated with things in the world (Lakoff, 1988).

As a consequence of this symbolic representation, objectivism "tr[ies] to analyze meaning, truth and reason without mentioning nonpropositional structures such as images, schematic patterns, and metaphorical projections, which are considered components of understanding, but not essential to meaning in the 'proper' sense". (Johnson, 1987, p. 18)

We speak of objectivist cognition considering it has a broader scope than the 'functionalist hypothesis', which states that a physical symbol system in a computer is 'functionally equivalent' to the operation of the human brain.

The objectivist approach determines a conception of knowledge whereby it involves a set of symbols that apply to a given domain of the world, the rules that manipulate them, and the set of mappings that give meaning to such symbols. As Lakoff (1987) points out, objectivist

> "... knowledge consists in correctly conceptualizing and categorizing things in the world and grasping the objective connections among those things and those categories".

Such objectivist theory gave rise to the idea of implementing expert systems and validates it as an equivalent for human knowledge. Why are we not using those large quantities of knowledge stored in the heads of experts and specialists that could allow the rest of ordinary people to reason as they do? It is only question of time and resources, and all the accumulated knowledge could profit everybody, as "computer scientists work with individual experts to explicate the expert's heuristics -to mine those jewels of knowledge out of their heads one by one...the problem of knowledge acquisition is the critical bottleneck in artificial intelligence." (Feigenbaum and McCorduck, 1983).

Stating the problem as dealing with a *bottleneck in knowledge acquisition* is a wrong and misleading one. This idea is equivalent to think of knowledge as a large amount of already formed concepts and relations through a narrow communication channel. The bottleneck is the result of having a large amount of knowledge (concepts) and a narrow channel: as we will see in next chapters, this is the application of a wrong metaphor to the concept of knowledge (container, contents, flow of concepts and so on). This metaphor "seriously misconstrues the theory formation process of computer modelling" (Clancey, 1989).

2.2 The objectivism does not work as well as it did.

Objectivism –called by other authors *metaphysical realism*- does not limit itself to a basic realism, that is, a commitment to the existence of a real world, external to human beings and the basis to human experience; to a concept of truth that is not based merely on internal coherence and a commitment to the existence of stable knowledge of the external world.

Objectivism is based on four interrelated doctrines that state:

1) The world consists of entities with fixed entities and relations held among them at any instant. This structure is mind-free, that is, independent of the understanding of any beings.
2) The entities in the world are divided naturally into categories called natural kinds. All natural kinds are sets defined by the essential properties shared by their members.
3) All properties are either complex or primitive; complex properties are logical combinations of primitive properties.
4) There are rational relations that are held objectively among the entities and categories in the world (Lakoff, 1988, p. 124)

One of the main fields where objectivist theory could be applied is biology, but in fact, it has been in analysing biological kinds that objectivism has revealed itself as being not just a falsifiable theory but a false theory. Among the fallacies pointed out by one of the main figures of modern evolutionary biology (Mayr, 1984) are:

1) Species do not have a homogenous structure with all members sharing defining properties. It is only possible to give statistical correlations among properties.

2) Since a species is characterised partly in terms of reproductive isolation, it is defined not purely in terms of internal properties of individuals, but in large with their relation to other groups.
3) A species is not defined only by properties of individual members. It is characterised in terms of its gene pool, though no individual has more than a portion of the genes in the pool.
4) If one considers populations distributed over broad areas, there is not always a distinct point at which one can distinguish one species from another.
5) The concept "belongs to the same species as" is not transitive. There are documented cases of populations A, B, C, D and E in contiguous areas, where A interbreeds with B, B with C, C with D, D with E, but A does not interbreed with E. Since "belongs to the same set as" is always a transitive relation, species cannot be sets.
6) Biological species do not always have the necessary conditions for membership. Both interbreeding capacity and morphological similarity are part of the characterisation of a species. But they may not always go together.
7) Status as a separate species may depend on geographical location. Natural kinds, on the other hand, are not defined relative to habitat. Cited by Lakoff (1988, pp. 124-125)

It is evident that all categories studied in HCI (entities, classes or types) are examples of natural kinds. We can suppose that considering classes and entities as classical categories -sets of entities defined by fixed properties and relations holding among them at any instant- are, among other considerations, quite easy to represent and implement in classical programming languages that considering complex kinds of entities with changing characteristics.

Anyway, the example of science –evolutionary biology- refutes the philosophical assumption that kinds are classical sets with attributes. An HCI would have to initiate a process of reconsidering the classical kinds of entities and classes.

2.3 Experientialism

There have been many authors –mainly in the last 15 years- who have criticised the objectivist theory of mind. A group of them, (Lakoff, Johnson, Langacker, Fauconnier, Turner, and others) who have proposed the new approach of cognitive semantics or Experientialism, focus their theory on a different approach of mind. This new approach gives a central role to the body in characterising meaningful concepts and points out the human capacity for creating meaningful concepts and modes of rationality.

The background of cognitive semantics embraces other authors like Putnam (1975, 1981) and Dreyfus (1972, 1988) and even extends to philosophers like Heidegger, Merleau-Ponty and Wittgenstein. These last thinkers came to the conclusion that human understanding was a skill akin to knowing how to find one's way about in the world, rather than knowing a lot of facts and rules for relating them. For these thinkers, philosophy was finished all right, since its attempt to treat intelligence as rational or at least analytic had never worked (Dreyfus, 1988).

2.3.1 Image Schemas

Experientialism states that the research in many domains indicates that mental processes involved when using language and establishing inferences "make use of *image-schemas*, which are *nonfinitary meaningful symbols* of the sort excluded by the strict mathematical characterisation of algorithmic manipulation". (Lakoff, 1988, p. 120).

Johnson explains that his concept of image-schemas is derived from a term first elaborated by Kant, meaning nonpropositional structures of imagination, focusing "on embodied patterns of meaningfully organised experience, such as structures of bodily movements and perceptual interactions" (Johnson, 1987, p. 19). This idea of *embodied* patterns is essential to experientialism as it considers that meaning is based mainly on bodily experiences.

At the same time it is important to point out how different this view is from a mainstream view; the most popular version of this general view regards schemata as abstract conceptual and propositional event structures. Under this classical interpretation, when buying a car, for example, there are typical participants (salesperson, buyers), props (new cars) a sequence of normal events (buyers going to car lot, salesperson showing various cars) and so on; all these elements making up a schema, as a recurring organization of conceptual and propositional knowledge and values that we share about typical situations and events (Op. cit. p. 20).

Such as defined by Johnson, a "*schema is a recurrent pattern, shape, and regularity in, or of, these ongoing ordering activities*. These patterns emerge as meaningful structures for us chiefly at the level of our bodily movements through space, our manipulation of objects, and our perceptual interactions" (Op. cit. p. 29. italics in original). In other words, experience is structured in a significant way prior to, and independently of, any concepts. That's why the term 'nonpropositional' or 'nonfinitary' is employed in relation with schemas.

A simple and general example of image-schema is the FROM-TO or SOURCE-PATH-GOAL schema (Figure 2)

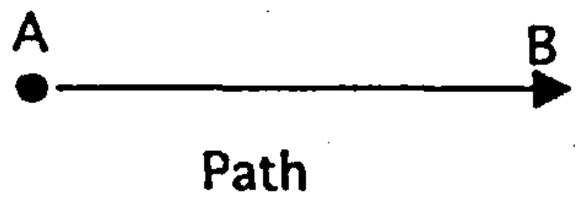


Figure 2 Source-Path-Goal Image Schema

The schema consists of three elements: a source point A, an end point B and a vector tracing a path between them. There is also a relation specified as a force vector moving from A to B. This PATH schema is a recurring structure shown in a number of apparently different events such as:

- throwing a baseball to your friend
- punching your brother
- giving your mother a gift

There are even some examples which have to be considered metaphorically, such as 'the melting of ice into water', with points A and B representing state (e.g. solid or liquid) of a substance (water). In fact, this is a particularly interesting example, as it is different from our three other examples that can be considered as *physical movements* schema. In the 'melting' example, even if there is physical change, it has to be projected metaphorically from the bodily movements world to a conceptual one. The metaphor used in this case would be *The melting is*

a path (from solid to liquid), so the example is a combination of image-schema and a metaphor or, just as Johnson puts it, a metaphorical projection from an image-schema.

This melting example is the prototype of many other metaphorical projections, as we will show in Chapter 3.

Another important image-schema should be considered. Let us see the Container schema, based on that we constantly experience with our bodies both as containers and as things in containers (e.g., rooms and caves).

The structure of the Container schema is defined by three elements: Interior, Boundary and Exterior. This basic structure is arranged so as to yield a basic 'logic'. Everything is either inside a container or out of it (P or not P). If container A is in container B and X is in A, then X is in B, and so on, as shown in Figure 3 (Lakoff, 1988). A metaphorical projection from this schema is, for example, The Visual Field is a Container, generating such expressions like 'things *come in* and *go out* of sight. Memory is also conceived as a container, and ideas *come to* our minds.

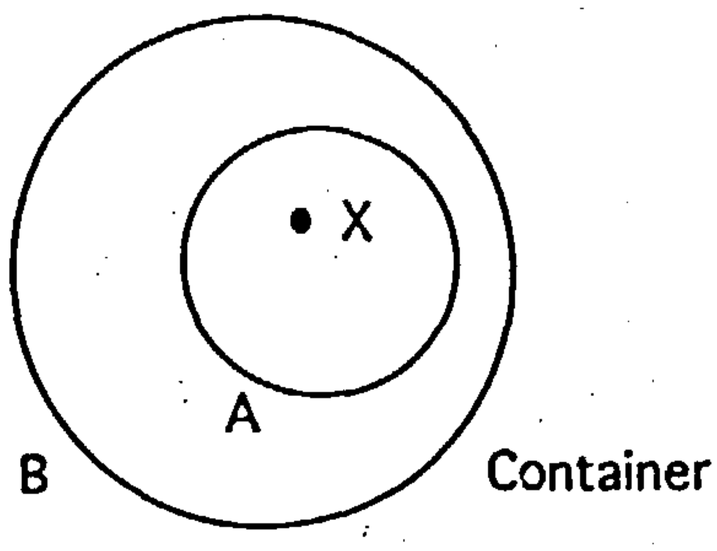


Figure 3 Container Image Schema

A third important schema is Link. In relation with our bodily experience, our first link is the umbilical cord. The basic structure corresponds to a couple of entities A and B, and Link connecting them. The basic logic inherent to the schema establish that if A is linked to B, then A is constrained by, and dependent on, B. There is also a symmetry, as if A is linked to B, then B is linked to A (Figure 4).

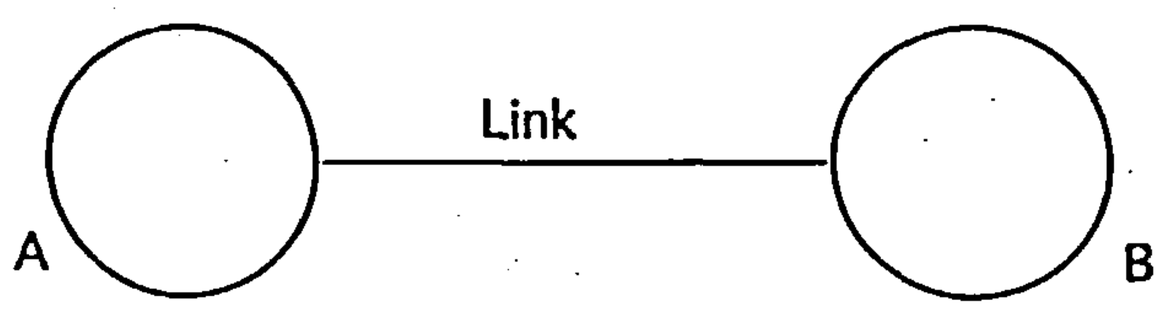


Figure 4 Link Image Schema

Johnson (1987) and Lakoff (1987, 1988) have shown that, in addition to image-schemas, we have to consider a set of imaginative processes for forming abstract cognitive models from image-schemas such as metaphor, schematisation, metonymy and categorisation. In particular, they have shown that there are metaphors mapping image-schemas into abstract domains, preserving their basic logic and that metaphors are not arbitrary, but are themselves motivated by structures inhering in everyday bodily experience (Lakoff, 1988, p. 144).

Some interesting examples of linguistic image-schemas are presented by Turner (1996, p. 16):

Simple image schemas can combine to form complex image schemas. For example, the *goal* of the *path* can be the *interior* of a *container*. This combination produces the complex image schema *into*. Alternatively, the *source* of the *path* can be the *interior* of a *container*, producing the complex image schema *out of*. The *path* can intersect a *container*, producing the complex image schema *through*.

Image schemas have also a main role in producing categories. To recognise several events as structured by the same image schema is to recognise a category.

The term experientialism is intended to include a wide spectre of experiences: basic sensory-motor, emotional, social, cultural and other available to all human beings. The authors remind us that this term should not to be taken in the empiricist sense "as mere sense impressions that give form to the passive tabula rasa of the empiricists". For them, experience is "active functioning as part of the natural and social environment" (Lakoff, 1988, p. 120).

So, the meaningful conceptual structures derive from two sources:

"(1) from the structured nature of bodily and social experience and

(2) from our innate capacity to imaginatively project from certain well-structured aspects of bodily and interactional experience to abstract conceptual structure" (op. cit. p. 121).

There are some basic differences between objectivism and experientialism that it is worth showing. Objectivist cognition considers a disembodied human thought, derived from the fact that the body and the mind are two different aspects of the individual. On the other hand, experientialism cognition sees human thought as involving the structured experience that comes from having a human body, in particular from innate sensory-motor capacities.

The other difference is that while objectivist cognition considers meaning from the point of view of a "correspondence theory" (the association of symbols with external objects), experientialism sees meaning as involving an imaginative projection using the mechanisms of metaphor, categorisation, schematisation, etc. These mechanisms are used to move from what we experience in a structured way with our bodies to more abstract structures (cognitive models).

2.3.2 Metaphors

Metaphor is a cross-domain mapping - conceptualising one domain in terms of another - and is central to our thought processes (Lakoff, 1993). As an example of metaphor, we may consider a love relationship as a journey.

Our relationship has hit a dead-end street.

In this example we have defined two different domains; the source domain (travels) and the target domain (love). We can refer to this metaphor in terms of names which gives the following structure: TARGET-DOMAIN IS SOURCE-DOMAIN. In this case we have LOVE IS A JOURNEY. The *name* is used for the *mapping*; the set of correspondences that characterise it. For the LOVE IS A JOURNEY metaphor we have:

The lovers correspond to the travellers
The love relationship corresponds to the vehicle
The lovers' common goals correspond to their common destinations on the journey.

When dealing with metaphor it is important to realise that, whilst the names of the mappings take a propositional form, the mappings themselves are not propositions. Metaphors are sets of conceptual correspondences, they are cognitive processes, resulting in a set of propositions; but we have to differentiate between the metaphor and the resulting propositions, as in the following example.

Another example of metaphor:

This theory has very good foundations

This sentence corresponds to the metaphor THEORY IS A BUILDING and shows also how, from a concrete domain (construction), we establish a projection to a more abstract domain (that of theories). At the same time we can see how the meaning of the sentence derives from original meanings associated with Building and Construction. The metaphor re-elaborates the original meaning producing a new one that applies to Theories.

2.3.2.1 The conduit metaphor

A well-known metaphor is the conduit metaphor, first analysed by M. Reddy (1993, first edition 1979). This metaphor reflects quite singularly the objectivist philosophy -the mind contains thoughts, language transmit ideas, human communication achieves the physical transfer of thoughts and feelings, etc.- and it is embodied in many expressions which are manifestations of the metaphor. Such expressions could be:

- You have to *put* each *concept into words* very carefully
- Try to *pack* more *thoughts into* fewer words
- Insert those *ideas* elsewhere *in* the *paragraph*
- Don't *force* your meanings *into* the wrong *words*

The sentences above show meanings, thoughts or ideas going into some container (words and paragraphs), but we also find the opposite flow of objects (thoughts) from the container to somewhere else:

- Can you actually *extract* coherent *ideas from* the *prose*?
- Let me know if you *find* any good *ideas in* the essay
- I don't *get* any *feelings* of anger *out of* his *words*

Reddy explains that,

> "in order to investigate the effect of the conduit metaphor on the thought processes of speakers of English, we need some alternate way of conceiving of human communication. We require another story to tell, another model, so that the deeper implications of the conduit metaphor can be drawn out by means of contrast" (Reddy, 1993, p. 171)

Reddy constructs an alternate paradigm (which may be considered as a non-objectivist one), which he calls –and has to be interpreted in the sense that man is the measure of all things - *subjectivist*, and uses it to give another interpretation of sentences like:

> You'll *find* better *ideas* than that in the *library*

From this new point of view, there are of course no ideas in the words, and therefore none in any books, nor on any tapes, records or libraries. He continues arguing that all that is stored in any of these places are "odd little patterns of marks or bumps or magnetised particles capable of creating odd patterns of noise. Now, if a human being comes along who is capable of using these marks or sounds as instructions, then this human being may assemble within his head some patterns of thought or feeling or perception which resemble those of intelligent humans no longer living. But this is a difficult task, for these ones no longer living saw a different world from ours, and used slightly different language instructions". (Op. cit. p.187).

Reddy's arguments, in 1979, are similar to those currently used by Experientialists, recognising that texts are instructions to create *mental spaces* -as we will see in Section 2.4- which, as any active complex process, will re-create or re-enact meaning. He indeed observes that, being the new context different to that in which text has been produced -the expression he employs is a 'different world'-, the new meaning is likely to be different from the original one, supposing that both things are comparable.

What Reddy intended in his paper was to show how our thoughts or ideas about 'thoughts and ideas' are biased by the underlying conduit metaphor on which all sentences used to speak of communication are based. The danger, he points out, is that

> "humanists, those traditionally charged with reconstructing culture and teaching others to reconstruct it, are not necessary in the scheme of the conduit metaphor. All the ideas are 'there in the library', and anyone can go in and get them". (Op. cit. p. 188)

We will argue, in the following chapters, that the danger is not in the metaphor itself, but in the discourse mounted on this metaphor, its ideological aspects which are not only linguistics but social, economics, cultural, etc. We could say that even if language is based on a metaphor that constraint meaning the critical work performed on the language itself can transform such a meaning.

So, the paradigm shift can be produced by changing the underlying metaphor of a given domain, but before finding such a renewed metaphor the work performed on language produces the same effects as Reddy's paper shows. Perhaps the author is aware of this fact when he says that: "If one should look, I daresay even the present article is not free from conduit metaphor expressions". (Op. cit., p. 177) Thus, the inferred idea is that the role played by the article in changing our ideas about communication will not decrease its importance by the fact that it may contain expressions derived from the conduit metaphor itself.

The concerns manifested by Reddy are maybe a consequence of giving some expressions a determining role, thinking that expressions by themselves may change important aspects of politics, cultural policy or economics. Thinking of expressions as instructions to create mental spaces constrained by culture in general -and new discourses in particular- allows us to reduce the impact of a given metaphor. A totally different case is that of discourses, such as some political discourses may show us.

2.3.3 Categorisation

Experientialism has shown that, apart from those categories that can be defined as classical, there are others which cannot be accommodated by an objectivist point of view. Lakoff summarises a classical category as one of the points of the objectivist doctrine:

> "Conceptual categories are designated by sets characterized by necessary and sufficient conditions on the properties of their members. According to objectivist metaphysics, the only kinds of categories that exist in the world are sets." (Lakoff, 1988, p. 130)

Classical categories are those which we usually interact with when defining entities or classes. As they are defined as sets, the translation to computing constructs is direct. But the interesting fact is the collection of non-classical categories (see Section 3.5) that may be useful for our interests.

The first type of non objectivist categories in the strict sense are fuzzy sets. But, as Lakoff has stated, objectivist cognition may be extended to admit fuzzy sets, as "one might well maintain that fuzzy conceptual categories could be represented by symbols that got their meaning by being associated with fuzzy real-world categories." (Op. cit. p.130)

Some highly useful categories cannot be accounted for by association with something in objective reality: is a *week*-schema an internal representation of external reality? Lakoff distinguishes two kinds of cases, depending on whether the nature of categories is an aspect of the mind or is determined by the nature of the human body:

- Cases where the nature of the human body (perception or motor capacities) determines some aspect of the category.

- Cases where some imaginative aspect of the mind -schematic organisation, metaphor, metonymy, or mental imagery- plays a role in the nature of the category.

2.3.3.1 Basic-level categories

In the first group, where perception or motor capacities has a chief role, we find Basic-Level Categories which are thus basic in four respects:

From a perceptual point of view, basic-level categories map to an overall shape, to single images with a fast identification. From a functional point of view, they are associated to a general motor program or general cultural functions. In order to organise our knowledge, most attributes of categories members are stored at the basic-level and the advantage for communication is that they are the shortest and most commonly used words, which we first learn when we are children and they are the first to be included in the lexicon.

In terms of mental images, we can form a general mental image of some objects or biological beings, like chairs or dogs. But we cannot form a mental image for superordinate categories like furniture or animals. We have mental images for elements of the furniture category: chairs, tables, armchairs, beds, etc. but not for a generic piece of furniture.

Similarly, we have general motor programs for using chairs and tables, but not for using furniture in general.

Basic-level categories are formed by entities we physically interact with, thus determining basic elements of perception and function. The communication aspect and knowledge organisation ground on both perception and function. In fact, "basic-level categorization is defined not merely by what the world is like, but equally by how we interact with the world given our bodies, our cognitive organization, and our culturally-defined purposes." (Op. cit. p.134)

A consequence of these considerations is that most of our knowledge is organised at the basic level.

2.3.3.2 Categories defined by Schemas

The first issue to clearly point out here is that 'schema' and 'image schema' are quite different concepts. Schemas (or frames) are cultural creations that give a framework to other derived concepts. What experientialism tries to show is that a lot of words (like *Monday*), are not to be associated to things of the real world. There is no such a thing as a *week* in the real world, but it is a cultural creation that is a function of the culture where it was created. The example of Bali, where weeks of various lengths exists simultaneously, gives us an idea of the relativism of such a concept.

Thus, in order for Monday to have a meaning, we need to have defined previously a general structure such as Week, which have been called 'schemas' or 'frames'. So, for concepts such as 'seller' or 'buyer' to have a meaning, we need a previous frame, in this case the 'market' schema, defined in a larger social and economic organisation. Strange as it may seem, a seller is not 'an external reality' but a cultural creation, an invention of our imaginative structures: words are defined only relative to such schemas or frames (Lakoff, 1988).

2.3.3.3 Categories defined by metaphors (or metonymies)

Another typical way of defining categories by means of an imaginative aspect of the mind is by using metaphors.

The typical expression TIME IS MONEY defines a conceptual structure for time. As a consequence of this way of categorising, it is possible to say somebody is stealing money to mean that the person uses her job's time for a personal concern.

Among the PATH image-schema examples, 'the melting of ice into water' is an example of categorisation based on an image-schema on which a metaphorical projection is applied.

Other categories are sometimes the result of using a metonymy -a well understood or easy-to-perceive aspect of something and use it to stand either for the thing as a whole or for some

other aspect or part of it- in the projection, such as the case of the housewife mother stereotype. This subcategory is used to stand for the entire category of mothers in defining social expectations (Lakoff, 1987, p. 84). Most of social stereotypes are a special way of categorising selecting a subcategory and projecting on the entire category the attributes of the source subcategory; these are metonymic models of categorisation.

Most of metonymic models are, in fact, not models of categories but models of individuals, such as when using the White House expression to indicate the USA government (or Pentagon to indicate the CIA organisation).

2.3.3.4 Summary

Lakoff argues that we organise our knowledge by means of structures called *idealised cognitive models*, or ICMs, and that category structures and other cognitive effects are by-products of that organisation. Each ICM is a complex structured whole, a gestalt, which uses four kinds of structuring principles: (Lakoff, 1987, p. 68)

"*Propositional models* specify elements, their properties, and the relations holding among them. Much of our knowledge structure is in the form of propositional models...

Image-schematic models specify schematic images, such as trajectories... or containers. Our knowledge about baseball pitches includes a trajectory schema...

Metaphoric models are mappings from a propositional or image-schematic model in one domain to a corresponding structure in another domain...

Metonymic models are models of one or more of the above types, together with a function from one element of the model to another. Thus, in a model that represents part-whole structure, there may be a function from a part to the whole that enables the part to stand for the whole." (Op. cit. p. 113-114).

2.4 Mental Spaces

We have seen, when dealing with metaphor definition, that a metaphor is a cross-domain mapping - conceptualising one domain in terms of another - and that mechanism is central to our thought processes (Lakoff, 1993). In this definition we employ the concept of *domain* or *conceptual domain* in order to show how we get a mapping between two conceptual domains, the source and target domains. Lakoff speaks of domains of experience as a wide and somewhat fuzzy concept, one domain of experience being for example love or journeys. "A conceptual domain refers to a vast organization of knowledge such as our knowledge of...dreaming or education" (Turner and Fauconnier, 1995)

There is another way of conceptualising both terms of the metaphor, and is by means of the concept of *mental space*. As mental spaces are fine grained concepts, aimed at differentiate projections between them and resulting in a more clear description of cognitive processes, we are to give a general description of this new concept.

In the line of cognitive semantics, Gilles Fauconnier proposes to study the construction of domains and the principles whereby are mapped (linked), given a structure, incremented altered or merged. The concept of mental space refers to partial cognitive structures that

emerge when we think and talk "allowing a fine-grained partitioning of our discourse and knowledge structures" (Fauconnier, 1997, p. 11)

> "Simplifying somewhat, the overall scheme might be summed up as follows: language does not link up directly with a real or metaphysical world, in between takes place an extensive process of mental construction, which does not mirror either the expressions of language responsible for setting it up, or the real world target situations to which it may be intended to apply. Following current fashion, this intermediate level may be called cognitive, it is distinct from objective content, and distinct from linguistic structure. The construction takes place when language is used, and is determined jointly by the linguistic forms which make up a discourse, and by a wide array of extralinguistic cues, which include background information, accessible schemata, pragmatic manifestations, expectations, etc." (Fauconnier, 1988, p. 62)

What is intended by this approach is that language expressions *do not in themselves* have meaning in the classical sense. They are 'instructions' to build certain kinds of mental construction at the intermediate cognitive level. The constructions involved in this process of meaning unfolding are those of *interconnected domains, and are called 'mental spaces'*. Spaces are set up (at this cognitive level) by grammatical expressions -adverbials, preposition phrases, subject-verb sentences, conjunctions and others that are referred to as 'space-builders'. Mental spaces may be linked to one another by 'connectors'. A connector establishes counterpart relations: it establishes a mapping between an element of one space onto one of more elements of another.

This concept of mental spaces and connectors apply to general situations and not only to metaphors. In the case of metaphor, it would be the case of a source mental space, a target mental space and connectors that map elements from both spaces.

In the general case, "the basic idea is that, as we think and talk, mental spaces are set up, structured, and linked under pressure from grammar, context, and culture. The effect is to create a network of spaces through which we move as discourse unfolds. Because each space stems from another space (its 'parent'), and because a parent can have many offspring, the space network will be a two-dimensional lattice" (Sweetser and Fauconnier, 1996, p. 11).

A frequent example of (at least) two spaces unfolding, is that of a movie where a connection is established by a link between actors (real world individuals) and the characters they portray.

In this example, the name of the president's counterpart (the actor Harrison Ford) is used to identify the character. The starting point is an initial (base) space (B) from which another space ('child') is generated (M), and structures the former in various ways. Our starting base space is produced by 'In *Air Force One* ', the space of the movie, initially without any structure as we do not know anything about it (supposing that we have not seen the film). The second part of the sentence -'Harrison Ford flies...'- creates a child space (the space of the real world, in which the actor Harrison exists) with a minimum structure as he, HF, flies in the presidential Boeing. This structure can be expressed as '*HF flies* '. But there is a connector that links Harrison Ford to the character President of the movie, so the child space gives its structure to the initial, base space, resulting in the more clear structure 'the president flies...' which is incorporated then into the initial space.

This is a simple case in which there is no metaphor, but the sentence unfolds in a two-space mental construction. At the same time it shows us in a practical way the access principle:

> "If two elements **a** and **b** are linked by a connector F (b=F(a)), then element **b** can be identified by naming, describing, or pointing to its counterpart **a**" (Fauconnier, 1997, p. 41)

In the film example, the application of the access principle clearly shows that the character is described pointing to its counterpart, the actor, resulting in a different meaning as if to describe the situation directly using the character. The different meaning is the result of the creation of two mental spaces in place of one. In a classical linguistics interpretation it would be only one meaning where the *real* referent is the same in both cases: the movie.

The last example shows how a mental space is structured by another. But this is only one of all possible cases, and there are others in which a mental space is structured by an idealised cognitive model -ICM-. The ICM could be a frame, for example, that of buying and selling. In such a frame, there is a buyer, a seller, merchandise, currency and price and a rich set of inferences pertaining to ownership, commitments, exchange and son on. If a sentence like *Jack buys gold from Jill* occurs in the discourse, and if Jack, Jill, gold identifies elements a, b, c in a mental space, then those elements will be mapped onto the appropriate slots in the 'buying and selling' frame, as illustrated in the following figure:

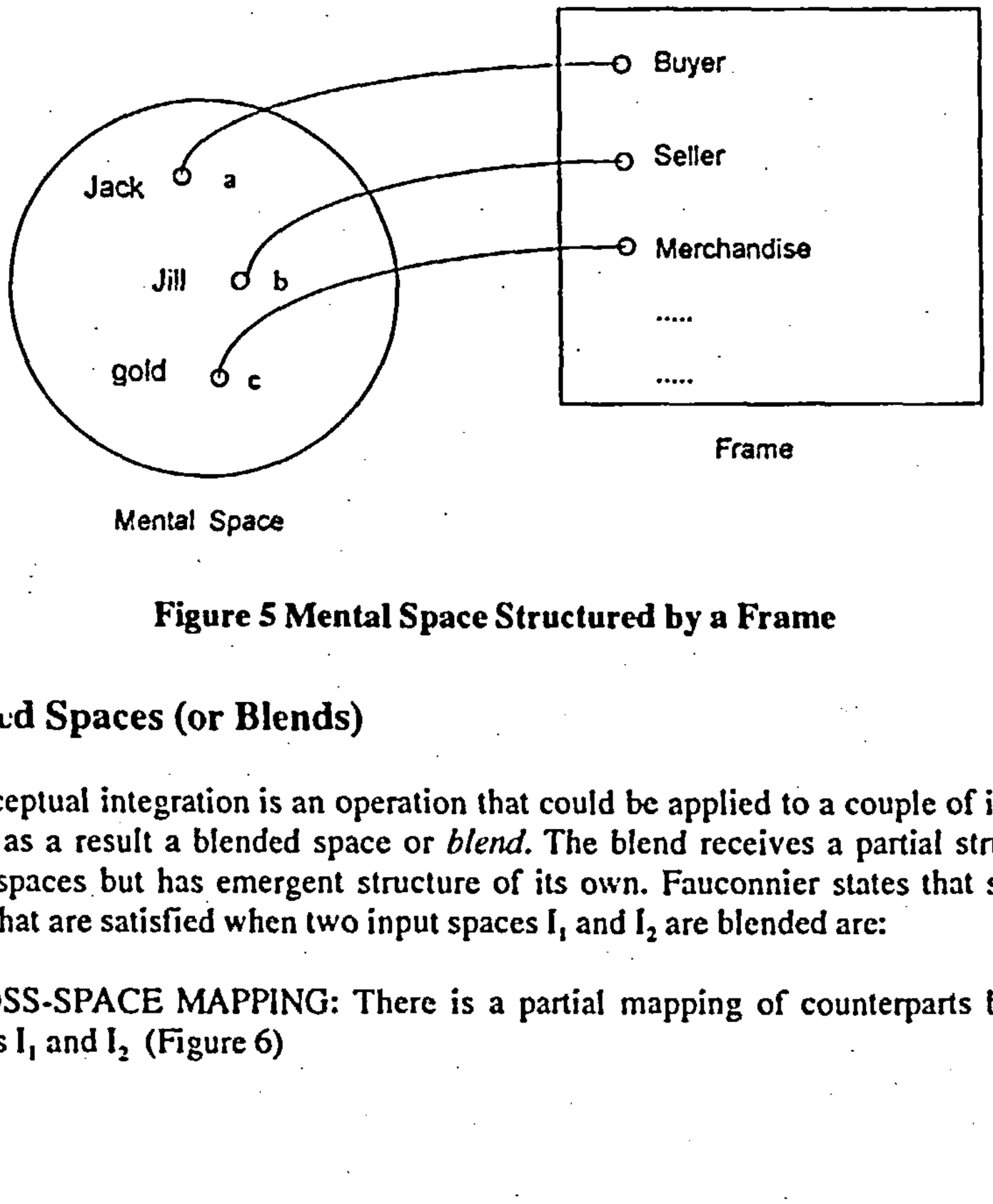


Figure 5 Mental Space Structured by a Frame

2.5 Blended Spaces (or Blends)

The conceptual integration is an operation that could be applied to a couple of input spaces which give as a result a blended space or *blend*. The blend receives a partial structure from both input spaces but has emergent structure of its own. Fauconnier states that some of the conditions that are satisfied when two input spaces I_1 and I_2 are blended are:

(1) CROSS-SPACE MAPPING: There is a partial mapping of counterparts between the input spaces I_1 and I_2 (Figure 6)

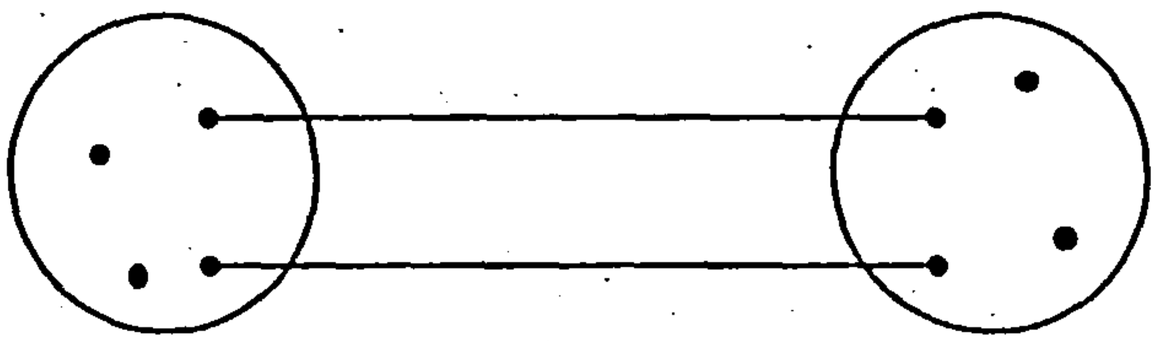


Figure 6 Cross Space Mapping

(2) GENERIC SPACE: There is a generic space, which maps onto each of the inputs. This generic space reflects some common usually more abstract structure and organization shared by the inputs and defines the core cross-space mapping between them. (Figure 7)

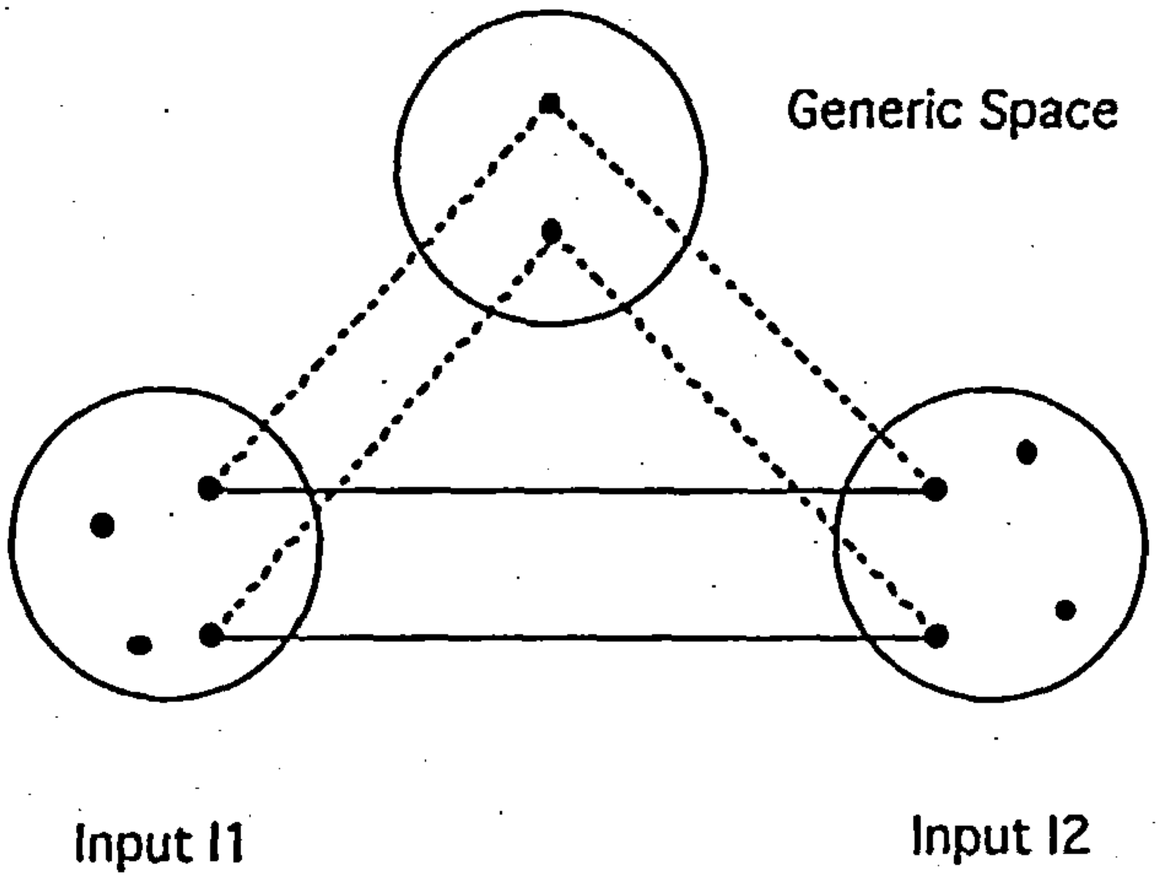


Figure 7 Generic Space

(3) BLEND: The inputs I_1 and I_2 are partially projected onto a fourth space, the blend. (Figure 8)

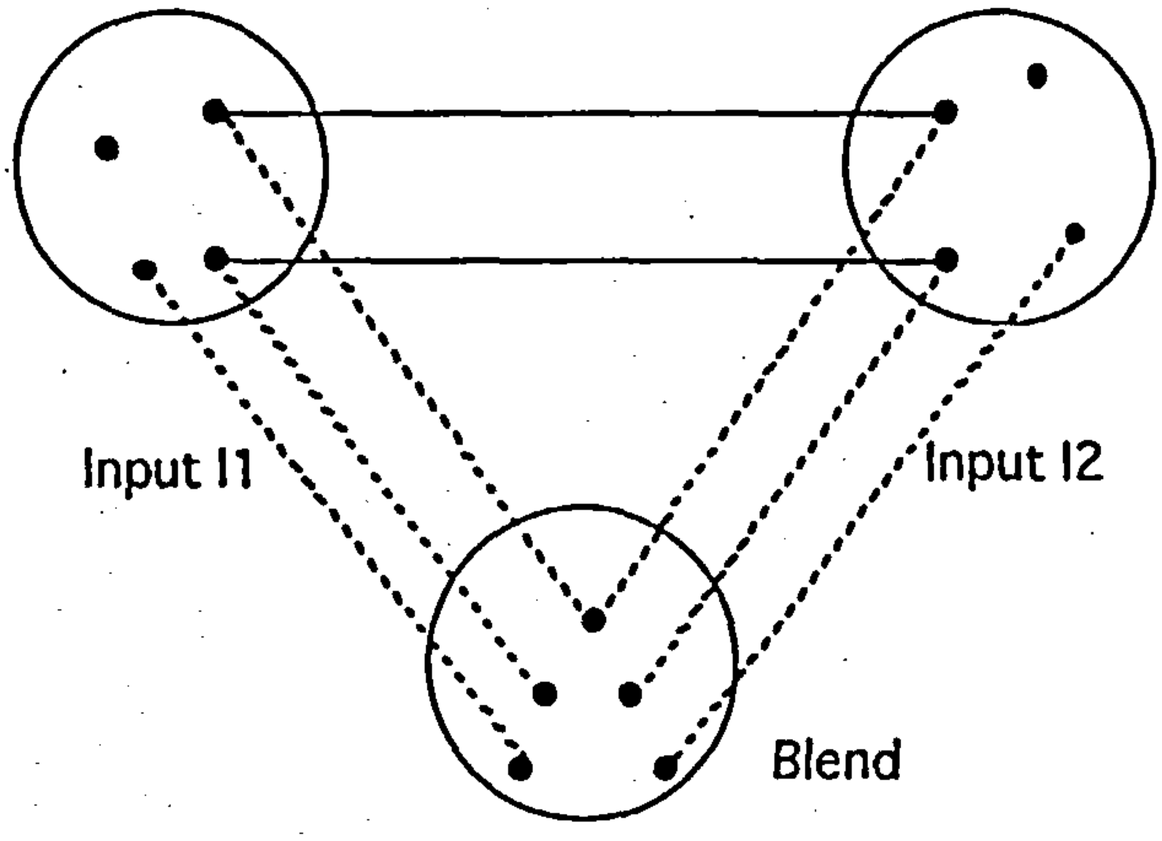


Figure 8 Blend (or Blended Space)

(4) EMERGENT STRUCTURE: The blend has a new –emergent- structure not provided by the inputs.

The emergent structure has features ruled by the following principles:

COMPOSITION: Taken together the projections from the inputs make new relations available that did not exist in the separate inputs

COMPLETION: Knowledge of background frames, (cognitive and cultural models), allow the composite structure projected onto the blend from the Inputs to be viewed as part of a larger self-contained structure in the blend. The pattern in the blend triggered by the inherited structures is 'completed' in the larger, emergent structure.

ELABORATION: The structure in the blend can then be elaborated. This is *running the blend*, an elaboration or refinement of the blend. It consists in cognitive work performed within the blend, according to its own emergent logic.

These rules have been stated by Fauconnier (1997) and are a useful starting point to our objective of using conceptual integration in the frame of HCI.

This new point of view gives us a more detailed knowledge of metaphors. When speaking of metaphors as mapping we usually think that there are two input mental spaces from which we project structure onto a new (blend) space and get an emergent structure. But in everyday expressions we find the blend as a ready-made construction:

This theory has been built on firm evidence

Here, the metaphor is produced by putting together a noun -*theory* - and a verb -*build* - that are not usually employed this way. The blend is already given in the sentence, so it is the process of unfolding meaning that produces both input spaces: *This theory* generates the base space from which another space is derived -*has been built* -, that of material constructions and

their own features of robustness, firmness, etc. Both spaces eventually appear as a consequence of a cognitive process based on the emergence of mental spaces.

Another important point is that any metaphor implies a blend, but most blends are not necessarily derived from a metaphor. Let us see an example of conceptual change shown by the creation of imaginary numbers.

"In Fauconnier and Turner (1994), we look in some detail at the case of number theory in mathematics. Imaginary numbers first showed up in the formulas of sixteenth-century mathematicians Jerome Cardan and Raphael Bombelli. They were considered to be only notational expedients, with no conceptual basis ('sophistic', 'imaginary', 'impossible'). And even though John Wallis was able to provide a detailed geometric analogy for such numbers as early as 1685, one had to wait at least another century for a comprehensive mathematical theory of complex numbers that was conceptually acceptable to mathematicians. This emerging theory is a remarkable blend, in which numbers have properties of numbers (multiplication, addition, and so on) and also properties of point in space (or vectors), such as Cartesian coordinates and polar coordinates (angles and magnitudes). Operations in the blend are characterized in terms of the new conceptual structure. Multiplication, for instance, is the addition of angles and the product of magnitudes". (Fauconnier, 1997, p. 167)

2.5.1 The Buddhist monk story

A nice story, quoted by Turner (1996), shows that the developed structure in a blended space can change our view of the input spaces:

"A Buddhist monk begins at dawn to walk up a mountain. He stops and starts and varies his pace as he pleases, and reaches the mountaintop at sunset. There he meditates overnight. At dawn, he begins to walk back down, again moving as he pleases. He reaches the foot of the mountain at sunset. Prove that there is a place on the path that he occupies at the same hour of the day on the two separate journeys" (p. 72)

The proposed solution to the problem is stated as an ingenious one by imagining the Buddhist monk walking up as his double walks *down on the same day*.

"In that blended space, it is clear that there is a place on the path that the two Buddhist monks occupy at the same hour of the day. The place is where he meets himself. He must of course meet himself, since someone who traverses a path in the space of a day must meet someone who traverses the path in the opposite direction in the space of the same day, however much they both walk as they please. This inference, that there must be a place that the two travelers inhabit at the same time of the day, is projected from the blend back to the input spaces to create a point of connection between the input spaces of the two journeys, although no encounter occurs in either of them" (p. 72)

The input spaces are the outward and return journeys. We imagine the Buddhist monk walking up and his double walking down in the blended space. In the blend we find the solution to the problem and an encounter place where monks meet each other. The interesting issue is the back projection to both input spaces, so the solution simultaneously provides a new structure for both input spaces, transforming them as a result of the cognitive process of solving a problem.

"Blending is a dynamic activity. It connects input spaces; it projects partial structure from input spaces to the blend, creating and imaginative blended space that, however odd or even impossible, is nonetheless connected to its inputs and can illuminate those inputs". (Turner, 1996, p. 83)

The second important characteristic of blended spaces is that they enable inferential work to be performed. A blend produces knowledge. The story of the monk has been solved in a blended space in which some inferences were performed to show how both monks meet each other at a certain time of the day.

As we have seen, apart from the blended space there is also a generic space that corresponds to the abstract structure where both input spaces share and connect them. In the case of the Buddhist Monk, its generic space has a single traveller taking a single journey from dawn to sunset, over a single distance along a single path.

"The generic space does not specify the direction of the journey (up or down), the date of the journey, or the internal form of the journey (starting and stopping, moving slower or faster). This degree of inspecificity allows the generic structure to be projected equally well onto the space of the ascent on the first day, the space of the descent on the second day, and the blended space where both journeys occur on an unspecified day" (Turner, 1996, p. 87).

A question that could be posed regards the actual conceptual existence of the generic space. Is it not only a name for structure shared by the input spaces? In the example above we see that the generic space with the abstract structure can be projected in terms of only one of the input spaces, the ascent journey, for instance. Thus, the generic space is produced as a projection of some abstract structure from any single space and applies to a couple of them when they share such an abstract structure.

2.5.2 Maps as blended conceptual spaces

Maps provide another interesting example of blends. A map is a visual representation that brings together information from many different mental spaces associated with what the map represents.

"These spaces include street topography, names of places, latitude and longitude, appearance, water versus land, distances, number of hours it takes to travel city to city, ...and so on." (op. cit. pp. 97-98).

When representing a territory filled with coloured bands to indicate temperatures, the combination in the map signifies mental space connections and not mental space blending. However, such representations of mental space connections will be associated to many types of conventional blends. For example, there is a conventional mental blend of temperature and colour (hotter is redder) and the idea of changing this conventional blend would imply serious misunderstandings by users of the map.

2.6 Parabolic projection

Our partitioning of the world into objects involves partitioning the world into small spatial stories because our recognition of objects depends on the characteristic stories in which they

appear: We catch a ball, throw a rock, sit in a chair, pet a dog, take a drink from a glass of water. (Turner, 1996)

Thus, the concept of story is intimately associated with objects in which they participate. We categorise objects by means of the stories they are associated with where each story represents an event or action and therefore the set of stories related to an object allows for the establishment of the set of interactional properties.

This is the reason why Turner (1996) proposes to use the word parable much more widely than the common English term: *Parable is the projection of story*. Turner's preference to use this literary term -in place of the more usual projection- is due to the idea that everyday minds have much to do with literature.

Unlike simple projection, which projects from individual elements of one mental space to another, a parable can project whole structure between mental spaces. In particular, a

> "parable often projects image schemas. When the projection carries structure from a 'source' we understand to a 'target' we want to understand, the projection conforms to a constraint: The result of the target shall not be a conflict of image schemas" (Op. cit. p. 17)

The usual case is when we project spatiality onto temporality, in which case we are projecting image schemas. We usually think of time as having a spatial shape, linear or circular. This way of conceiving time and events in time arises by projecting skeletal image schemas from *space* onto *time*.

2.7 Perception as integration of multiple fragments

The following quote from Mark Turner (1996) gives us a more detailed idea about such complex cognitive processes:

> "It appears that there may be no anatomical site in the brain where a perception of a horse or a concept *horse* resides, and, even more interestingly, no point where the parts of the perception or concept are anatomically brought together. The horse looks to us like obviously one thing; yet our visual perception of it is entirely fragmented across the brain. What the brain does is not at all what we might have expected" (op. cit., p. 110) (Italics in original).

The visual perception of different attributes occur in a fragmented fashion throughout the brain and, surprisingly, there is no place where the whole image is re-assembled. This is a paradox, the same way that the visual field is projected upside-down onto the retina because of the simple optics of a lens, as there is no place in the brain where the image is turned right side-up.

This complexity indicates that meaning could not be mental objects bound in conceptual places but rather complex operations of projection, binding, linking, blending, and integration over multiple spaces (op. cit., p. 57).

2.8 Situated Cognition

The criticism to objectivist cognition made by the approach of situated cognition is, in some aspects, very similar to that of experientialism. A central idea of situated cognition is to avoid equating knowledge with representations of knowledge. So it is a question of articulating the nature of knowledge, representations and change in the workplace. The authors that support this approach (Clancey, Suchman, Lave, Maes, Wenger, etc.) consider that knowledge is not a thing but a capacity to interact. The following quote could be considered as a program definition for situated cognition:

> "Human conceptualization has properties relating to learning and flexibility that make human knowledge different from procedures and semantic networks in a computer program. Situated cognition research explores the idea that conceptual knowledge, as a capacity to coordinate and sequence behavior, is inherently formed as part of and within physical performances." (Clancey, 1994)

This is a close idea to that of bodily nature of the mind that states experientialism (some titles are illustrative, e.g. *The Body in the Mind* by Johnson. (1987))

> "Knowledge is more analogous to energy than a substance. Knowledge and representations are not interchangeable: We can represent what someone knows, but the representations are not the knowledge itself (the map is not the territory)". (Clancey, 1994). Conversely, it is possible to act without manipulating linguistic representations at all.

An objectivist idea of knowledge can trigger the following sentence: *They have knowledge but cannot act.* This conception equates knowledge with representations and acting with applying representations, suggesting an apparent paradox: A person can "have knowledge" but not be able to use it.

Another classical objectivist idea (which is the underlying assumption of most knowledge and software engineers) could be that *knowledge should be stored before it gets lost.* This is a way of equating knowledge with representations (what we "capture", "elicit" when interviewing an expert), considering the memory (or the mind) as a place where knowledge is *stored.* The extension of the idea is that learning is like storing representations in a knowledge base. "This also suggests that using representations involves merely accessing them and applying them, leaving out the reconceptualization that always occurs when we read and comprehend text" (Clancey, 1994).

Clancey points out that the new way of conceiving how representations are created in the course of our activities can be difficult to understand given that the orthodox view of knowledge as stored symbols and speaking as a process of translating ideas into words is so deeply rooted in our society.

We are not diminishing the importance of the role representations play in human behaviour, but focusing on the cycles of behaviour where these representations have their important moments. After perceiving representations, the human being reorganises (reorients) his activity. This dynamic and complex process of continuous reperceiving, readjusting and replanning our activities through representations is an integral component of our intelligent behaviour. In terms of Activity Theory, representations are theoretical artifacts we use to re-structure the activity itself as well as other representations.

We see the big difference between both approaches: while knowledge engineering focus on symbolic representations as the 'real' knowledge (and sometimes equating the *original* knowledge -in the head of the expert- and its representation), the situated cognition approach gives representations the important role they play in the entire complex and dynamic activity.

Finally, it is interesting to see how Feigenbaum –the head of the DENDRAL project, a pioneering in expert systems- admits in one surprisingly frank passage how different are experts from expert systems:

> "Part of learning to be an expert is to understand not merely the letter of the rule but its spirit. [The expert] knows when to break the rules, he understands what is relevant to his task and what isn't... Expert systems do not yet understand these things". (Feigenbaum and McCorduck, 1983, pp. 61-62).

Summing it up, what expert systems –or computers in general- *still* can't do (and presumably never will do) is being able of perceiving the situation, the context, which makes it possible the creation of meaning.

Chapter 3

Where modelling notation comes from

3.1 Using image schema and metaphorical projection

We are going to see that most notation elements used in HCI methods are produced by a metaphorical projection from an image-schema. In some cases there is a combination of either image-schemas or projections or both, as in the case of an element being at the same time the origin of a PATH image-schema and a CONTAINER (for instance, in a State Transition Diagram, we have a container –a state- from which starts a path image-schema –a state transition). Entity-Relationship Diagrams, Data Flow Diagrams (DFD) or State Transition Diagrams are examples of models based on containers and link or path image schemas.

3.1.1 Entity-relationship models

Entity-relationship models are based on two metaphorical projections: the entity is a CONTAINER, and the relationship is a LINK.

The entity, as a container, mainly contains attributes with different characteristics (key, alternate key, foreign key). The relationship, as a link, has only the function of relating two different entities without any type of direction. As relationships are usually given a name (or even two), this means that another type of metaphorical projection has to be established: the path. In order for a relationship's name to have a meaning, it is important to know *from where* the relationship originates and where it goes. The name is associated with a given *direction* of the path image-schema upon which it is based.

So we can differentiate two cases, the first one with both, container and link, image-schema as source domain for the metaphorical projection, as shown in Figure 9:

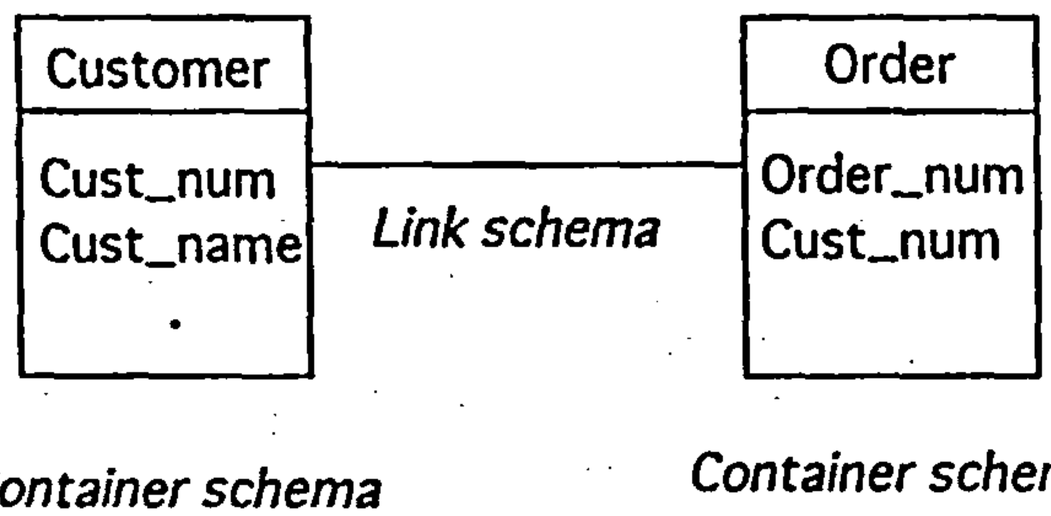


Figure 9 Entities as Containers with Link Schema (Relationship)

The second, with container and path image-schema as source domains. Depending on whether there is only one name or two associated with the path image schema, it could be thought of one or two path image-schemas, each one going from one entity to the other (Figure 10)

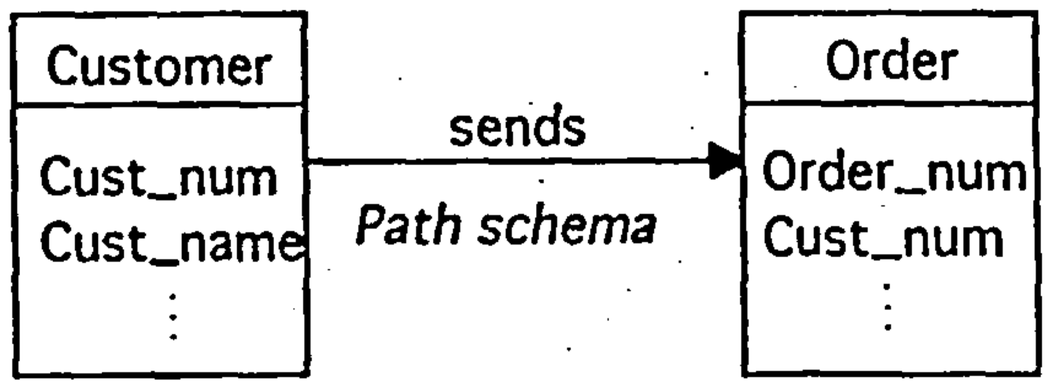


Figure 10 Entities with a Directed Relationship (Path Schema)

When using two names each one corresponds to each of both path image-schema, but it is usual to represent the model with only one segment, not even including an arrow to indicate the direction. This situation is represented in Figure 11:

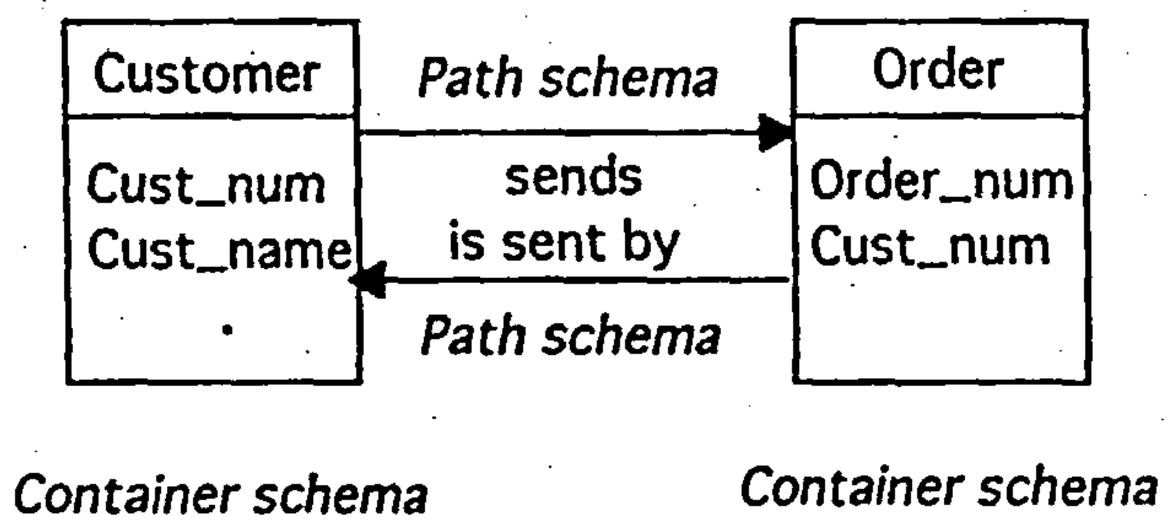


Figure 11 Entities with two Directed Relatiuonships

The final conceptualisation is the result of superposing 4 metaphorical image-schema projections: two from a container source domain and two from a path source domain.

Perhaps it is interesting to differentiate two meanings that could be derived from Figure 11. The first, which we have defined as a Path image-schema, refers to the graphical representation and means the sense of the path in reading the relationship: *customer sends an order*. The second meaning corresponds to paths that orders (or customers) may make, a question not considered in the model.

When defining the container image-schema, we have seen that the structure of such schema implies a direct logical fitting of other containers into the original and in this way structuring a container by others. This is the logic of sets, whereby if container A is put in container B, all elements of A are also elements of B, so X belongs to A and -as A is in B- also belongs to B:

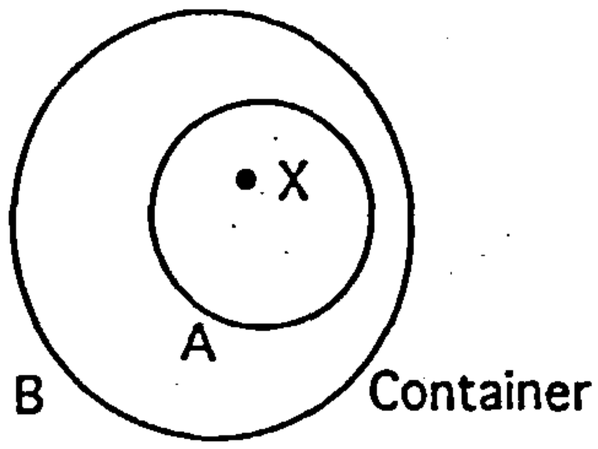


Figure 12 The Logic of Containers

When using an entity A containing another B, the usual meaning is just the opposite to that used in set theory, indicating that all attributes of A are also attributes of B. So, even if the representation sometimes used is that of one container in another, the meaning is subordinate where the contained entity possesses all attributes of the container as shown in figure 13:

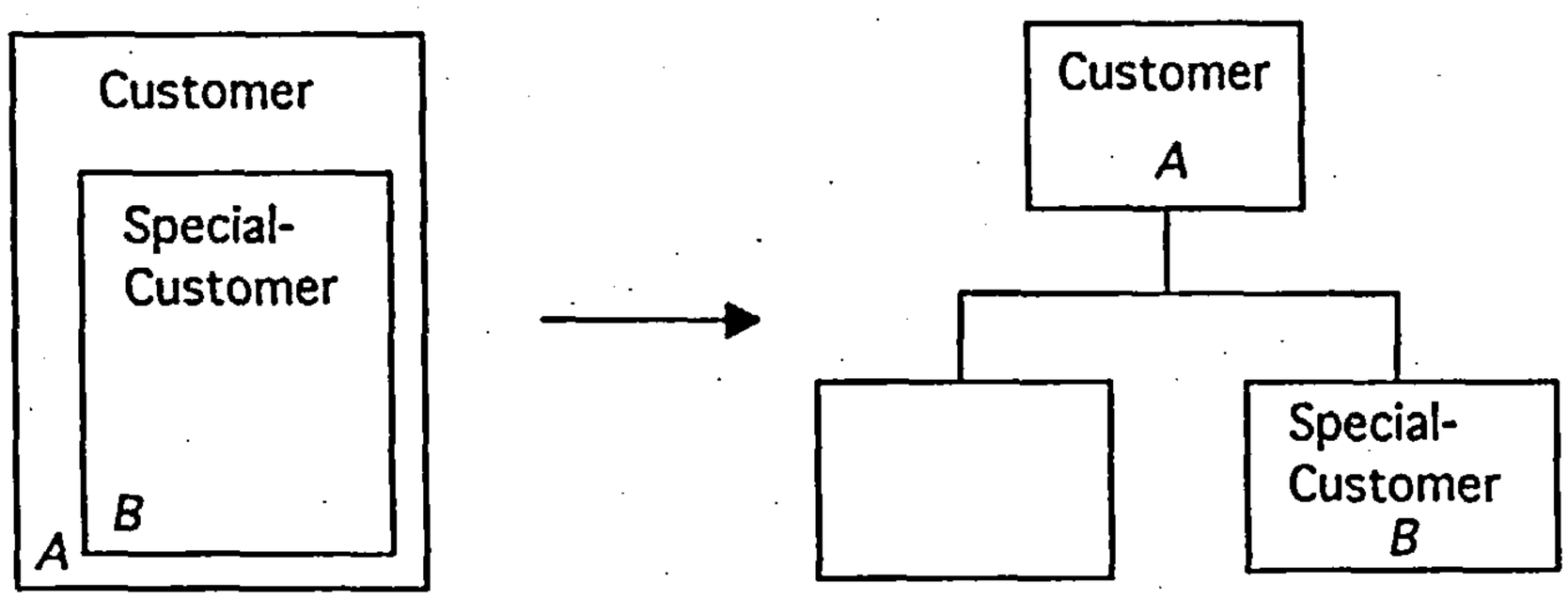


Figure 13 A Contained Entity is a Subordinate Entity (Subclass)

From a cognitive point of view, entity A (superordinate) has been built from a collection of basic-level categories and generalising all common attributes to the superordinate. This category generation is not a problem for, given a particular case, because the generalisation projection could be reversed in a specialisation direction, resulting in a special case not included in the general category prototype. Both examples are typical in Object Oriented methodologies.

3.1.2 State Transition Diagrams (Finite State Machines)

A typical State Transition Diagram (STD) is built by means of a composition of PATH image-schemas. The end or goal point of a path is the origin or source point of yet another path. There are indeed some places that are both an origin and an end point at the same time.

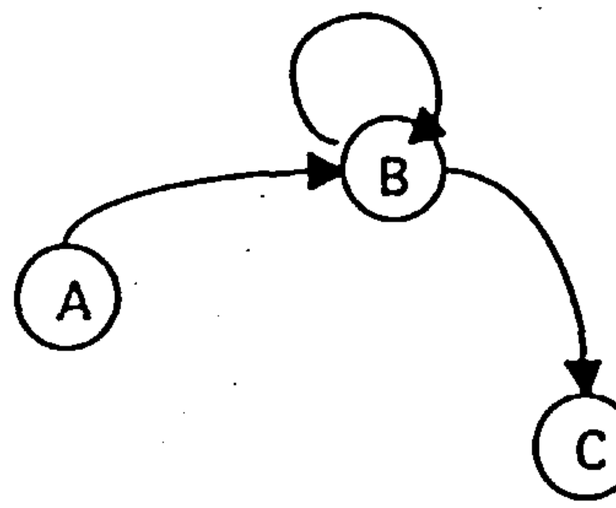


Figure 14 State Transition Diagram as composition of Path Image Schemas

In a STD we need to indicate where the path comes from and where it goes. Each node -representing a state or location- is usually conceptualised as a point. A change in conceptualising the node -as a container in place of a point- produces the transformation of the STD into a State Chart where the structure of each node is expressed as another STD at a lower level:

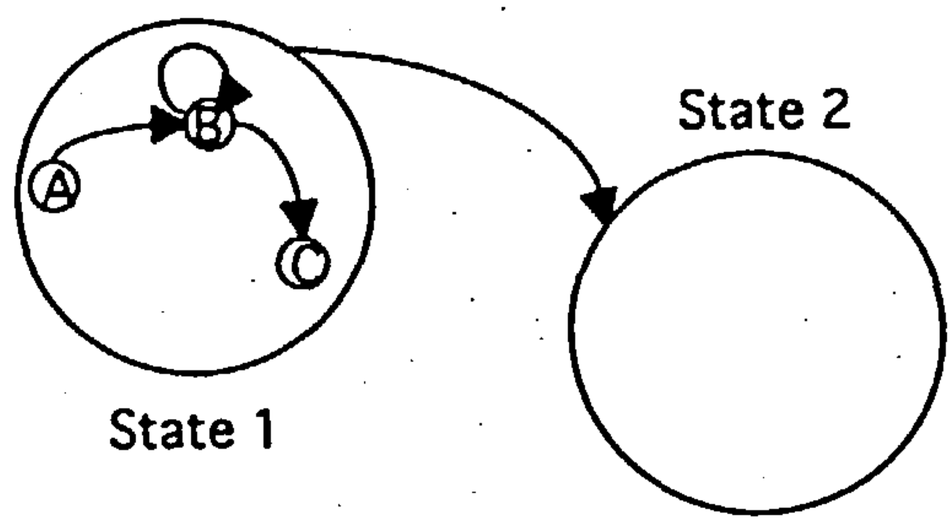


Figure 15 Superstates as Containers

We can think of these levels as being equivalent to mental spaces; we pass through different levels as if we were constructing new mental spaces. This special way of categorising -like Russian dolls- can be called *categorisation by container*. We will see how this type of categorisation can lead to difficulties when there is a long cognitive distance from an initial node to an elementary (or goal) node; the cognitive distance being measured in terms of the number of levels or mental spaces to be covered (as when trying to pass from a Contextual Bubble to the first level bubble in a complex system)

3.1.3 Data Flow Diagrams

At this point, it would be useful to show the difference between metaphor and analogy. The main difference between both concepts has been pointed out by Presmeg (1997) when she says that the word analogy covers two different concepts: similes, which are explicit analogies ('a is like b') and metaphors, which are implicit analogies ('a is b').

'Metaphor is a specific form of analogy' (p. 267) but showing similarities in an implicit way has a strength that is lost in the explicit form. In this sense Sfard (1997) says that

> "the main point to remember is that *metaphor has a constitutive power* and thus functions *a priori*: It brings the target concept into being rather than just sheds a new light on an already existing notion...*the act of creation itself is a matter of metaphor*" (op. cit. pp. 344-345, italics in the original).

Analogy, or simile, on the other hand, makes a comparison between two already constructed concepts, even if this comparison does not leave our understanding of either the target or the source unchanged.

Before explaining our metaphor (THE SYSTEM IS AN INDUSTRIAL PLANT), it is interesting to distinguish -among many others- two different uses of the term metaphor. Some metaphors play a pedagogical role helping to teach or explain theories that already have a sound theoretical formulation (like the planetary metaphor of the atom in Physics). Others are used to organise a cognitive model that gets its whole meaning when the underlying metaphor is recognised.

> "Nevertheless, it seems to me that the cases of scientific metaphor which are most interesting from the point of view of the philosophy of science (and the philosophy of language generally) are those in which metaphorical expressions constitute, at least for a time, an irreplaceable part of the linguistic machinery of a scientific theory: cases in which there are metaphors which scientists use in expressing theoretical claims for which no adequate literal paraphrase is known. Such metaphors are *constitutive* of the theories they express, rather than merely exegetical". (Boyd. 1993. p. 486).

One important issue to be emphasised is that constitutive metaphors, once detected, can be used as pedagogical ones. But it is possible to have for the same theory two different metaphors: one constitutive, the other pedagogical (Imaz, 1995). This is an essential issue in order to argue the usefulness of some constitutive metaphors -such as the desktop metaphor- when applied to training purposes.

In fact, the real question is: how can we detect a constitutive metaphor?

The discovery of that type of metaphor will be based on the linguistic use, the *linguistic machinery* in Boyd's expression (let us observe again the same metaphor, *something is a machine*) of the theory.

Even if we analyse the linguistic use to detect a possible candidate, we must take into account that "the locus of metaphor is not in language at all, but in the way we conceptualise one mental domain in terms of another." (Lakoff.1993. p. 203).

Data Flow Diagrams (DFD) are based on a metaphor. Even if one process is also categorised as a container and its structure is determined by another DFD at a lower level, the main metaphor on which the model is based is THE SYSTEM IS A FACTORY. In such a plant there is a collection of processes interconnected by pipes or assembly lines. The raw material for one process originates from other processes, external sources or stores containing by-products of yet other processes.

The lines connecting two processes (data flows) have a direction but they are thought of as containing data elements that move from the source process to the target process. So, data flows correspond to equivalent pipes or assembly lines containing elements to be processed in the next production step. An alternate point of view is to consider data flows as an instance of the CONDUIT METAPHOR, whereby a message is an object in a container; communication is transferring an object spatially from the speaker to the hearer (Turner, 1996). We prefer the unifying metaphor of the factory (Figure 16) because it is equivalent conceptually and we do not need to employ a combination of two metaphors: two containers joined by a conduit.

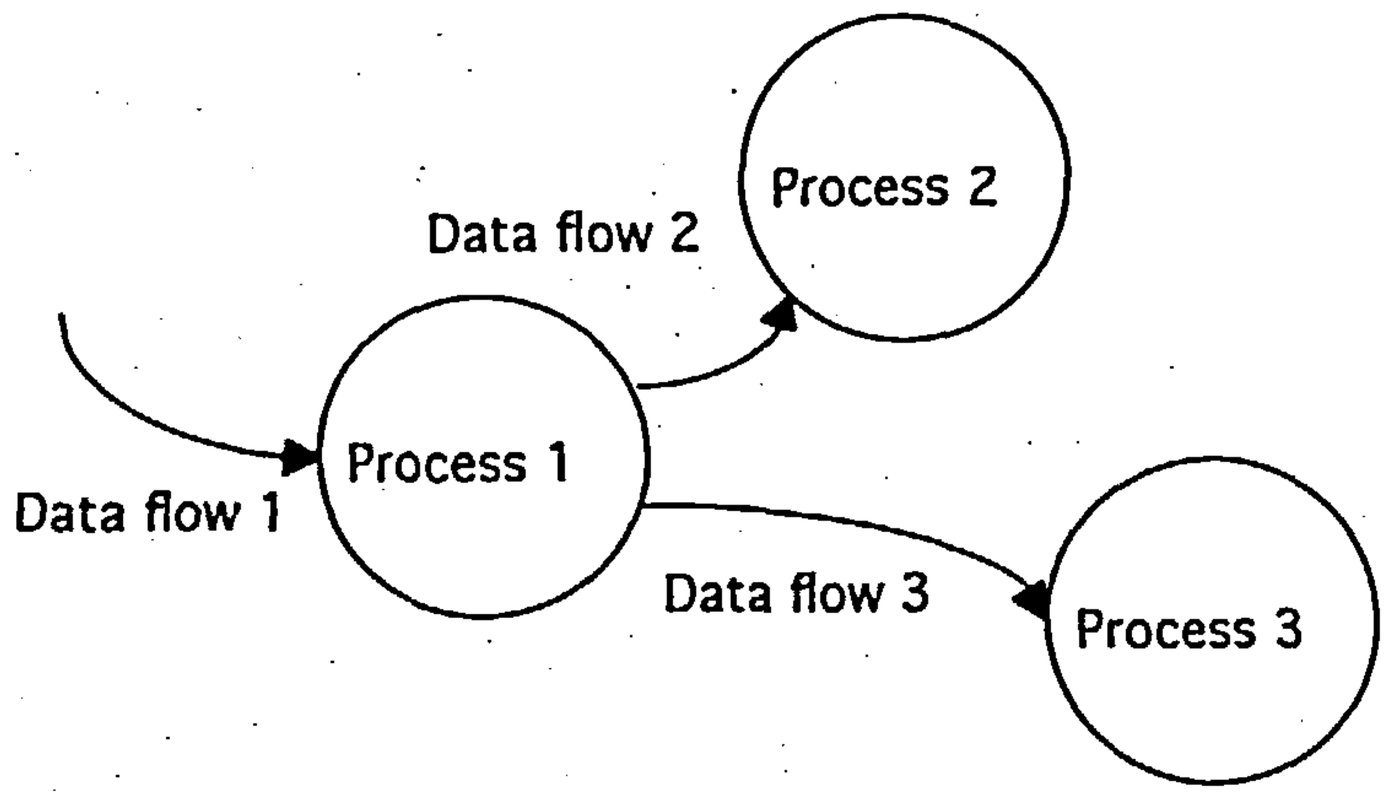


Figure 16 The Factory Metaphor Applied to Data Flow Diagrams

The idea of giving structure to a process by means of a sub-model (another DFD) contained in it is a projection of the concept used in structured programming (stepwise refinement), but subsequently renamed as the Top Down Approach.

3.2 Problems with the Top-Down Approach

The successive structuring of a container in terms of a collection of other containers leads us to ask about the validity of this way of categorisation. The original idea by Dijkstra (1972) is clearly stated by Martin and McClure:

> "The one programming problem of paramount concern in the 1970s was complexity. This is the problem of programming in the large. It became possible and necessary to develop large, highly interrelated systems of programs. When the same heuristic programming practices used to develop small, stand-alone applications were tried in this new environment, they failed miserably. It soon became obvious that the problem of scaling up was indeed very difficult" (Martin and McClure, 1988, p. 103).

As pointed out by the authors, top-down programming is a popularised version of Wirth's stepwise refinement method. It represents an important advancement over earlier, ad hoc programming methods because it is an orderly way of attacking large programming problems. However, it is an informal, non-rigorous method (Op. cit. p. 104).

3.2.1 Long cognitive distance

There are two additional important issues -from the experientialist point of view- to be taken into account when considering top-down methods:

- When in structured programming a relatively complex process was subdivided into a set of sub-processes, there was a short cognitive length from the source domain to the target domain, as each of the sub-processes will have to be implemented in a programming language, which gave the resulting meaning to each of the sub-processes. There was a translation from one domain into another, resulting in a cognitive projection of length one (considering the total number of mental spaces involved in the cognitive process). A proposition such as *Read Input File*, used to give a name to a process or function, had a direct equivalent in terms of a set of programming sentences, which implemented the desired function.

- When trying to scale up the top-down approach for further complex problems, it would be a greater cognitive length when categorising a given space in terms of others which themselves are to be categorised the same way. The main concern is the categorisation method, *categorising by containers*, a very different model from the four models established by experientialism and a large group of cognitive scholars: *propositional models*, *image-schematic models*, *metaphorical models* and *metonymic models* , each of them having a direct projection cognitive length of one (number of mental spaces involved).

The idea of conceptualising by containers is in fact the application of a metaphor: the mind as a container. But we have already seen in Chapter 2 that parable gives a different view of meaning "as arising from connections across more than one mental space. Meaning is not a deposit in a concept-container. It is alive and active, dynamic and distributed, constructed for local purposes of knowing and acting. Meanings are not mental objects bounded in conceptual places but rather complex operations of projection, binding, linking, blending, and integration over multiple spaces." (Turner, 1996).

The problem with successive unfolding is that the meaning is left until the final implementation is written, all intermediate levels not having a well defined categorisation. That implies the ongoing application of top-down and bottom-up approaches in order to find a reasonable solution for the problem, and it goes without saying the necessary required skills to produce it. The occasional apparent seamless fitting of top-down approach is a result of the good performance of well-skilled designers, not of a correct application of the method.

3.2.2 Basic-level events

In Chapter 2 it has been argued that the main categories we use in our everyday activities are basic-level categories. These categories are defined, in fact, at some *intermediate* level, they are formed by entities we physically interact with, thus determining basic elements of perception and function. So, if we want to try an equivalent of basic-level categories as a starting point in designing new systems, we have to look for basic-level events which will be involved with in interacting with the new system in order to define its properties.

"The relevant notion of a 'property' is not something objectively in the world independent of any being; it is rather what we will refer to as an *interactional property* -the result of our

interactions as part of our physical and cultural environments given our bodies and our cognitive apparatus. Such interactional properties form *clusters* in our experience, and prototype and basic-level structure can reflect such clusterings". (Lakoff, 1987, p. 51, italics in original).

This problem has been already set out by Ed Yourdon, who has criticised this 'classical approach' of drawing the context (Level '0') diagram and proceed directly from this single bubble in the context diagram to a high-level (Level '1' or higher) DFD. Yourdon points out some of the problems derived of this 'pure' top-down approach: analysis paralysis, the six-analyst phenomenon or the arbitrary physical partitioning. He proposes to design the high-level DFD based on a "list of main events". (Imaz, 1995)

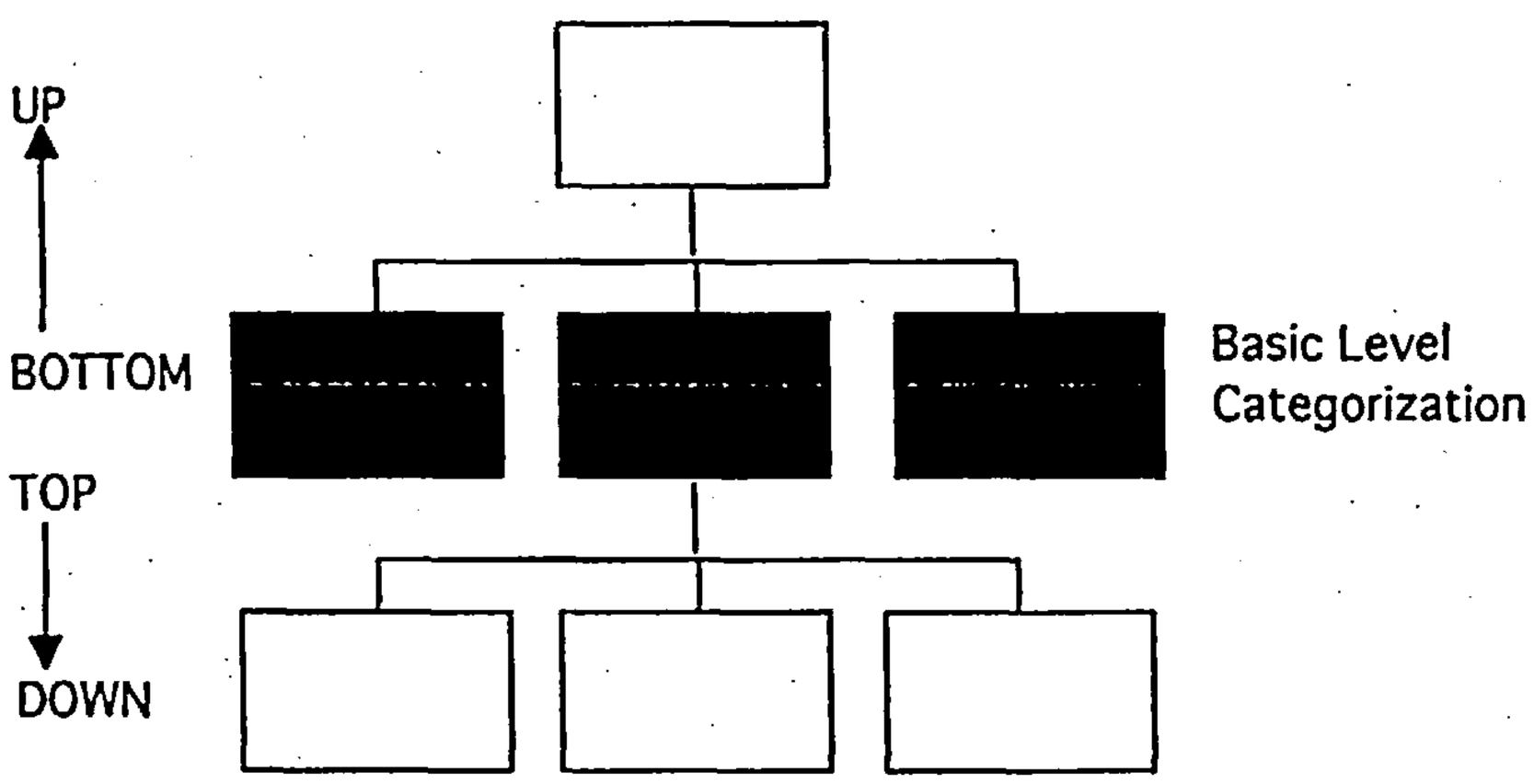


Figure 17 The "Middle Out" Approach

Even if Yourdon has cleared stated the 'top-down' problem –analysis paralysis, the six analyst phenomenon, arbitrary physical partitioning- (Yourdon, 1989, p. 360), he has not established a method for detecting the *main events*. As a consequence of all what has been discussed, the search for main events means a search for basic-level events. This would imply the specification of some type of *scenarios* in which an interactional model could be established.

Another characteristic derived from the top-down approach is the resulting hierarchical structure. This is a way of assimilating the structure of the model to well known structures used to represent biological and historical phenomena.

3.2.3 Scenarios used to define basic-level events

Lakoff's main thesis is that we organise our knowledge by means of structures called idealised cognitive models (ICM) of many types, one of which will be specially important to our aims: the scenario.

> "The scenario ontology consists typically of people, things, properties, relations, and propositions. In addition, there are typically relations of certain kinds holding among the elements of the ontology: causal relations, identity relations, etc. These are represented structurally by link schemas, each of which is categorized as to the kind of relation it represents. Scenarios also have a purpose structure, which specifies the purposes of people in the scenario. Such structures are represented metaphorically via SOURCE-PATH-GOAL schemas". (Lakoff. 1987. p. 286).

The scenario ontology has been studied by some authors (Carroll, 1995; Jacobson, 1992) using different names (scenario, use case) but in agreement with the cognitive basis experientialism had already defined. For instance, Jacobson et al. (1995) define a use case as

> "...a sequence of transactions in a system whose task is to yield a measurable value to an individual actor of the system."

In this definition we have causal relations (sequence of transactions), people (actors that communicate with the system), purpose structure (the measurable value to the actor) and so on, which constitute all the elements that define the scenario ontology.

3.3 The design method vs. the exposition method

There is another important issue regarding the top-down approach. Even if a pure top-down approach offers -when designing- a 'false sense of security' (Martin and McClure, 1988) in place of a coherent and more rigorous method, it could be used as a clear method for the exposition of the design results.

The neat and well structured documentation obtained as a result of applying a top-down method has likely determined that the same approach was used for designing systems. The method of categorising used in design is not necessarily the same as the method of exposing the results in a comprehensible way aimed at training or discussion of results.

So, once the design has been finished and has to be presented to an audience, the top-down approach is an excellent way to explain the characteristics of such an artifact. The top-down is the use of a travel metaphor, whereby we first overfly a territory indicating the main cities and towns. After landing at each of the cities and visiting them –the travel might be applied to visiting the town with another transport such as bus and stopping to see some places in detail- we continue to fly to the next city. The problem with the metaphor is that we need to have already existing cities to visit; the flight, on its own, can only give us some very general guidelines but not precise information in order for us to plan a new city.

A method that cannot be bettered for an exposition is not necessarily a good methodological tool for building new artifacts. What is needed is an approach based on some cognitively representable pattern. We will return later to the top-down approach once we will have presented some additional ideas, like scripts and stories.

3.4 Object-Oriented Methods

In the case of object-oriented methods we are going to analyse only the structure and the dynamics (system behaviour) of these types of models.

3.4.1 Structural domain

The building of the Object Model (OM) (used in most of the methodologies) is a direct application of classical categories; that is, the world consists of entities, the properties of those entities and the relations held among those entities. In order for an object to belong to a category, it is necessary and sufficient that the object share the same properties as other members of the category. This is the usual way of thinking about real life categories, and this thought is reinforced by such methods as entity-relationship diagrams and OO class diagrams.

We are not intending here to say that such diagrams are not useful, we only point out that classical categories are an oversimplification of reality, and that we frequently need to add mechanisms to the models in order to capture features that do not fit adequately in a classical model. As we will see later in Chapter 7, some patterns may be useful in order to represent a more complex picture of real problems.

Many authors who support the object-oriented (OO) methodologies claim that the resulting models are closer to the real world than structured models are ("an object is an abstraction of a set of real-world things such that all the real world things in the set –the instances- have the same characteristics" (Shlaer and Mellor, 1992, p. 67). What do they really mean by this? Is it not an objectivist point of view to be refused? Or is there some clear understanding of what happens with OO models?

Experientialism provides us with useful tools in order to refuse the objectivist point of view. The way objects are defined in OO programming languages, design and analysis show the immediate difference between an *invoice* object and a real world invoice: we cannot send a message to a real world invoice in order for it to print itself, nor can we send it most of the messages an invoice object can receive and respond to.

So, the first point is that objects –conceived as conceptual integration from various input mental spaces: real invoice object and the workplace- are a cognitive construct, a resulting frame which can be used for technical purposes. We may conceive this cognitive construct as the models (analysis and design levels) used to build object with object-oriented languages. As can be seen, these last objects (the invoice object) in turn, are also objects of the real world. Strictly speaking, even the artifacts representing the cognitive constructs are also objects of the real world. So, it would be useful trying to differentiate the various meaning of the word *object*: something of the real world (an invoice), a cognitive process (the elaboration of a conceptual integration), the representation of the result of such a cognitive process (model), and the implementation of the model (using a programming language).

In this sense, OO objects are close to the original real world objects in that they share the same names and most of the abstracted attributes we use to refer to them. But one must remember that attributes included in the implemented object (for example, a Java object) are abstracted from clusters of interactional properties we ascribe to real world objects and not

attributes of the real world objects. Once implemented, the behaviour of OO objects belongs to the realm of real world objects and attributes.

The main advantage of OO methodologies is that we need not start from a quite general concept equivalent to a Context Process and continue going down through various levels. In OO methods the starting point is a set of conceptual objects, which can be directly derived from the domain of study. In this sense this initial model is the equivalent of the Entity-Relationship (ER) Model without the need for an heterogeneous model such as the DFD.

Even if the starting point is better in OO than in Structured Methods, we need additional important information in order to specify what methods we are to include into objects. This additional information is the set of interactional properties of the system to be designed, the set of *basic-level events*.

We have to differentiate between two basic-level concepts. The first one applies to the equivalent of basic-level categories: when conceptualising a given domain, those basic-level categories are expressed by ER models or by OM (Object Model). Such things like entities or conceptual objects are our basic-level categories to be used in designing a system.

The second type of basic-level concepts corresponds to basic-level events, which provide the additional information for modelling.

In terms of mental spaces, we can consider that the *first step* of design –analysis model- is obtained from a blend of two input mental spaces: the first mental space is composed by basic-level categories (entities or conceptual objects), the categories we use in the domain we are to create artifacts in; the second mental space is composed by basic-level events, which are derived from the analysis of the interacting properties of the artifact to be created.

A *second step* of design –the design model- could be obtained applying a second blend to an input mental space that is, in fact, the blend we got during the first step -analysis model- and a second input space which regards the technical environment –constraints-, in which the system is to be implemented.

3.4.2 A design model with mental spaces and blends

In this model -drawn in terms of mental spaces and blends- we can see how mental spaces and blends are related through projections. These projections are not directed as far as constructing one mental space a reverse projection is considered. For example, when considering the structure of both basic-level categories (entities or classes) and basic-level events, some reverse projections can be applied as to modify the requirements of mental space.

The requirements of mental space is conceptualised in terms of stories of the domain, so projecting rich structures to be applied to all other mental spaces including the design blend space. Unlike conceptual models and methodologies there is no sequence of steps to be applied in order to construct mental spaces since all of them have to be considered in a simultaneous network of mental interactions; a modification in one space can disseminate structure modifications in all other spaces. Turner (1996) has pointed out this aspect:

> "A blended space has *input spaces*. There is a partial projection from the input spaces to the blend...Sometimes, those input spaces will be related as source and target,

> in just the way we have seen so often. Crucially, blended spaces can develop emergent structure of their own and can project structure *back* to their input spaces. Input spaces can be not only *providers* of projections to the blend, but also *receivers* of projections back from the developed blend" (p. 60. italics in original)

Only some of all mental spaces participating in the cognitive process of *design*, are represented in the model. When we conceptualise basic-level categories in terms of entities or classes, it is evident that the real space we are considering is a blend of *domain* basic-level categories and some frame or schema such as Entity-Relationship or Object-Oriented methods. The domain basic-level categories are extracted from the stories of the domain. Similarly, basic-level events are already produced in a blend derived from basic-level *actions* of stories of the domain and some frame or schema such as Use-Cases or Scenario techniques.

Another important question regarding basic-level events is that these events correspond to interactions with the system to be designed and not to events of the domain in general. Thus, some events of the domain can be modelled by entities or objects and are not considered basic-level events.

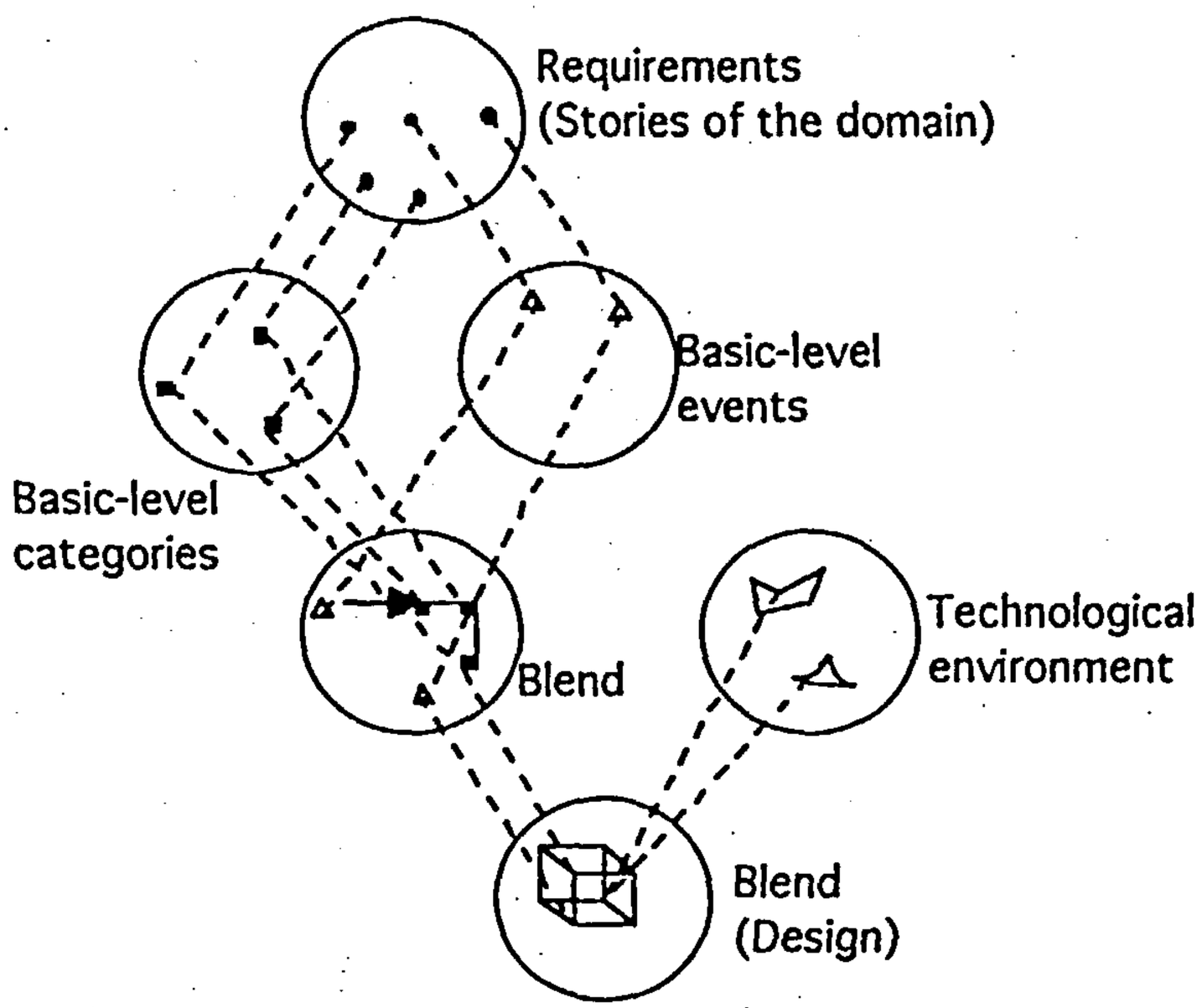


Figure 18 Chain of Multiple Blends

Being this model a cognitive one it describes general aspects of methodologies whether it be structured, object-oriented, or functional, etc. Using the former example of an invoice, we build an input space with attributes of invoice derived from the domain of invoicing. These are the basic level categories of the entity (or object) invoice, represented as attributes. The basic level events are activities performed in the context of invoicing (creating an invoice, sending or receiving an invoice, printing an invoice, etc) and are represented as actions or operations. The

first blend corresponds to the module or object as represented by the analysis models. The input mental model of technological environment comprises features such as programming language, database systems, operating system, performance constraints and so on. So, the result of the second blend will be the modules or objects implemented in the environment in which they are to run.

3.4.3 Basic-level categories

In OO methods, the main starting activity is choosing classes, or basic-level categories. Another activity is that of finding categories at a higher level. In this context it would be interesting to separate two different concepts: *classes* and *categories*, even recognising that classes are also categories. The difference could be established in terms of classical and non-classical categories, the former corresponding to classes and the latter to what we defined as categories. The classical example of categories, such as defined here, is a command in a menu bar. In the word processor I am using, the Edit command comprises the classical editing functions like Cut, Copy, Paste, etc., but also such functions like Find and Go To, which cannot be considered editing functions the same way as Cut and Copy.

Some recent papers recognise the important role of cognitive principles in conceptual modelling. "It appears that the most fundamental modelling activity -choosing classes- is not guided by general principles and hence that choices of classes cannot be evaluated systematically...Since classification is intended to represent human knowledge of a domain, choosing classes should be guided by cognitive principles (Parsons and Wand, 1997, p. 64).

The authors propose such concepts as *Potential Classes* and *Class Structure*:

In Potential Classes, a guiding principle is that *of abstraction from instances*, which means that if no instances from the relevant universe share a specified set of properties, there is no economy of representation and therefore it would be cognitively wasteful to store such a set of properties. Even if this principle is perhaps obvious for abstraction from a given set of objects, it is less obvious for methods based on inferring new classes from existing ones. (op. cit., p. 65).

Another guiding principle of Potential Classes is that of *Maximal Abstraction*, which refers to the condition that if certain properties shared by all instances are not part of the class definition, some similarities among the instances in the class are lost

In Class Structures in order to eliminate loss of information and minimise redundancy two principles are proposed: *completeness* and *non-redundancy* .

Completeness indicates that in a relevant universe of instances and properties, every property should be used in the definition of at least one class in the set of classes. It is interesting to observe how the authors in the abstraction of instances and properties have created two different mental spaces and thus consider that "if a property is part of the relevant universe, instances possessing it have 'meaningful differences' with respect to those that do not" (Op. cit. p. 66). In this conceptual framework that refers to classes, instances and properties are considered independently, as if properties are not interactional -physical and conceptual- attributes derived from concept meaning (class definition).

The second principle is nonredundancy, indicating that if a class is a subclass of several other classes it should be defined by at least one property which is not in any of its superclasses. Thus, no class is defined only in terms of the properties of a set of other classes.

Regarding Class Structures, two main properties are considered: *multiplicity* and *inclusion*. In line with results of cognitive sciences, *multiplicity* means that more than one class structure may exist for a relevant universe having more than one property. This statement refers to the fact that we usually witness several views of a domain in coexistence and all of them supporting cognitive economy and inference.

Inclusion states that every potential class can be included in some class structure. Inclusion ensures that every concept of interest can, in principle, be included in a model that supports cognitive principles.(Op. cit. p. 67)

The concepts presented so far in relation with classes, instances and properties reflect how useful cognitive considerations could be and thus they should be employed to gain additional knowledge of conceptual modelling.

3.5 Categories and classes

An interesting (an important) issue is the discussion about categories and classes. It is clear that classes are classical categories, but are there some categories that are not classes? In such a case, what type of relations can be established between them? And, mainly, are categories useful in the context of object-orientation (OO)?

3.5.1 Non-classical categories

As we have already seen, HCI methods are based on classical categories with set-theoretical properties. When we analyse entity or class diagrams, we find a variety of constructs whose goal is to address some of the problems derived from having to treat non-classical categories. For instance, the usual case of trees with multiple branches are used to include a wide range of quite similar classes. Many authors have pointed out the difficulties of treating inheritance through multiple levels. Subclassing a class A directly to give it different behaviour implies hard-wiring the behaviour into its own class. But this approach makes the class A harder to understand, maintain, and extend and one cannot vary its behaviour dynamically (Gamma et al., 1995).

Another issue is the restriction of moving an instance -an object- from one class to another. The usual solution is to eliminate the first instance and to create a second one from the new class that we want the latter to belong to.

When dealing with non-classical categories, it is not only a question of attributes whose values could be set to null in order to represent instances with variations. The problem is associated with behaviour that could or could not be present in different instances. To think of objects (or entities) as elements, which may only vary values of the associated attributes is to think of them in terms of classical categories, as we usually do.

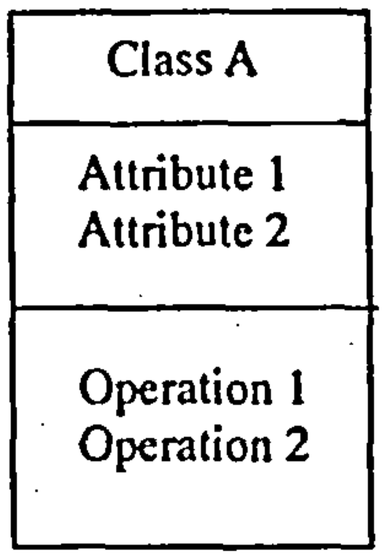


Figure 19 A Class as a Classical Category

Remember that a non-classical category represents a collection of elements around one of them that can be considered as prototypical. The sharing of features between all elements of the category is not a differential characteristic since the sharing occurs between some elements separately. We could use object-oriented notation to represent non-classical categories. In order to do so doing we need a mechanism that allow us to:

- Add new behaviour
- Change a behaviour

This is equivalent to dynamically changing one object from one class to another, as when stated in terms of a hierarchy of classes. Unlike a classical category –a class- a non-classical one has to be defined as a network of classes. In this case there is no fixed class to define our objects but a configuration of classes with associations between each other. Each instance of a non-classical category implies a different configuration of classes.

The proposal of using patterns to include additional behaviour in some circumstances is a valid one to plan a more general solution to the problem of representing general categories. The pattern suggests the possibility of constructing a main, central class around which we include a network of associations with other classes. We can conceive this central class as a minimal nucleus of shared attributes or even as a mere support to integrate the rest of the network.

The pattern used for such a purpose might be that of "strategy". Such as explained by Gamma et al., where we use the Strategy pattern when "many related classes differ only in their behaviour. Strategies provide a way to configure a class with one of many behaviours" (op. cit., 1995, p. 316).

This is only a general and theoretical analysis, without taking into account issues like the number of objects implied by the pattern ("strategies increase the number of objects in an application" (op. cit., p. 318)) or the need for clients to be aware of the different strategies ("a client must understand how Strategies differ before it can select the appropriate one" (op. cit. p. 318). We only intend to show an approach capable of representing non-classical categories without optimisation considerations.

Figure 20 shows the structure of the strategy pattern, which can be repeatedly applied.

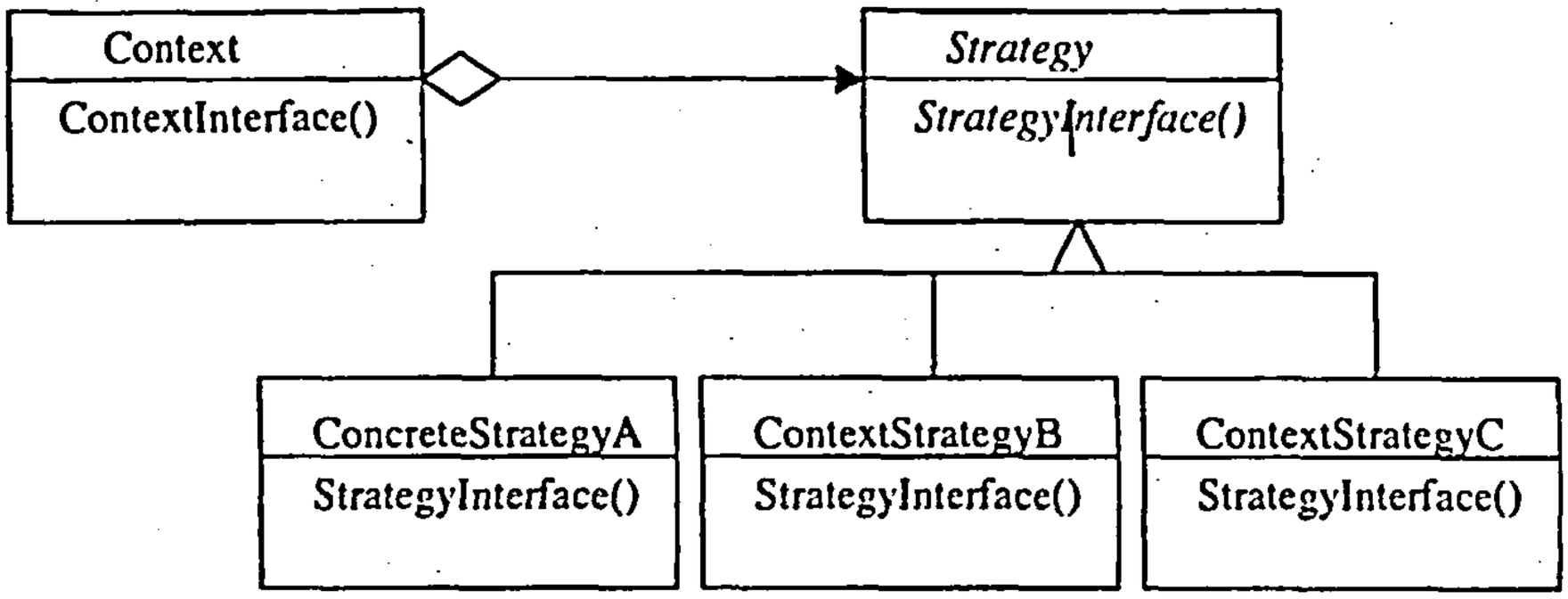


Figure 20 The Strategy Pattern

An example of a class designed according to the strategy pattern is an Employee class. If a company has different types of employees, classified according to the way they are paid (weekly, bi-weekly and monthly), we may associate the context class (Employee) to another strategy class, for example, Type of Employee. This strategy class is, in fact, an abstract class with three different subclasses: Weekly, Bi-Weekly and Monthly pay. So, we get a flexible category of employee by associating the original Employee class with an object corresponding to one of three classes: weekly, biweekly or monthly pay. This strategy pattern avoid the problem of having different classes for each type of employee and the subsequent problem of deleting and creating employees objects each time an employee changes his pay method. Using this strategy pattern, changes in the way of paying will result in changing the associated pay subclass.

The next figure shows how this pattern could be applied to a general schema, with a central class and a few features that may be added to the central class. Each of these features –represented as abstract classes- has many *concrete features*. A concrete instance of the non-classical category would be represented as a network of objects, each one contributing a set of attributes and functionalities.

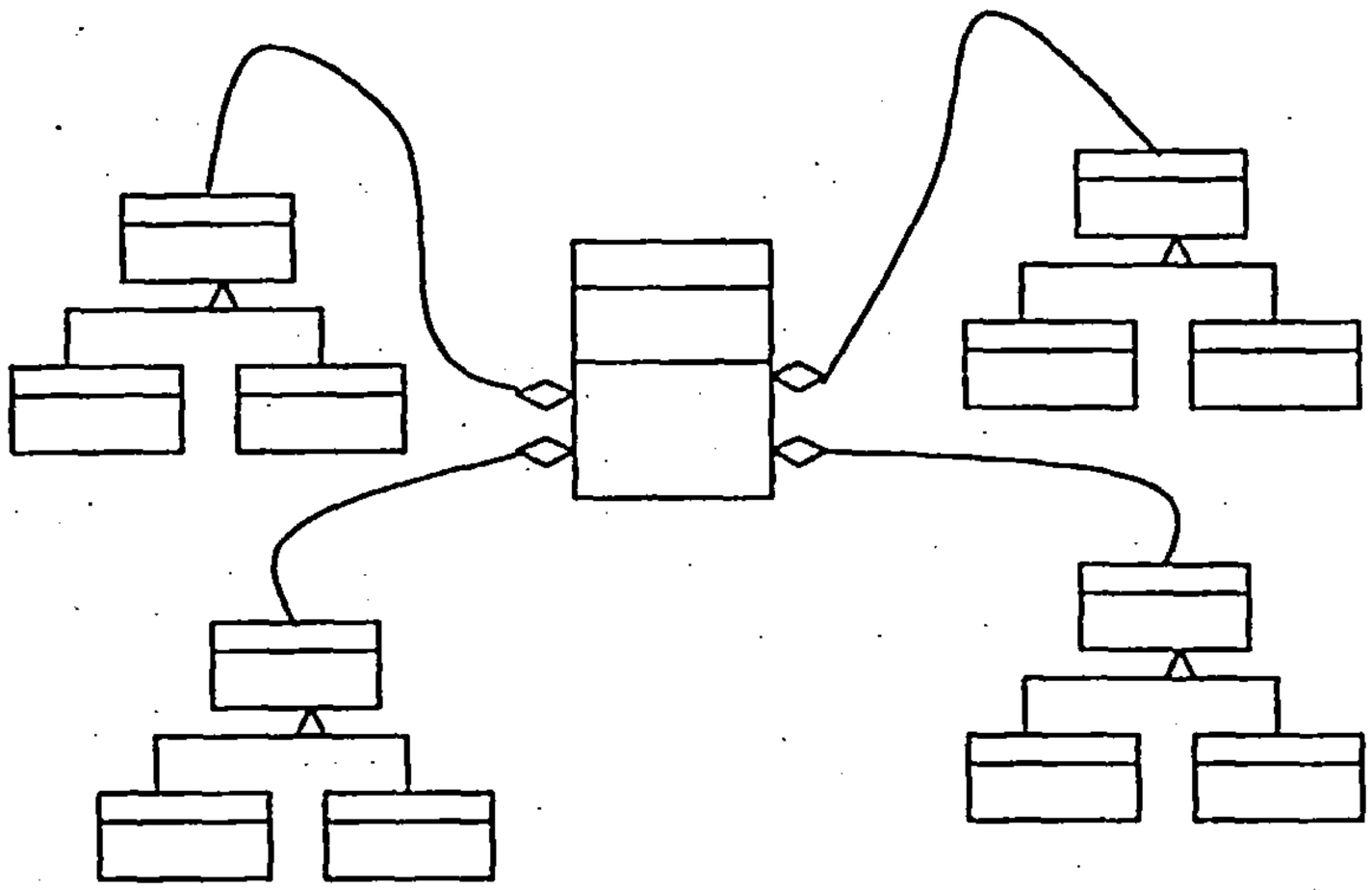


Figure 21 A Diagram Representating a Non-Classical Category

In section 2.3.3 we have presented the concept of classical and non-classical categories. A recent article (Keepence and Mannion, 1999) also describes some patterns for solving the problem of defining families of products. In the case of reuse approaches, the aim is to maximise the reusability, so a maximum variability is desirable. But the problem with usual approaches like the generic-system one is its limited effectiveness, as the number of family products increases, the number of common features decreases. It is clear that we are facing a non-classical category, even if the authors do not employ such a term.

The interesting point is their definition of a *family*: "for example, in a family of four systems, A, B, C, and D, there often exist common features between A and B, B and C, and A, B, and D, but rarely is one common to all systems" (op. cit. p. 103). This definition fits the Wittgenstein's concept of *family resemblances* exactly: the classical category has clear boundaries which are defined by common properties, but a category like game does not fit the classical mould, since there are no common properties shared by all games.

Members of a family resemble one another in various ways: they may share the same build or the same facial features, the same hair colour, eye colour, or temperament, and the like.

"But there need be no single collection of properties shared by everyone in the family" (Lakoff, 1987. p. 16).

The chosen metaphor –that of considering things or objects as members of the same family- has a high degree of expressiveness.

Even if most categories could be considered as non-classical, the fact that simplifying them to adapt to usual HCI notations as classical entities or classes may have some practical advantages. It is easier to treat a collection of simple defined classes with a set of shared attributes and behaviours than to consider a much more complex schema that takes into account the maximum variability. But it would be interesting to consider the design problems

that appear indirectly as a consequence of such over-simplification of the definition of classical classes and –mainly- to be aware that *natural* categories are non-classical categories.

3.5.2 How the concept of category may be used in HCI projects: The Flex Project

We are going to see how this concept of categories may be applied to a concrete project. In the previous section we have seen that the concept of category is broader that the classical concept of *entity* –structured methods- or *class* -object-oriented methods. While this is the usual way we can consider the concept of category, there are other uses where the concept may be applied: the group of options we introduce in a general list (menu) of possible functions.

The way we group a set of functions is usually quite a difficult issue. Normally in a given word processor there are categories obviously based on activities like *edit*, *insert*, *view* or *format*; but other categories have been conceptualised in terms of artifacts (*tools*) or objects (*fonts*) we deal with when editing texts.

As we will see in section 7.2.3 there are categories built on their relational properties, not on their intrinsic characteristics, supposing we can speak of such a thing as an intrinsic characteristic. These relational characteristics are relational because they are related to human activities in which we are continuously producing new categories.

One of the examples we offer in Chapter 7 –among others- is *foods not to be eaten while on a diet*. Is this an inner characteristic of foods? Obviously, not. It is a relational characteristic derived from our need to lose weight, to perform an activity with such a goal in perspective and, consequently, to choose a given set of foods.

There are other nice literary examples. The classical one is the Borges' classification –categorisation- of animals:

> *On those remote pages it is written that animals are divided into (a) those that belong to the Emperor, (b) embalmed ones, (c) those that are trained, (d) suckling pigs, (e) mermaids, (f) fabulous ones, (g) stray dogs, (h) those that are included in this classification, (i) those that tremble as if they were mad, (j) innumerable ones, (k) those drawn with a very fine camel's hair brush, (l) others, (m) those that have just broken a flower vase, (n) those that resemble flies from a distance.* (Jorge Luis Borges. *Other Inquisitions*)

Once observed in detail these categories are derived from a set of activities. The first (a) derives from the Emperor's breeding of animals, the second (b) from the activity of embalming, (c) from the activity of training animals and so on. Even those apparently odd categories like *fabulous ones* or *those not included in this classification* may be derived from human activities such as imagining fabulous tales (or novels) and from classifying objects in function of their belonging or not to previous categories (based on activities).

What about (m), *those that have just broken a flower vase*? Considering our concern of maintaining or preventing damage to the home or palace furniture (or pottery), it is evident that those animals that broke a flower vase are immediately classified in this new category. It must be taken into account that even conceptual integration is an activity, as well as all cognitive process in general. So, even a rare literary piece aimed at producing extravagant classifications like Borges' text may be explained on the basis of human activities. All these categories have

one thing in common: they are activities applied to a given category of (self-animated) objects: animals. Even stray dogs derive from the activity of keeping an eye on the group of dogs.

The Flex Project is described (See Appendix A) as being aimed at providing "the technology for bringing the information age into the home for everybody to use. The target product is a new concept for a device called the Home Information Center (HIC), which describes a device for multimodal access to information and services relevant for the everyday life in family. The FLEX project will provide the FLEX Technology software for supporting intuitive navigation in and visualisation of information and flexible queries for information, as needed for the realisation of a HIC" (Description of the Flex Project, 1999).

In Flex, the concept of category is also bound to a given activity or purpose as can be deduced from the following definition: "For practical systems designers will select dimensions considered useful by a person or a community for classifying a given collection of information items for a certain purpose" (Information Space: analysis and definition of its central concepts. Contribution to D2.1.1, 1999).

Some comments made at earlier design stages of the Flex Project (Elaboration of Scenario 99-01-4) point out that some information spaces being dealt with are essentially static: "When thinking on how to classify an information space (IS) like the cooking book it's important to realise that in this case we are dealing with an essentially static space composed of reference information as opposed to essentially dynamic spaces (made of temporary information) like newspapers". They conclude that an initial "consequence of dealing with an (essentially) static IS is that its categorisation space is equally static".

We consider that there are many categorisations in the IS involved in HIC. One set of categories is that which is applied to a book of recipes (ingredients, geographical place, food type, difficulty and so on), and that can be considered as fixed because the content provider –provider of the recipes book- has already applied such categories to the recipes of the book.

There are other categories that may be applied to *My Notes* IS. This is an essentially dynamic Information Space in which we include whatever may be occurring in our everyday life (parties, people attending a given party, recipes for a party and so on). This dynamic characteristic of the IS implies that we may be able to make any type of query about the information space. In this sense, each query gives a category as result.

In Scenario 99-01 there is a description of Pia selecting *Guests Preferences* (a previous category –possibly- created by Pia herself) in which a list of people may be searched. From this list Pia selects *Mary Morrisey* because she vaguely remembers that Mary suffers from some type of food intolerance.

Once the party in which Mary Morrisey has participated is over, Pia may introduce the menu she chose for that party, in order for the next time she will invite Mary to select "menu of dinners with Mary Morrisey" because she likes to change menus when inviting the same people. This would be a dynamic category of "people and menus in parties", not previously defined as a fixed category. As queries evolve and apply to the same or different IS, the results we get are a new IS so we may want to store it as a new IS.

The fact of using structured information in order to store it in an Information Space with a fixed set of categories may determine that the categorisation space will be static. But other IS like My Notes may have dynamic characteristics, derived from the fact that the user introduces

notes like "27/11/99. Dinner with Mary Morrisey, X, Y and Z. Menu: Shiitake Mushroom and A, B and C". If the query is a search by content, it will be possible to detect a category of parties in which Mary has been invited and a given menu was served, or a category of menus offered when Mary was invited.

The usefulness of a IS is derived from its capacity to face new situations, in this case to create new categories as needed by the users. The maximum flexibility is to employ all words –concepts- used to describe a given story (the party) and use them to make queries that may produce some new categories.

We have to think of Information Spaces as sets of stories described by a set of words. When we decide to categorise –and, at the same time, to reduce the amount of information stored in the IS- we withdraw some of the words that constitute the categories and we put them together in separate locations and add links from stories to the words that describe the categories. This has a negative side since it makes the structure rigid and we can only search by the group of categories already established. The obvious consequence of categorising using the whole set of words employed in the story description is that we can use any of these words in order to categorise such stories: we may create indexes using each word in order to make searches more performing.

Chapter 4

Metaphors and Figurative Language

4.1 Metaphorical is not literal

We have already discussed the general concept of metaphor in Chapter 2, in the context of Experientialism. Now we will see different views of metaphor, in particular some points of view that question the validity or the usefulness of metaphor. These points of view have, in general, the common ground of literalness: it is generally assumed that what we use in any scientific domain is a *literal* language, not a figurative language or a language full with metaphors.

In this traditional, folk theory, the literal meaning is not seen as just one kind of meaning among others. In fact, very few theories of linguistic meaning offer any explicit discussion of literal meaning, as it is usually assumed that the only theory of meaning that could exist is a theory of literal meaning and nothing else (Gibbs, 1994).

One consequence of this traditional theory is that there must be literal concepts and literal meaning in order for people to communicate successfully, given our different subjective experiences. As Gibbs points out "the idealized, mythical view of literal meaning as being well specified and easily identifiable in thought and language is incorrect. It is, in fact, quite difficult to specify the literal definitions and concepts and the words that refer to these concepts" (op. cit. p. 26).

There are many arguments in favour of speaking metaphorically, the more relevant being that of the amount of metaphors we use in everyday language without our noticing it (Lakoff and Johnson, 1980). Among the arguments employed to show why the use of metaphor is not only pervasive but also necessary for conceptualisation, we can recognise the three following.

4.1.1 Metaphors are expressive

In Chapter 5 we will see a collection of examples which illustrate how we continuously use conceptual integration based on metaphors in software development and methods. One of the reasons for using such concepts like *layered architecture* or *broker*, is that metaphors provide a way of expressing ideas that would be extremely difficult to convey using literal language or extremely long without contributing in clarity. This first aspect of the communicative function of metaphor is called the *inexpressibility hypothesis* (Gibbs Jr., 1994, p. 124) and refers to the fact that literal language usually lacks the expressiveness to predicate of our thoughts. Such as pointed out by the author, expressions like *The thought slipped my mind like a squirrel behind a tree*, even when we try to translate it in literal language, we have to use a language that is essentially metaphorical (e.g., *The thought went away* and *The thought evaded me*). The last two expressions use the classical metaphor of thoughts as auto-animated objects, capable of coming in and going out.

4.1.2 Metaphors are compact

As can be observed in most cases, metaphors have the function of providing a particularly compact means of communication. This is the *compactness hypothesis* (op. cit, p. 125) which summarises the idea that language can only partition the continuity of our conscious experience into discrete units comprising words and phrases that have a relatively narrow referential range. In this case –and using another metaphor- it is a way of freeing language from an oppressive situation. In general, literal language does not enable people to convey a great deal of information succinctly in the same way that metaphor does. In the case of *layered architectures*, we attain a homogenous image with only one concept –that of *layer*- to convey a set of different dependency relationships between pieces of software that would be longer to express.

4.1.3 Metaphors are vivid

Another interesting issue of metaphor is its capacity to capture the vividness of our phenomenological experience. This *vividness hypothesis* manifests the capability of metaphors to convey complex configurations of information rather than discrete units and so enabling the communication of more vivid images of our subjective experience. Although this vividness capacity to convey subjective experience would apparently be less useful in our HCI methods, it is a matter of looking for some icons used in the user interface. The fact is that vividness is not as rare as it would appear as we can see by the use of the bomb icon in the Mac interface, which shows how vivid could be the metaphor used to express a troublesome situation and the potential reactions from users as we will see in Chapter 5.

4.2 The desktop interface metaphor

The desktop interface is a well-known example of discussions about its being either a metaphor or not, so we are only to give a brief introduction in the light of our previous analysis of mental spaces. The Mac interface was built on the basis of two conceptual inputs: one is that of ordinary work in an office with folders, documents, a desk, a trash, and so on. The other is the traditional field of computer commands, sent to the computer in the form of an expression in a specialised language.

The question of considering the Mac interface as a metaphor is based on the idea that most of the traditional functionalities of ordinary work have been maintained as expressions in everyday interface tasks. We *open* a folder (or a document), we *close* them, *throw* folders or documents into the *trash*, we *empty* the trash, and so on. The tasks we do when dealing with the interface mimic our ordinary office job without thinking of the equivalent computer commands. As this metaphor has limitations other interfaces have tried to overcome them by extending it to some other room or building metaphor as the Magic Cap (Gentner and Nielsen, 1996).

So, both input spaces participate in a metaphor before they are included in a blend relationship. This is the reason why the desktop interface is called a metaphor straight out. But when observed in detail it is evident that we are dealing with a blend rather than a metaphor; the blend being based on the metaphor.

As usually occurs with blends, the desktop interface has a new emergent structure. The Mac interface has the particularity that any loaded diskette can be unloaded by dragging it to the trash, something that would rarely occur in the real world. The same way, we can close a document by clicking on an associated button or open folders by double clicking on them. These are new functions that neither exist in the real world, nor in the universe of classical computer commands. They are emergent functions in the blend space, a new conceptual domain, which has its own characteristics to be considered when this blend is used to construct new metaphors and blends.

4.2 Metaphor as illness

Things have changed so much that we have moved from illness as metaphor (Sontag, 1988) to consider metaphor as harmful (Halasz & Moran, 1982) or even pathological (Monin & Monin, 1994). In this last paper the authors affirm that "there is a danger...that if human attributes are ascribed to the computer, personnel in IS begin to act out of metaphor pathologically" (Op. cit. p: 283).

Let us listen to their arguments. Based on convincing arguments, they point out that some authors "remind us that classical management theory expects people to perform like 'inanimate cogs and wheels'...and demonstrates that Taylor's ideas, built into the technology itself, make 'the workers servants or adjuncts to machines that are in control of the organisation and pace of work' " (op. cit. p. 284). Indeed this is a way of producing social agents through a social discourse in which there is an underlying metaphor whereby human beings are machines. Other authors (Foucault) have demonstrated that the consequences of this type of discourse is the production of *subjects*; in its etymological sense of human beings that are *tied* by social relationships.

We cannot deny the social or even psychological effects of this type of discourse based on the underlying metaphor *human beings are machines*. But looking into it in some details, the real dangerous effects on workers is not a metaphor per se, but the *whole discourse*, which can be based on other metaphors as well. And it is evident that discourses are not isolated but they are embedded in social practices, the total effect potentially being dangerous. That is why Foucault treats discourse as an assemblage of 'heterogenous' elements; techniques, artifacts, practices, experiences, fictions and language, including metaphors (Agre, 1997).

Once accepted the potential risks of some discourses based on some metaphors -because the *target* domain of the metaphor is the *human being*- we can no longer agree with the authors when they say that "a more significant metaphor is the inversion of this image: that computers have been personified, and as a consequence our expectations of their performance is modified by this linkage: that furthermore, it is because we are largely unaware of the emotional persuasion of this metaphor that we are so easily influenced by it". (Op. cit. p. 284).

It is difficult to see, such as the arguments the authors employ try to do, any symptom of danger or illness derived from the fact that, for the last forty years, we have been talking about *booting* the system (or *bootstraping*), *aborting* a command or application, *purging* and so on. Most of people around us continue being as neurotic or 'normal' as before starting to use such computing concepts, and nobody thinks of real life *abortion* when this problem occurs during system *execution* (in fact, has anybody thought that one can be *executed* in the electric chair when speaking of system execution?).

As it is said in the cited paper, "a new and vital metaphor may, through repeated use, evolve into a new literal meaning, or polysemy". In that case, why fear the effects of the original metaphor since a new meaning has emerged?

The authors add that some of the implications are that "computer users everywhere may have been persuaded, through the language of applications, architects, and end-users to think of computers as having the powers, potentialities, and personalities of humankind" (Op. cit. p. 287). We would like to say here that what has really spread the idea of computers as human minds are AI and expert systems disciplines. As it has been shown in Chapter 1 (Experientialism), cognitive semantics aims at criticising this conception because of the erroneous idea of knowledge that it embodies, but it is not a question of the dangers of metaphors themselves.

The real danger appears in other discourses, such as the Cyborg one. This "is the discourse of human-machine symbiosis: treating the whole world as a militarized command-and-control structure that operates on rational, technological principles, from the highest levels of global strategy to the lowest levels of individual cognition" (Agre, 1997, p. 31). Agre is describing one of both discourses analysed by Edwards (1996), the other being the closed world discourse. Edwards' idea about discourse is similar to Foucault's: "A discourse, then, is a way of knowledge, a background of assumptions and agreements about how reality is to be interpreted and expressed, supported by paradigmatic metaphors, techniques, and technologies and potentially embodied in social institutions". (Edwards, 1996, p. 34. The quote is Agre's).

As it may be clearly seen, metaphors are only a small element on which other complex constructions such as discourses or ideologies are built. "Unfortunately, -explains Agre- the most common response to such a discovery is to place the blame on metaphors themselves, rather than on the unreflexive neglect of metaphors and their consequences"

4.3 Some other uses of the word *metaphor*

In our discussion we have tried to clearly address the sense of metaphor as a language expression and as a cognitive process that could be derived from language expressions. But there is another sense that can be found in the HCI literature. For example, T. Ericksson (1990) starts his paper giving the classical definition of metaphor and clearly explaining how we use them in our ordinary language:

> "Without any intention of being poetic or fanciful, we speak of argument as though it is war. Arguments have *sides* that can be *defended* and *attacked*.. Facts can be *marshaled* to support one's *position*; *strategies* can be employed..." (Ericksson, 1990, p. 66)

Indeed he includes a paragraph that points out the close relationship between body and mind, a relationship studied by G. Lakoff and M. Johnson in some of their books. "It's also important to note that we don't just talk about arguments as though it were war: being in an argument feels like a conflict; when we *lose* an argument, we feel bad". (Op. cit. p. 66) (Underlined in the original).

So far, a pure and correct definition of metaphor. The problem starts when Ericksson speaks of introducing the metaphor in the interface, giving at this point a new sense to the word metaphor. It could be said that this new sense is a metaphorical use of the word metaphor.

Let us see how this change works.

> "Just as metaphors invisibly permeate our everyday speech, so do they occur throughout the interfaces we use in design. Just as we use military terms to make the rather abstract process of arguing more tangible, so we use object and container terms to make the Macintosh file system more concrete. And just as we experience arguments as real conflicts, most Macintosh users believe that when they move a document icon from one folder to another, they are really moving the document itself (what is 'really' happening is that a pointer to the file is being moved -of course, pointer is a metaphor too...)" (Op. cit. p. 66).

There has been an imperceptible change, even if it actually was a somersault. When analysing this example, we have to separate different layers as we usually do in computing. On each layer, there is a logical language that has to be translated to a different one, on a lower layer. When we employ the sentence «*Perform until*» in a language such as Cobol, we are using a construct that exists at the Cobol layer, but not necessarily at the layer of assembler or machine code. The word *until* has a meaning that does not exist in lower levels. At lower levels there are groups of instructions executing and asking for a condition to be satisfied. It is the higher level that adds meaning corresponding to the included *intentionality*: we repeat the group of instructions *until* the condition is satisfied. That is why the *Perform Until* construct seems to us as completely literal, reflecting exactly what the construct does; in fact we are embedding our intentions in the expression without our being conscious of it.

In the case of the interface, there is a metaphor on which the computing mechanism is based: the desktop metaphor, as a thought process. This metaphor inspired the artifact we usually use in order to move one document from one folder to another. As the computer artifact is inspired on some actions from the real world, it is logical that we use the same expressions as those employed in our everyday workplace. But we deal with icons (graphical representations of some other objects) that are blend concepts and not metaphors any more: they are derived from the real world and from the computer domain giving rise to a new, synthetic concept. A proposal (by B. Nardi) we are going to see later, calls these synthetic concepts *visual formalisms*.

So it is clear that there is a difference between some cognitive mechanisms we use in design in order to create new computing artifacts (which in the case of graphical interfaces could be named visual formalisms) and the artifacts themselves which are used in an operating interface environment.

If we use the word metaphor when speaking of documents, folders and so on, why not apply the same criterion and say that *Perform until* is also a metaphor. This way we would continue calling most of the computing constructs metaphors.

Pattie Maes proposed, during an interview (Maes, 1997), a biological metaphor in connection with intelligent personal agents: AGENTS ARE INSECTS. Actually, this expression could be derived from her comments, but the important thing is that the metaphor brings to us an input mental space, the world of insects with a set of actions they do. We think immediately of some of their characteristics: the mobility, the way they interact ("for example,

if a couple of ants put down some food somewhere, then other ants are more likely to leave their food in the same location. Suddenly that's where the whole ant colony ends up storing its food" Op. cit. p. 17) In an ant society none of the components are critical, they leave pheromone on some objects they interact with, and so on. This is a valuable source of inspiration, as it is possible to project some of the structure of this input space to the blend: "agents could leave the digital equivalent of a pheromone at documents they deemed relevant to their users, thereby attracting more agents toward those same documents". (op. cit, p. 17).

This useful idea has to be refined and developed, it is a starting point, a raw material that could finally become a personal document agent, the same way that all initial metaphors are to be refined –transformed- in order for them to become useful computer-based signs. When interacting with this type of agent it would be possible to say -as P. Maes does- that the agent *leaves a pheromone* on documents and so speak of the agent as a metaphor. But it is clear that the agent is the final result -a software object- of an initial metaphor and not the metaphor itself. It is a quite acceptable position to call the agent a metaphor, but while taking into account that we are facing two different senses of the word.

As we will see in Section 4.10 about literal and figurative language, the metaphor is very useful as a source of inspiration, as the idea of pheromone –continuing the figurative language- is the seed to produce an equivalent mechanism that could *attract* other users to some specific documents when querying the document database. Summing up, what we propose is to maintain these figurative descriptions –as usually occurs in everyday computing language- of agents as insects, leaving pheromone on documents, moving along different virtual spaces, etc as they are essential to a creative design process. Maintaining such descriptions -with a suitable mechanism to trace the path from the source figurative language to the implemented computer based signs- would help us to produce new ideas based on the same original metaphor.

In one of my jobs I remember a programmer who explained out loud to her colleagues the program she was developing. It was usual to hear her comments this way: *I* go to this point, then *I* move the content of X to Y and so on. This was a very particular (and interesting) way of saying what happened with the program. It was a way of anthopomorphising the computer program. In this case, a quite personal metaphor appeared: *The program is Me*. The metaphor occurs in a different space: the relationship between the programmer and the program; it exists in language and thought. It is not a quality or object belonging to the computer, neither to the graphical interface; it is a pure human quality as only human beings are creators of meaning.

4.4 Beyond Models and Metaphors

Bonnie Nardi (Nardi and Zarmer, 1993, p. 5) argues that "basing user interfaces on mental models is...impractical, and provides a confusing design guideline for developers. Metaphor are useful for some purposes, but inadequate to serve as the basis for interfaces of any complexity which must express their own rich semantics". She proposes *visual formalisms* as a basis for interface design.

As Mental Models have received criticism from many different directions (cognitive semantics, situated cognition and activity theory) we are not going to repeat here any of their arguments. Our interest is to analyse Nardi's arguments in relation to metaphor and her proposal for visual formalisms.

Some of the issues Nardi points out are:

4.4.1 Metaphors lack precision

"Metaphors are good at suggesting a general orientation, but not good at accurately encoding precise semantics" (Nardi and Zarmer, 1993, p. 17). This is a quite acceptable position, considering that a metaphor is a very general and vaguely defined mapping between mental spaces. The question is that, in the design process, we are looking for new and helpful ideas that can contribute to our final product.

The use of a good -and sometimes various- metaphor is a way of going in the right direction, a general but useful orientation. But the most important issue is that both input mental spaces produced by the metaphor are the raw materials to be used in the refinement process of blending. There is a set of ideas from the source mental space that is automatically incorporated in the blend, as well as a selectively projected structure producing a coherent, integrated, emergent structure specific of the blend

It is due to the massive projection from familiar inputs that we are able to conceive a totally novel activity that is immediately congenial and accessible (Fauconnier, 1997, p. 172).

4.4.2 Metaphors and the practical programmer

Nardi continues claiming that she sees few opportunities to create reusable metaphors. She argues against the argument that metaphors are a basis for user interface design, exposed by Carroll and Thomas. This is the time to point out that we are likely facing the second meaning of the word metaphor. Considering metaphors in this latter sense, in which there is neither a well defined computing concept nor a cognitive one, it is evident that they "do not promise to provide a rich set of reusable application objects that can be bundled together into toolkits to support the development of specialised applications" (Nardi and Zarmer, 1993, p. 19).

Such as we have seen, based on a metaphor (the interface as a desktop) we have used a blend in order to produce some of the computing constructs. The final product is a module, an object, a plug-in or anything you want. You can even call it a metaphor, but clearly differentiating that they are two different concepts: one cognitive and the other belonging to computer sciences or programming. The confusion derives from taking two different things using the same name. The computing construct will be as reusable as any other object could be. The case of the Next operating system shows us how many of the interface objects are reusable by the new applications developed with it, supposing that folders, windows and trashes are metaphors in the sense defined by Carroll and Thomas.

4.4.3 Prior knowledge and metaphors

We could not agree more when Nardi points out that "spreadsheets are not metaphorical ledger sheets. Spreadsheets are *tables*. A spreadsheets works because its tabular interface is a superb mechanism for representing and displaying the relational semantics of numerical quantities...Furthermore, a *table is not a metaphor*. It is a generic visual display..." (op. cit.). A table is not a metaphor, yet *it is based* on a metaphor: a spreadsheet is a table. This is only the beginning of the story, its progression implies the construction of a complex artifact: a blend.

A blend is produced as projections from input spaces. Sometimes these input spaces belong to a mapping which itself defines a metaphor. But this is not always the case, usually both -

considering that the input spaces are two- spaces are connected partially without defining any metaphor.

If we analyse a spreadsheet, we can consider a mental space coming from the domain of tables, or ledger tables. This mental space includes many actions that occur in the activity of producing and updating ledger tables. The second mental space, when dealing with computer artifacts, is usually that of computer commands or functions. But there is also a third mental space, that of a calculus activity. The resulting blend from these spaces produces an emergent computing artifact: the spreadsheet. This blending is the result of a cognitive activity with consecutive refinement steps, a never ending process of design as new contradictions between the artifact and the activity can derive a new elaboration of the computer artifact.

4.4.4 Visual formalisms

Finally we are presented with the concept of *visual formalism*, as it can provide a solid basis for dealing with the concerns in relation to metaphors and mental models. The definition was given by David Harel in his article 'On Visual Formalisms' (CACM, 31, pp. 514-520) and cited by B. Nardi:

> 'The intricate nature of a variety of...systems and situations can, and in our opinion should, be represented by *visual formalisms* ; visual, because they are to be generated, comprehended, and communicated by humans; and formal, because they are to be manipulated, maintained, and analyzed by computers (emphasis in original)'.

Is it possible to get a clearer example of a concept defined by a blend? The term *visual formalism* defines by itself a blend space. This blend space is connected to two input spaces: one is the space of human activity in general, in which tools and artifacts -in this case conceptual artifacts- are to have some ergonomic attributes: they are to be generated, comprehended and communicated by humans, so a visual presentation is better than an abstract one.

On the other hand, the second input space is that of computing, in which objects that are to be manipulated, analysed and maintained by computers have to be formally defined. It is in blend spaces that we construct those artifacts such as tables, graphs, maps, control panels and so on. Some of them derive from successful metaphors, other from projections of mental spaces directly related with workplaces. But the construction of these objects is the concern of computing science while cognitive science try to explain the mechanisms with which we design and construct these conceptual artifacts.

4.5 Computers as theatre

This well-known title by Brenda Laurel (1993) presents us with an interesting new metaphor in which the emphasis is put on the action rather than on actors or agents. She explains to us that "in the theatre, enactment typically occurs in a performance area called a stage...The stage is populated by one or more actors who portrays characters. They perform actions in the physical context provided by the scene and light designers. The performance is typically viewed by a group of observers called an audience" (Laurel, 1993, p. 15).

Following the mental space's unfolding, we are lead from an initial space (theatre) to a child one, the stage, in which enactment and performance occur. From the stage space we move to

the actor's space and then to the character's space. This last space is given a structure by character's performing actions in a physical context (the scene and lights) which move the focus again to the stage space. From the stage space we are presented with a new space -that of designers- only presented in order to structure the stage space (the designers design scene and lights). Finally, a new mental space, that of observers or audience, is brought to the centre of our attention.

This space unfolding aims at giving a structure to a general domain, theatre, in terms of many mental spaces. So, when Laurel gives us the abstract graphical aspect of a theatre, (Figure 22) we rapidly establish the existence of -a least- two spaces:

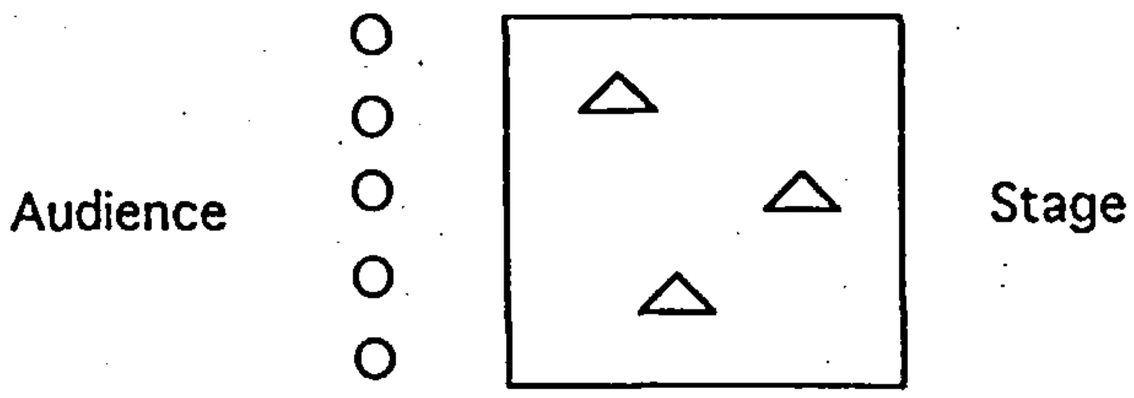


Figure 22 The main Spaces of the Theatre

This is the initial metaphor, which then is worked out as a classical blend. The observers are represented as circles and actors as triangles. Since a blend is the projection from two different spaces, in this example they are namely *audience* and *stage*. Laurel explains how the blend is produced:

> "The users of such a system, I argued [in my dissertation], are like audience members who can march up onto the stage and become various characters, altering the action by what the say and do in their roles" (Op. cit. p. 16).

This excellent example of a blend shows us how, in the new mental space produced by the blend, 'actions are altered', the representation is no longer the same as it was. The blend is told as a story, the usual way of producing blend in language. In the graphical representation (Figure 23), we *observe* the *blended* theatrical representation:

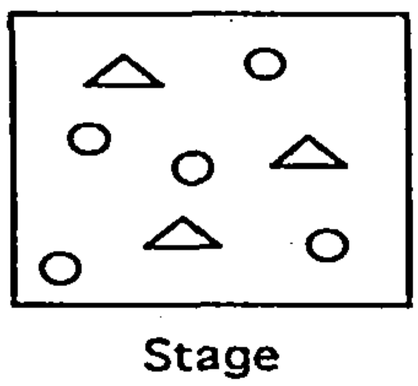


Figure 23 A Blended Theatrical Representation

If we have any doubt about the effects of a blend, Laurel continues by pointing out that "people who are participating in the representation aren't audience members anymore. It's not that the audience joins the actors on the stage; it's that they *become* actors -and the notion of 'passive' observers disappears" (Op. cit. p. 17). So, some of the concepts that have a clear meaning in one of the input spaces -audience, passive observers-, have no meaning in the blend space.

The resulting blend has -in Laurel graphics (Figure 24)- its own expression:

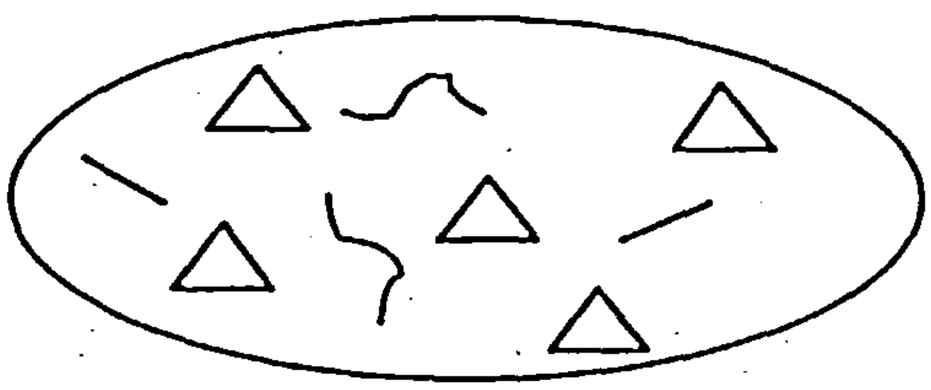


Figure 24 A Blend with Emergent Structure

It would have been difficult to find a better graphical expression of a blend, in which we have elements of the input spaces (the triangles representing actors or generic agents), but there is also an emergent structure not found in any of the input spaces (shapes and objects of the *virtual* environment). It is interesting to hear to speak, at this point, of virtual environment. It is as if the blend automatically generates virtual structures, structures that do not exist in the 'real' input spaces -knowing that spaces only exist on a intermediate cognitive level and not on a real nor metaphysical.

Laurel proposes using theatre as an interface metaphor but, mainly, in a deeper way: as a fundamental understanding of what is going on in human-computer interaction. We can add two important issues to this proposal:

- As we have shown, what we are to use is not theatre as metaphor, but a blend emerged from theatrical input spaces. From this blend, we can project structure in order to get another new blend, that synthesised from theatrical grounds (the former blend) and computer commands and icons (a second input space). The suite of mental spaces is usually more complex than the simple scheme of one metaphor projecting structure to other space.

- What Laurel proposes is the use of metaphor as a *generative* source. As Agre says (1997, p. 34), "a metaphor is 'generative' in the sense that a research community can extend its own discourse by carrying one element after another through the mapping".

When we face metaphors, we are tempted to trace their referential properties, to find the source origins and sometimes to try to invalidate the metaphor as a literal projection in which the target does not work as the source does. But the metaphor has to be understood in dialectical terms as the result of a sustained conversation with the phenomena rather than an unproblematic relation of designation between discrete terms and discrete things (Op. cit. p. 35).

4.6 Can we think without metaphor?

Another way of expressing the same question would be: is the concept independent on the metaphors for that concept? . The answer Lakoff & Johnson (1999) give to this question is a loud "No". As they point out,

> "the metaphors for love are significantly constitutive of our concept of love. Imagine a concept of love without physical force –that is, without attraction, electricity, magnetism- and without union, madness, illness, magic, nurturance, journeys, closeness, heat, or giving oneself. Take away all those metaphorical ways of conceptualizing love, and there's not a whole lot left. What's left is the mere literal skeleton: a lover, a beloved, feelings of love, and a relationship, which has an onset and an end point. Without the conventional conceptual metaphors for love, we are left with only the skeleton, bereft of the richness of the concept. If somehow everyone had been forced to speak and think about love using only the little that is literal about it, most of what has been thought and said about love over the ages would not exist. Without those conventional metaphors, it would be virtually impossible to reason or talk about love. Most of the love poetry in our tradition simply elaborates those conceptual metaphors" (op. cit. pp. 72-73).

A likely concern regarding this exposition about the need of metaphor for our reasoning is that the example used is that of love, a subjective matter for which the use of metaphor might be indispensable. The question would be to know whether the same argument holds for other domains, in particular Human Computer Interaction. We will see, in in Section 4.10, how metaphor used in the context of conceptual integration is also indispensable for conceptualising categories in the field of the computing and software engineering. Meanwhile, we will see how metaphor could be used for analysing different levels of the organisation: the organisational, the workplace and the operational levels.

4.7 Metaphor at different levels

Each of these levels –organisational, workplace and operational- subsumes the next, so that the true requirement engineering would imply the analysis of the three levels. Some authors propose the use of ethnomethodology to analyse the first level, the organisation (Jirotka and Goguen, 1994; Viller and Sommerville, 1999). At this level, it is mainly the *business* or *organisational processes* that are specified. It is important to detect the main underlying metaphors as they may be determining the solution to be chosen to design the next (workplace) level. In relation to the approach proposed by Viller and Sommerville (op. cit.), it is interesting to observe how the intention is to represent the *awareness* of people about what goes on in their environment. The aim of an ethnomethodology study would be to make people aware of their environment, and the analysis of underlying metaphors could help to make that representation more complete as we will see in the next sections.

There is –perhaps- a less direct causal relation between the workplace level and the operational level, as most of the categories employed in the cognitive level are the usual constructs of most of the modelling notation. But the entities represented by the notation would include all the underlying assumptions expressed by different metaphors. So, when the main design decisions are taken, simultaneously we are imposing a given viewpoint with all underlying metaphors that will have been crystallised in the models.

We can now see that this consideration of 'levels' implies that a spatial metaphor has been used in order to describe the concerns of HCI. It implies that depending on where we observers place ourselves, we have different *perspectives* or *points of view*. It is evident that the spatial metaphor refers to a *social place* in the complex network of social relationships, but this social place is rarely explicated with all its consequences. At the same time, the rationalist approach of a neutral observer independent of bias, with an objective point of view and without spurious interests immediately appears as questionable. The most objective point of view is considering the observer along with all the biased ideas and interests embodied in the social role being played.

4.7.1 Organisational level

At the organisational level there are at least two different perspectives which can be taken: the organisational view and activity-based view (Sachs, 1995). Adopting one or other of these perspectives leads to different visions of actors, relationships, skills, commitments and so on. The foundation of this approach is the spatial metaphor, the 'view' we adopt. The organisational view takes the point of view of management; about workers needing training, working on tasks, with procedures, using techniques and so on. The alternative point of view is of activities with people discovering problems, learning in informal conversations, creating communities of practice (Lave & Wenger, 1991) and developing knowledge and skills.

It is not the aim of this section to analyse the collection of metaphors that could build the core of the ideas of these different views. We can only quote some of the metaphors used by Lakoff (1996) in relation with work. He points out that there are two different common metaphors for work, each of which uses moral accounting: the Work Reward metaphor and the Work Exchange metaphor. The first can be stated as follows:

> • The employer is a legitimate authority
> • The employee is subject to that authority
> • Work is obedience to the employer's commands
> • Pay is the reward the employee receives for obedience to the employer...
> [the second] can be stated as follows:
> • Work is an object of value
> • The worker is the possessor of his work
> • The employer is the possessor of his money
> • Employment is the voluntary exchange of the worker's work for the employer's money.' (pp. 54-55).

Both metaphors define work from the organisational point of view. It would be interesting to include some additional metaphors regarding other aspects of each point of view. A last example will show how a given problem (the 'set-up-to-fail' syndrome) could be based on a metaphor: THE EMPLOYEE IS A CHILD (and hence that lacks knowledge and skills). One consequence of the metaphor is the representation the boss has of the situation: 'when an employee fails -or even just performs poorly- managers typically do not blame themselves. The employee doesn't understand the work...or isn't driven to succeed, can't set priorities, or won't take direction' (Manzoni & Barsoux, 1998. p. 101).

The main point about representations derived from an organisational view is the danger of failure produced by a poor vision of workplaces. Sachs (1995) describes a particular technology that was introduced in a telephone company to provide a Trouble Ticketing

System' (TTS). This system replaced an old one which 'allowed workers to talk one another. In these conversations, they compared notes about what was going on at each end of the circuit. If there was a problem, they figured out what it was and worked on it together. These trouble-shooting conversations provided the occasion for workers to understand what was actually going on in the job, diagnose the situation and remedy it' (p. 39). The organisation took the view that *employees are children*, abandoning their responsibilities and engaging others in conversation without any usefulness for the job. This view led to the apparently *objective* need and requirements for the TTS system, whose aim was to eliminate conversation. However, we would argue that such views are not independent of any observer. They are derived from metaphors embodied in a given discourse, and are suitable for some social interests, in this case the organisational point of view. At this organisational level, just detecting and modifying the underlying metaphor should help to reorganise the whole domain of needs and requirements.

4.7.2 Workplace level

Another level of HCI is the workplace level. Madsen (1994) provides a summary of the use of metaphors in system design. One of the cases extracted from a set of five included in the paper presents a design of a bank automated teller machine (ATM) taken from Maclean et al. (1991). The authors tell us that designers had personal experience of a bagel store that 'handled its lengthy queues by having an employee work along the queue, explaining the choices available and helping fill out their order on a form. The customers would hand over their forms when they reached the counter, enabling their requests to be processed more speedily' (p. 169). 'Their familiarity with the bagel store arrangement lead the designers to the innovative idea of having bank cards the customers could preprogram while waiting in line' (Madsen, 1994, p. 58).

In the case of the ATM example, it is evident that we are not comparing two different existing systems (the bagel store vs. the bank ATM) as the latter is not implemented yet. When we apply the metaphor ('the bank ATM system is a bagel store') we create two mental spaces: one corresponds to the similar elements (the *ground*) and other to dissimilar elements (the *tension*). There is a dynamic relationship between both ground and tension that allows some elements of tension to pass to the ground, giving this process its constitutive power to the metaphor. While considering that the employee working along the queue explaining the choices available belongs to the tension, there is no new conceptualisation about the ATM system. It is when we move this idea to the ground (similar elements) that the new vision of the system appears: it is possible to replicate the employee task of helping to fill the form in its equivalent way of pre-programming the bank card.

The metaphor suggests some possible generic elements in the tension mental space. One of these possible generic elements is the employee working along the queue, but the equivalent in the ATM system is not immediately identified. Sometimes the generic space, with the common, usually more abstract, structure and organisation shared by the inputs will work as an intermediate phase before arriving at the final blend space. The generic space contains what both input spaces (the bagel store and the ATM system) have as equivalent abstract structure: *the possibility of doing some task while waiting in line in order for the counter to speed up the process*. Finally, the blend space will contain the actual form of speeding up the process as a programmable bank card. The blend space is built upon structure coming from both input spaces and from other spaces as well, in particular from that of the current bank card

technologies. This last mental space is important, as it will allow the proposal of a programmable bank card to be actually evaluated and implemented.

What the metaphor has contributed to is to the *envisioning of* new functionalities and, consequently, the modification of the requirements of the system being designed. The design process is a complex one, and we have been warned (Davis, 1993) about the difficulties of completely defining the requirements before beginning the design phase. During design users uncover new possibilities which may result in additional functionalities. The role of metaphor is to anticipate what normally can be detected in the design phase, even if not all new functionalities could be anticipated by metaphorical design. On the other hand, some of the functionalities detected by a metaphorical design may not been recognised by users, as the possibilities offered by new technology are not available to them. The blend offers a wide choice of technological spaces to be taken into account.

We can conclude that metaphors are the original generative force, but we have to use blend spaces (and generic spaces) as a way of making triggering concepts workable or of elaborating these triggering concepts. The same way as general software engineering methods advocate documenting some critical design decisions through design rationale, we would advocate to include in the ontology of HCI an additional issue about metaphors used in design with a trace to all triggering concepts and the resultant blend spaces.

4.7.3 Operational level

At the organisational level metaphors can help in generating needs and requirements, even if in the Trouble Ticketing System such needs were inadequate to solve the (apparent) problem of employees engaged in time-consuming conversations. An alternative metaphor (EMPLOYEES ARE MATURE INDIVIDUALS and engage in conversation when needed to solve their problems) would have determined that trouble-shooting conversations were not a problem but the solution. At the workplace level, metaphors can help us to envision new aspects and opportunities of the systems we are designing. In this case metaphors are artifacts which produce new functions during design. But it is not metaphor on its own that gives the design solution; different blends offer different design alternatives.

At the operational level our focus is to consider the artifacts we use in constructing systems. The concepts and notations employed in systems design are artifacts for constructing new artifacts, the models themselves. In terms of Activity Theory, we differentiate between the artifact used as a means (the cognitive artifact, or concept) and the object to be built (the conceptual artifact), even if that object will subsequently be used to construct new objects (the domain artifact).

For example we describe the Entity Relationship (E-R) diagram as follows:

- Cogi.itive artifacts we use in order to build the diagrams. E-R diagrams use the cognitive artifact of an *entity* which consists of the entity concept and the associated notation (usually, a rectangle) and a *relationship* which consists of the concept of a relationship and a notation (usually a line)
- Conceptual artifacts, or conceptual models, which can be constructed using the cognitive artifacts
- Domain artifacts (models or representations in a given notation such as an entity-relationship diagram, for a given system such as a bank system) which have been built using conceptual artifacts.

The activity of constructing conceptual artifacts is based on primitive cognitive artifacts such as image-schemas. Johnson (1987) argues that the term *image-schema* (*schema* or *embodied schema*) 'reminds us that we are dealing with schematic structures that are constantly operating in our perception, bodily movement through space, and physical manipulation of objects' (p. 23). A schema is a '*recurrent pattern, shape, and regularity in, or of, these ongoing ordering activities*' (p. 29. Italics in the original). So image-schemas are artifacts that derive from our everyday elementary activities, they are produced in such activities. But as (cognitive) artifacts, we employ them in higher cognitive processes in order to conceptualize more abstract aspects of reality.

To understand the application of experientialism to the operational level, we can look at the following generic 'frame' (presented in terms of Activity Theory). This shows how the activity of producing conceptual artifacts may proceed. In such activities there are different elements involved:

- Cognitive artifacts such as image-schemas
- Cognitive processes acting on the cognitive artifacts such as metaphorical projection from image-schemas
- A subject, the community of participants in the design of cognitive artifacts
- Blends, that define the structure of new objects such as Entity objects, Relationship objects, State objects, Data-Flow objects, and so on.

We call *blends* the objects produced in this activity to focus attention on the mental spaces that are involved in the activity and on the new emergent structures that result. Let us consider the case of the Entity concept in terms of the above frame.

1) The image-schema on which the Entity concept is based is the CONTAINER. This image-schema is the source mental space. An entity is something *containing* a name and its attributes. There is an inside (its name and attributes) and an outside (other entities) which the entity may maintain relationships with.
2) The metaphor used is AN ENTITY IS A CONTAINER
3) The system designer and users of the system constitute the community of participants.
4) The blend produces the entity object by bringing together aspects from (i) the structure of the image-schema (interior, exterior, boundary), (ii) the logic of the image-schema (if container A is in B and container B is in C, then container A is in C), (iii) the mental space of geometrical forms (the visual notation associated with an entity is usually a rectangle) and (iv) the mental spaces of the domain (attributes).

4.8 A semiotic point of view

An interesting alternative point of view in relation to HCI is that of semiotics. When seen from this approach, interfaces are considered as a collection of signs, something that leads us -paradoxically- to a new and at the same time more conventional view.

Andersen points out that "the motivation for the discipline comes from the nature of computer systems: although in many respects computer systems can be conceived as tools in analogy with typewriters, pencils, brushes and filing cabinets, they differ from these tools by not primarily existing or being used as physical objects, but as signs". (Andersen, 1997, p. 1).

There are two semiotics traditions: the European tradition (F. de Saussure, L. Hjelmslev and A. J. Greimas) has worked with a dual conception of sign, considered as the integration of *signifier* and *signified* ; and an American tradition represented by C.S. Pierce and C.W. Morris which is based on *representamen* (roughly corresponding to the signifier), the *object* and the *interpretant* as two different aspects of the *signified.* In one of the earliest attempts to use Peircean semiotics in systems design, the interface was considered representamen, functionality (or type of computer system) as object, and context and values as interpretant.

In defining a prototypical computer-based sign, Andersen considers three main classes of features:

- "A *handling feature* is produced by the user and includes key-press, mouse and joystick movements that cause electrical signals to be sent to the processor. Handling features articulate user actions.

- A *permanent feature* is generated by the computer. It is a property of the sign that remains constant throughout the lifetime of a sign token, serving to identify the sign by contrasting it to other signs. Permanent features articulate systems states into parts.

- A *transient feature* is also generated by the computer but unlike permanent features, it changes as the sign token is used. It does not contrast primarily to other signs, but only internally in the same sign, symbolising the different states in which the sign referent can be. Transient features articulate system transformations." (Andersen, 1997, p. 193)

This classification is similar to what Shneiderman (1992, p. 205) considers as a portrait of direct manipulation:

- "Continuous representation of the objects and actions of interest
- Physical actions or presses of labelled buttons instead of complex syntax
- Rapid incremental reversible operations whose effect on the object of interest is immediately visible".

We agree with Laurel's proposal of removing Shneiderman's clause regarding labelled button presses, as we should consider physical actions only as key-strokes, mouse, joystick and finger movements considering current devices.

The assimilation of 'continuous representation' to 'permanent features', 'physical actions' to 'handling features' and 'incremental reversible operations' to 'transient features' is quite immediate.

As an example of handling features, the Macintosh handling system articulates a set of three distinctive parts: press mouse, release mouse and move mouse. There are also composite units built on the basic three elements: *click = press + release* which also may mean 'select', *double click = click + click*, which means 'open', and *drag = press + move + release* meaning 'moving something'.

Different computer-based signs have different handling features. A document icon, for instance, has the *select*, *open* and *drag* features while a button normally has only one, *click*, meaning in this case 'activation' or 'triggering' of some transient features.

The permanent and transient features are properties of the screen, the former consisting of patterns of pixels, the latter of changes of pixel patterns.

In the case of Mac folders, we can make a brief analysis of some of its characteristics:

Colour: grey | blue
Size: small | large
Shape: 2D folder | 3D folder

Even if we have the choice between two possibilities, both size and shape are permanent characteristics of a folder. Once selected one of them, we continue seeing the folder with the size and shape selected, both contribute to the characterisation of the folder's sign. So size and shape are permanent features of the sign.

In the case of colour, there are three possibilities:

Colour: light blue | medium blue | dark blue

and they indicate a different state of the folder: normal state (light blue), selected (medium blue) and opened (dark blue). Location is also a transient feature, as when moving the folder from one location to another we could be changing the folder or disk it belongs to. Regarding the colour there is another aspect to take into account, and it is the total number. This number is a permanent feature because it corresponds to the total number of possible states of the folder. As such, the particular shade of the colours does not matter since colours, in this sense, are only used to distinguish meaning. But once selected, each colour has a special meaning, such as we have seen in the case of Mac systems. Each colour corresponds to a transient feature.

An interesting everyday example of a sign is that of a bank note. The permanent features being size, shape and colour; none of them could be changed beyond a given limit without losing the essential characteristic of being a bank note. The handling features cover the usual actions we apply to banknotes: folding, putting into a wallet and so on; or less usual such as lighting a cigar. The most abstract features -transient- cover operations of changing the note for any object (plus possibly additional notes and coins of lower value) or for a set of notes of equivalent value. It is interesting observe that, in this sign we can detect some *virtual* properties: those objects against which the sign can be changed, as if virtuality were an inner characteristic of signs.

Andersen says that "the meaning of *icons* (which are commonly used in graphical interfaces) emerges from a likeness between expression and content, whereas this relation is arbitrary in *symbols*, and based on a convention. The reason I do not use this classification is that it is not sensitive to the characteristics of computer-based signs, namely that they can be handled and interact with." (Op. cit. p. 216. italics in original).

The classification Andersen proposes is as follows: *interactive signs*, *actor signs*, *controller signs*, *object signs*, *layout signs*, and *ghost signs*, each of them having a Petri-net description. This interesting classification could be contrasted with the (only) three types of objects proposed by object-oriented notation: *interface object*, *control object* and *entity object*. (Jacobson et al., 1995). Maybe object models might become richer with this comparison.

It could be argued -for those who like to call folders a *metaphor* - that such concept has been projected from *only* one input space, that of desktop activity. The metaphor, in any case, was the entire mapping between this input space and another one consisting of computer commands. The folder is only one of the elements of the metaphor, one of its components, not the metaphor itself. But what we eventually have in the interface is a computer-based sign (or visual formalism), the product of a blend.

The considerations about shape and colour allow us to detect other input spaces that contribute to the blend and show the complex type of analysis to be made before producing this final computer-based signs. We cannot apply the poor (as reducing and hiding complexity) concept of metaphor the way it is sometimes used. Metaphor is only a rich initial point, a choreographic view, against which we can validate our signs in order to enhance its qualities, but most of the work still has to be made in the blend space: features (handling, permanent and transient), shapes, colours, sizes, relational concepts (each of them are defined in function of the others) so common in interface design, and so on. This does not reduce the importance of metaphor, as we have already shown (Imaz, 1995) metaphors are at the basis of many *paradigm shifts*, one of them being the desktop paradigm.

4.9 The anti-Mac interface

Another interesting point of view is expressed in Gentner and Nielsen (1996). The authors say that "although the use of metaphor may ease learning for the computer novice, it can also cripple the interface with irrelevant limitations and blind the designer to new paradigms more appropriate for a computer-based application" (p. 72). They continue arguing that there are three classic problems with metaphors:

- "The target domain has features not in the source domain (e.g., telling the user that 'a word processor is like a typewriter' would not lead the user to look for the replace command)

- The source domain has features not in the target domain (a typewriter has the ability to mark up any form you receive in the mail, but a user trying to do that on current computer systems will normally fail)

- Some features exist in both domains but work very differently (the treatment of white space representing space characters, tabs, and line feeds is very different on typewriters and in word processors). Therefore, users may have trouble seeing beyond the metaphor to use the system in the ways it was intended. It is possible to design interfaces to map very closely to metaphors (e.g., Bob Mack's NOSE -no surprise editor- really did act like a typewriter in all respects), but those interfaces will often be very low-powered and suited mainly for walk-up-and-use situations". (Gentner and Nielsen, 1996, pp. 72-73).

This argument grounds on the assumption that there is no difference between the building and the scaffolding. It is evident that there are features in each of the domains that are not in the other, or when they are present in both domains they are different: it is primarily principle of any partial mapping. But what has to be analysed is not the underlying metaphor (scaffolding) which allowed the Mac interface to be designed but the blend obtained (building) as a computing artifact.

The analysis of the underlying metaphor may help in designing a good training course, provided that all differences between domains are carefully examined and specified. The first model of an atom was based on the metaphor: *the atom is a planetary system.* Although there are many important differences (why electrons -unlike planets- do not fall into the nucleus?), the metaphor was useful in understanding many facets of atoms.

This way we could say to the user that 'a word processor is like a typewriter' in a general sense, while at the same time giving him/her the detailed differences because the metaphor is only a starting point, the final being the current computer artifact that is similar and also different to the metaphor. So, the training course is also a process in which we can use the same metaphor used in designing the artifact, but the process of training has to be continued to a final goal in which the real artifact is presented with all its particularities.

Some features, like the use of the backspace key in order to delete the last character typed, become so rapidly clear that it is not worth explaining the difference with a typewriter. A brief list with the main differences could be enough in a first approach to a word processor, making experience -an important component of training, not to be forgotten for the rest of the job.

An interesting opposite argument is provided by K. Nakakoji (1994):

> "The introduction of software to a new culture brings to light many hidden and unpredictable factors. A system's functionality is often unconsciously affected by underlying traditions of the culture in which the system is designed. For example, many of the fundamental concepts used in Western-language word processors -such notions as cursors, tabs, and margins- stem from the typewriter culture. The typical user of a Japanese word processor has had little exposure to the typewriter... Given this background, Japanese users have had a considerable amount of trouble understanding the concepts of cursor movement embodied in a typical word processor application" (p. 108).

I suppose the argument is clear enough in order to recognise the importance of the underlying typewriter metaphor used in the construction of word processors. It is interesting to observe that new versions of Microsoft Word (Office 2001 for Mac) allow beginning to write wherever you want in the document without needing to use tab icons. Old versions of Word are based on the typewriter, while new versions begin to give up some features of the underlying metaphor, reworking the initial conceptual integration.

The authors say that "we need to develop new interface paradigms based on the structure of computer systems and the tasks users really have to perform, rather than paradigms that enshrine outmoded technology". That is true, we have to use new paradigms, but they are to be faced up to older ones. The history of technology shows us -for instance- how the first models of cars had and interface directly inspired in horse coaches. This is the usual model of gradual evolutions in our cognitive processes. It is not possible to jump through higher levels without staying for a while in each of them.

In order to conceive new paradigms, we have to think carefully of the current paradigms, taking into account that the old model was cutting-edge technology some years ago.

Gentner and Nielsen's proposal about the anti-Mac interface involves the following basic principles:

- The central role of language
- A richer internal representation of objects
- A more expressive interface
- Expert users
- Shared control

4.9.1 The central role of language

It is argued that there is an important property that is missing in graphical interfaces: the ability to group complex collections of objects or actions and refer to them with a single name. Lakoff (1987) has demonstrated the capacity of metaphors in order to create new language categories. Indeed the title of Lakoff's book *Women, Fire and Dangerous Things* is chosen because these seemingly diverse things are classified as belonging to the same category by the Australian aboriginal language, Dyirbal. The equivalent of projection between domains we observe in metaphors is feasible using links or just creating new objects that integrate those to be grouped. Links (or equivalent) are the usual tools to build semantic frames that allow movement along different semantic paths.

As a natural language interface will remain unavailable for sometime yet, the authors propose a reference librarian (a type of agent) with which to negotiate the actual queries.

4.9.2 A richer internal representation of objects

A richer representation of documents means the inclusion, for example, of "its authors, topic matter, keywords and importance; whether there are other copies; what other documents it is related to; and so forth". (Op. cit., p. 79) Most of this information only requires techniques of full-text search in order to be extracted from the document. An agent would detect the fact that two or more documents are used together, thus creating a hypertext link between them to allow easier group access.

4.9.3 A more expressive interface

"The richer internal representation of objects will allow more intelligent interpretation of user commands, but it will also be reflected in a more expressive interface with a richer external representation" (Op. cit. p. 79). This point reflects the fact that some concepts are relational (each of them are defined in function of others), expressiveness of interfaces implying a richer internal representation (or more semantics) of objects. In terms of permanent and transient features, more expressiveness would imply increasing the number of permanent and transient features. As permanent features are usually expressed as size and form, and these in turn require more pixels, it is natural that the authors address the issue of screen size as a way of enhancing expressiveness.

4.9.4 Expert users

In short, this issue tries to analyse the difference between users that have grown up with computers and the current generation of users. The formers are users that will demand expressive interfaces with richer semantics.

4.9.5 Shared control

The central point here is the move to a decentralised control. In place of the individual user who controls all the tasks being executed, a mixed locus of control with both humans and computer-based agents is postulated as the new paradigm. This shared control will be the outcome of some protocol, as "in the real world, we don't want anyone rearranging the mess on our desks, but we don't mind if someone puts a Post-it note on our computer screen, empties the waste-basket overnight, or refills the newspaper stand with the latest edition" (op. cit. p. 80).

In the first part of this paper we observe a criticism of the concept of metaphor and Mac interface, as if the latter were the result of only applying a metaphor. In the second part there is a proposal of a new paradigm interface, which is described in terms of 5 main points.

As we have seen, points 2) and 3) are relational concepts, the 3) being a consequence of richer representations of computer-based signs. Point 4) establishes the causes for richer representations. Both 1) and 5) could be integrated as derived from the presence of computer-based agents, allowing -on the one hand- an intelligent negotiation of meaning in the case of using of natural-like language, and on the other a shared control of computer activities.

What makes apparently different both parts (Mac and anti-Mac interfaces) is that in the former, the argument is based on the metaphor underlying the interface; in the latter the new paradigm is defined in terms of the blend space. This way the underlying metaphor used to define the anti-Mac interface remains hidden.

Another significant point is that nothing is said about the permanence of some current interface's features: it is likely that some direct manipulation features will remain, as it would be difficult to imagine how an agent could write or draw on the user's behalf. Agents will be able to do some writing or drawing operations, but most of the tasks are likely to be done by human agents. Then, we should think of future interfaces as built upon older features, mainly direct manipulation for specialised tasks, even if some important new features are added to existing ones that are not based on direct manipulation.

As clearly stated by Maes,

> "the whole metaphor of direct manipulation, of viewing software as a tool that the user manipulates, was invented 25 years ago when the personal computer was first emerging and when the situation for the user was completely different...I think we need a new metaphor. The one we are proposing is that of software agents, software that is personalised, that knows the user, know what the user's interests, habits, and goals are. Software that takes an active role in helping the user with those goals and interests, making suggestions, acting on the user's behalf, performing tasks it thinks will be useful to support the user". (Op. cit., p. 11)

So we see that it is almost impossible to get metaphors out of our sight. Agents *know* the user's interests, they perform tasks they *think* will be useful; and so forth. *Agents are smart people.* The final metaphor implies also the concept of 'indirect management', based also in current companies structure, concepts derived from groupware and decentralised teams. We guess whether the new metaphor could be expressed as *The interface is an indirect management through human and computer agents.* The metaphor allows us to think productively of the new paradigm scenarios, as we think of details of the input mental space, our source of our new ideas. These ideas have to be transformed in the blend space, determined by other input spaces, one of them being the limitations of current technologies.

4.10 Metaphors and Figurative Language

We can sum up some answer to the arguments used to demonstrate that metaphors either are dangerous, inadequate or problematic.

Metaphor, per se, is not the source of any pathological symptom. Employed in the context of a more general framework (ideology or discourse) it could have perverse consequences, but in such a case metaphor is only one medium used by the ideological or repressive discourse that usually employs more expedite methods than metaphor.

Regarding the argument about the users believing that when they move a document icon from one folder to another, they are really moving the document itself, we can say that we use many devices without knowing exactly what happens in the black box. The metaphor has to be judged from its convenience and fitness for the current task, even if the user's mental model does not agree with the designer's mental model. The fact that the user thinks he/she is moving a document from one folder to another is not only irrelevant, it could even be beneficial for his/her performance in using the graphical interface.

In relation to the argument that the target domain of metaphor may have features not in the source domain -or the source domain has features not in the target or some features exist in both domains but work very differently- what is argued is exactly what happens in conceptual integration. "The operation of conceptual integration can recruit from many domains, and it can develop elaborate mappings and projections. It is not algorithmic or deterministic, but it is guided by optimality principles sensitive to purpose and situation" Turner (1998). So, it is the very essence of conceptual integration rather that a limitation of the metaphor, which -in turn- is one of the various input domains.

An extension of the previous reasoning considers that metaphors are useful for some purposes, but inadequate to serve as the basis for interfaces of any complexity, which must express their own rich semantics. What Harel calls a visual formalism is, in fact, a *conceptual integration* with all its complexities derived from the purpose and context. The desktop metaphor is a poor cognitive artifact as to be used directly as a computer-based sign; in fact, it is a blend built upon such a metaphor that is a richer artifact: it could be given a structure as complex as we need for the task to be performed. The important issue to be taken into account is that the conceptual integration was built using a metaphor as one of the source spaces. The metaphor is the basis upon which we construct the visual formalism, it allows a conceptual integration to be made as metaphor is the source of conceptual domains.

The last argument makes the usual mistake of assimilating the metaphor with the visual formalism (or sign) designed on the metaphor. Yes, there is a problem with employing the trash can to eject floppy disks, but it is not a consequence of using metaphors in designing interfaces. The problem derives of a poor –inconsistent- design decision when using the conceptual integration. As we will see later in Chapter 5, if we have applied one of the principles that lead the conceptual integration -the topology principle-, we would have designed an independent icon to represent the floppy disk driver. So, when dragging a floppy disk on that icon, it would has been transformed to represent the ejection of the floppy disk.

There is general trend in the computer science community to refuse any type of *informal* reasoning. Such as Goguen (1993b) notes, "some computer scientists (e.g., Edsgar Dijkstra) feel that any use of diagrammatic or verbal reasoning is bad, because it enables, and even encourages, sloppy intuitive thinking, as opposed to precise, linear, logical reasoning".

Goguen argues, on the one hand, that such views may have significantly impeded acceptance by a larger community of much of the work done in formal methods. I will argue, on the other, that much of the work done by the computer community is not only informal and *intuitive*, but even commonly uses *figurative language*. This would not be a singular affirmation if we do not try, simultaneously, to show that figurative language is a way of reasoning as valid as one that could be considered as *literal language*. Moreover, we will show how the start-up of some new concepts is invariably occurring as a metaphor or analogy.

In the last few years there have been a collection of works about the relationship between literal and figurative language. One of the most exciting disclosures is that between both types of languages there is a continuum, a gradient; "distinctions between the types are in fact graded, and judgements of literal versus figurative are accordingly graded". (Turner, 1998, p. 69).

Let us see first the types of conceptual integration in order to get a basic background, which could be used later on in our discussion on certain computing concepts.

We have seen in Chapter 1 that a blend implies two input spaces, a generic space and the blend itself. Conceptual integration is a basic cognitive operation that operates on two input mental spaces to yield a third space, the *blend*. So we can distinguish two different elements: i) a process or cognitive operation, and ii) the product of such process, the conceptual integration or blend. Sometimes, we will use the expression conceptual integration to refer either the process or the final product, though the usual meaning refers to the product of the process.

Another concept used in relation to conceptual integration is that of *integration network*. This last concept is a reference to the set of spaces involved in the conceptual integration, and there is a taxonomy that describes the type of integration networks.

4.10.1 Types of integration networks

A first type is the frame network, in which all spaces –inputs, generic and blend- share a topology provided by an organising frame. For example, if we compare two different regattas performed at different times in order to see whether one of them perform better than the other, we create four spaces: both input spaces of regattas, a blend space and a generalised space.

In this example all four spaces have the organising frame *boat making an ocean voyage*, and the blend has an extension of that frame: *two boats making ocean voyages and, moreover, racing as they make them.* This type of structure in which all spaces share the same topology of a generic space is called *shared topology network.*

The analysis of integration networks identifies *one-sided* and *two-sided* shared topology networks where either only one of the frames or some topology from both frames is projected to organise the blend.

Such as Turner points out, we can ask what the implications of the network model are for the literal versus figurative distinction. The conclusion is that this network model generalises an earlier claim that the same conceptual and linguistic operations underlie "figurative" and "literal" examples.

4.10.2 *Literal* integration networks

The example of a single-framing network such as "John is the father of James" has two inputs with the following relative status: one is a familiar abstract frame, while the other is a relatively specific situation with no competing frame. The familiar abstract frame is routinely applied to the conceptual domain (individual human beings) upon which the specific situation is built. "This type of integration network usually seems highly literal" (Turner, op. cit., p. 68).

Our HCI methods offer us many examples of a single framing network. If we say, for example, "Name is an attribute of entity Employee", there are also two input spaces with the following relative status: one is an abstract frame corresponding to the concept of entity (with an entity name, attributes, relationships with other entities and son on). Both Name and Employee are instances of the type name with no competing frame. So the abstract frame of the concept entity is applied to the conceptual domain (instances of type name) upon which the specific situation is built.

Once a given conceptual domain has been built, we apply specific situations to it (instances of elements that make up the conceptual domain) and at the same time as the domain is more and more used, the language constructions used to refer to such situations become increasingly literal (entrenched).

4.10.3 *Figurative* integration networks

> "By contrast, if two input come from apparently widely different specific conceptual domains, the result is a different type of integration network, namely, a shared topology network, whether one-sided or two-sided. The structure that applies to both of them (i.e. the generic space) is typically highly abstract relative to both of the inputs. Such a case is commonly thought to be figurative...In particular, highly two-sided shared topology networks...are typically judged to be highly figurative" (op. cit., pp. 68-69).

What we can say regarding literal vs. figurative language is that there is not such a clear opposition between both terms. Rather, it is a question of grades and, mainly, of degrees of entrenchment as we will see later on section 4.13. As Gibbs points out

"yet the search for a theory of what is literal about language and thought has not provided any clear answers to the question of what it means to say that we speak and think literally. Simply put, there exists no comprehensive account of literal meaning" (1994, p. 78).

4.11 Some basic examples

We can try to analyse some of the basic concepts used in computing that appear to us as highly literal: instruction, sentence, procedure, and abortion. The analysis aims at showing to which extent they are –in the origins- figurative, even if currently –as a result of entrenchment- they appear to us as highly literal.

4.11.1 Instruction

Let us imagine that we are in front of the first electric computer (ENIAC perhaps), which produces a group of electrical pulses resulting in a change of state or producing some given effect. This initial event is crucial, as there are no new meanings yet, only new things that are named using the words available at the moment and not describing completely the emergent situation.

So all such new things are to be named, and in order to produce the first blends we have to select the input spaces from which to project the main structure. It is very likely the meaning from which the word *instruction* has been selected is that of a detailed direction or procedure. That means a reference to a *judicial* frame, in which we usually assist to detailed directions on procedures.

An alternative meaning points to authoritative direction to be obeyed, that is, an order. So, from the beginning of the computing era there has been a special agency power in charge of the computer. Whether the authoritative direction to be obeyed was given to the computer or the computer gives this instruction to itself, the resulting projection is the same.

As an alternate name used for *instruction* is *command*; analysing the meanings of this last word we have a confirmation of the intentional feature transferred to the machine.

4.11.2 Sentence

In the case of a sentence -even if the group of *instruction*, *procedure*, *sentence* and *execution* could be wholly referred to the judicial domain- it is evident that it has been produced using an input domain from linguistics. At a given moment in the development of computer science the concept of *symbolic language* appears and logically the unit of any language is the sentence.

4.11.3 Procedure

One of the meanings of procedure points out to a series of steps taken to accomplish an end. This sense could be considered as a generic space while a second meaning, that of the *methods of conducting the affairs of an organisation* is a more specific frame from which the generic space may be projected. In this case, we observe that the generic space –steps taken to accomplish an end- is a concrete form of the SOURCE-PATH-GOAL image-schema, in which

some intermediate sub/goals have to be reached. So the concept of *procedure* is guided mainly by pressure from the SOURCE-PATH-GOAL image-schema.

4.11.4 Abort

This computing concept receives its sense from terminating an operation or procedure before completion. But this is also -in its origins- a figurative meaning derived from a generic space in which the fact of *giving birth before the embryo is capable of surviving* or *ceasing growth before full development* contributes to its generic structure. As usual, a series of metaphorical projections deploy a collection of meanings to the same word.

That is why the dictionary includes the following meaning: *A procedure to terminate execution of a program when an unrecoverable error or malfunction occurs.*

4.12 Ways of Conceptualisation

As new concepts emerge, they may seem figurative at first glance. After a while, they become more and more literal, as some of the following examples show: things like *layered architecture* or *sandbox* –which may appear as highly figurative to a beginner software engineer- are concepts with a *real, concrete* meaning for experienced ones. This evolution along a particular history is described by Turner, who has taken from another author (William C. Wimsatt) the concept of *generative entrenchment* inspired in the general opposition between innate-acquired.

Wimsatt (1986) says that he will speak of

> "features as being 'generatively entrenched' in proportions to the degree that they have a number of later developing traits depending on them" (citation taken from (Turner, 1991)).

The same way is possible to state that concepts and their connections have degrees of *generative entrenchment* where this last expression means the extent to which the rest of the conceptual system depends upon or is invested in that connection.

The classical theory (folk theory) of categorisation lead us to think of entities in the world as composed of features like red or small or hot. This same folk theory is continuously present in our software engineering methods such as entity-relationship diagrams or class diagrams of object-oriented technology. In such models, the basic class hierarchy is the equivalent of the categories of cognitive sciences and are defined by the features possessed by all the members of the category (or class). These features are inherited as we move down in the hierarchy, so the features of higher level categories from which a given category derives will receive all accumulated features in a top-down way.

Another aspect of categories is that we tend to think of them as fixed, invariable, and of a connection as true when it is part of the structure of these immutable categories.

We categorise to operate efficiently in the world: evolutionarily there is no genetic pay-off in perceiving difference except where it increases fitness. So we perceive things as equal at a given level of conceptualisation. The question would be: how do we decide more or less automatically to try to take a particular equation as expressing analogical connection rather than category connection?

Turner answers that how we approach the equation depends on where the two equated concepts are located within our category structures. There are many degrees of generative entrenchment between conceptual connections. Some of them are so deeply entrenched in our conceptual systems that we cannot imagine what it would be like to operate without them. As Turner says, "they have a high degree of cognitive indispensability" (Turner, 1991).

Other connections are less deeply generatively entrenched, as when we understand events not caused by animate agents in terms of actions performed by animate agents in expressions like "Death took him".

This continuous gradient of decreasing levels of entrenchment leads us to other expressions like "I hunger for your love" with a slightly less generative entrenchment.

The conclusion is that abandoning the dichotomy of literal versus figurative would dispel many problems in the theory of meaning, by unasking the questions that produced these problems (Turner, 1991, p. 141-142). Following his ideas, he points out that "to recognize an analogy requires us to bring to bear our detailed knowledge of category structure. To recognize analogy, we must know in detail where the equated concepts stand within our entrenched category structures, and how they are related to each other in those category structures, and at what level of categorization those relationships hold. In short, analogies exist by virtue of category structures; categories and analogies inhabit the same conceptual space. A statement is an analogy not with respect to the world but with respect to the category structure that is brought to bear upon it" (op. cit. pp. 147-148).

4.13 The Layered Architecture

A frequently used concept in software design is that of a *layered architecture*:

> "the *Layers* architectural pattern helps to structure applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction (Buschmann et al, 1996, p. 31).

In this case, the input space that projects its structure to the blend will be that of an architectural structure of layers. That means that each layer is based upon a lower level. Moreover we observe that there is basically a spatial metaphor ("software is a spatial structure"), a blend (the layered structure of the software) and a generic space in which there is a set of ordered elements, each element being supported by a previous one.

The connection between both input spaces (architecture and software) is quite deeply and generatively entrenched. Along the years the computing community has been using other similar connections as software *and* engineering, even if this last connection was established in order to project some characteristics of engineering processes to the software process. In the case of architecture *and* software the connection is stated for projecting some *design solutions* from the former to the latter, considering that an abstract architecture means a structure with some functional relationships added.

As the use of this type of conceptualisation generalises, it is the connections between the generic space and the software space that is directly established. The generic space of 'architectural structures' becomes a new conceptual domain of its own. So it is the candidate to

be one of the input spaces to a specific construction as when we say "This is a three layer architecture with a first layer that contains..."

4.13.1 Example of a Layered Architecture

Let us consider an example of a Java application, using the concept of package (UML) as a tool for grouping a collection of objects and with dependency relations between packages.

This blend is the result of various input spaces: the input space of a layered structure, an application software space (for executing Java sentences, that is opcodes instructions), one space of Java packages (which constitute the Java API) and the application software space developed in Java (the three application packages).

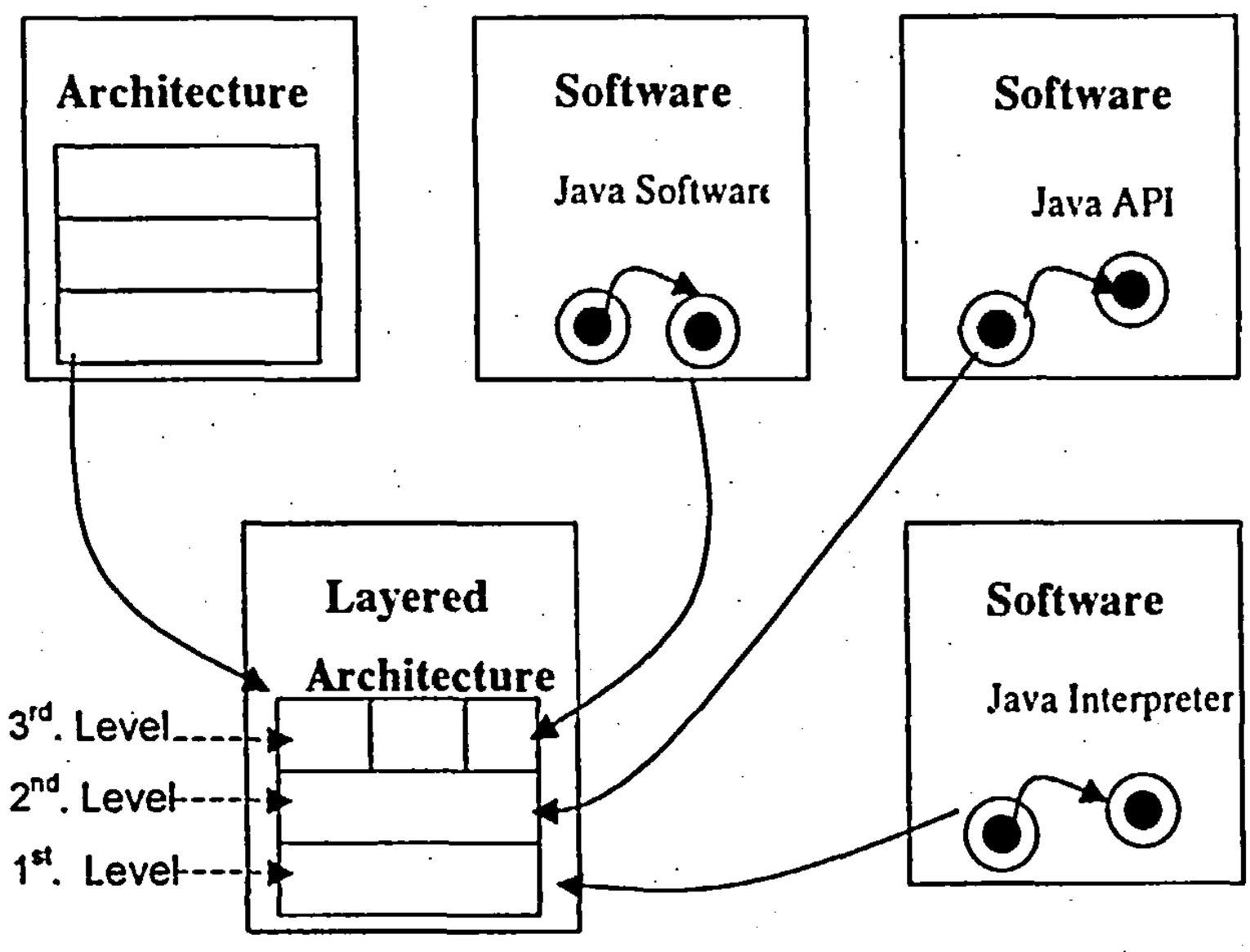


Figure 25 Domains of the Example's Conceptual Integration

From the frame of the first space, there is a projection of the layered structure, which functions as a group of slots that are to be occupied by elements from the other input spaces. In fact, each of the other input spaces will transfer the Java Virtual Machine software, the Java API software and the Java Application software.

Depending on the input spaces, the meaning of the relationship between two successive layers will change. But the important issue is that the blend gives coherence and uniformity to the structure independently of the input spaces. This uniformity is a consequence of the general dependency relationship between all the input spaces. In this sense, the blend allows a higher degree of categorisation to a group of differently related chunks of software.

In Figure 25 we observe the dependencies of each package of the third layer from the second layer of the layered architecture. In this case, this dependency means "objects of the package are built using elements of the second layer (the API)".

In the case of second and first layer relationships, we can see that the dependency means "The elements of the API are to be executed through the Java Virtual Machine". So there is a categorisation of *dependency* by which it can mean differently: a construction or an execution.

The third layer is divided into three (one for each package) tiers in order to establish an ordered relationship in a given layer. The dependency between tiers means "elements of the first tier uses (or call) elements of second one".

The fact of categorising different relationships using only one concept allows having a uniform structure, such as the layered architecture. This has the advantage of smoothly categorising things that would have been described differently, requiring a more detailed explanation with inclusion of unnecessary complexity.

4.14 The Broker

Another example of conceptualisation through a blend is that of Brokers. A broker is someone that acts as an agent for others, as in negotiating contracts, purchases, or sales in return for a fee or commission.

As this concept has been used to define "Services Request Broker" (SRB) and "Object Request Broker" (ORB), we would try to give a detailed description of the frame that such mental space projects into the blend.

The definition speaks of somebody that act as an agent for others, that is someone that can perform some type of negotiation and come back with the results. A broker may interact with a different organisation in order to perform different kinds of transactions (contracts, sales, and purchases). It is a relevant feature the fact that a broker could use different styles (or languages) as required by the context of the transaction. The broker adapts himself to the client that requests a service and may speak a language that the client can understand. Because his position as an intermediary –the definition speaks of "one that is in a *middle* position or state"- he may serve simultaneously each person (or company) asking for his services.

This frame has enough general structure as to organise the software necessary to intermediate between a collection of pieces of software (clients) asking for services (functions to be performed in order to get a given result) from other pieces of software. We intentionally write a *piece of software* and the name derived from the context to recall that all additional names (*client*, *server* or *service*) given to pieces of software are derived from the context, the intentionality and the blend built with the input spaces.

We can build a representation of a broker in the input mental space and a different one in the blend to show that the general organisation is applied to the specific context of software. It is possible that the representation used here for a broker may be influenced by the usual representation of a software broker or CORBA (Common Object Request Broker Architecture, (OMG, 1992)).

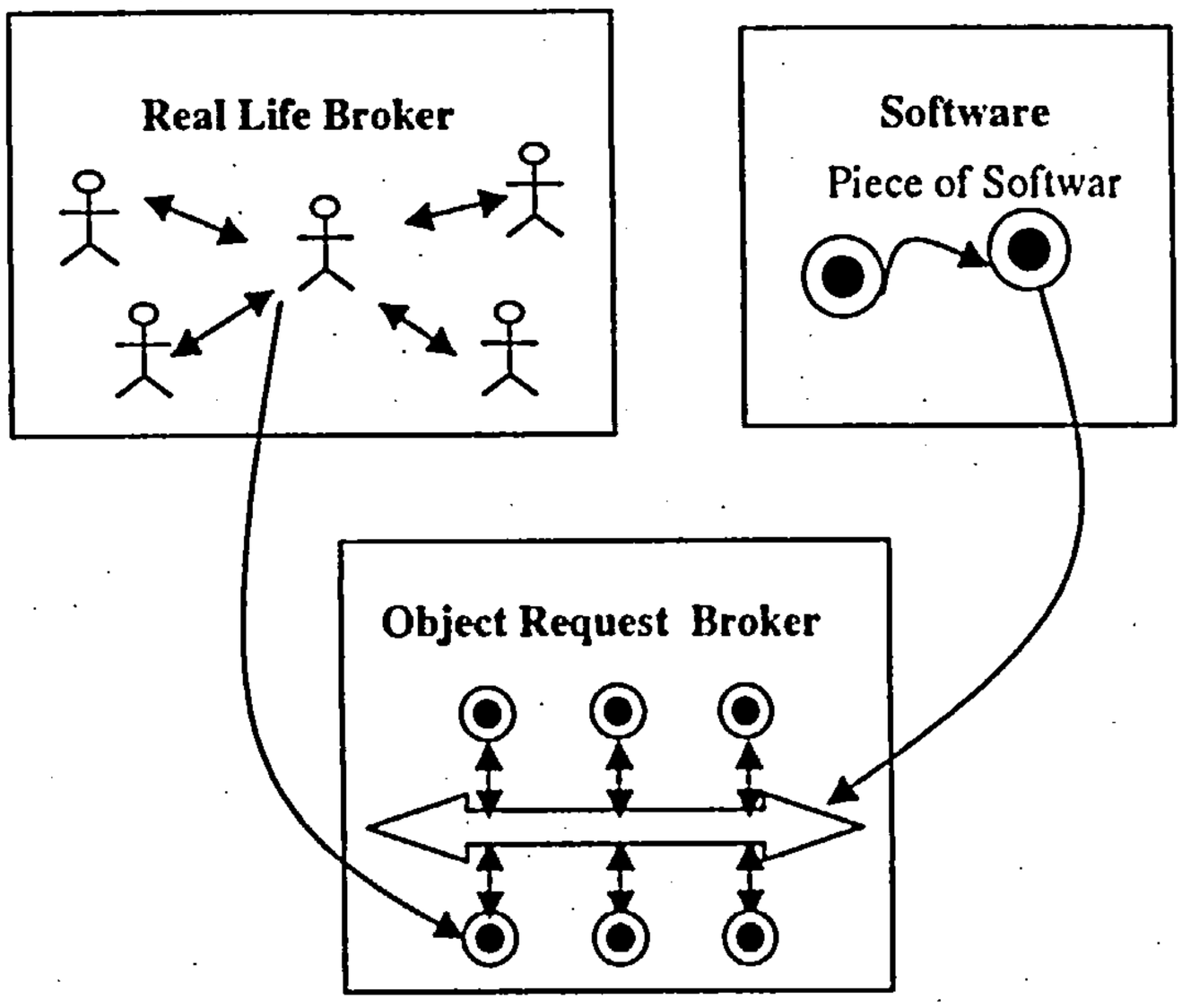


Figure 26 The Broker Conceptual Integration

In the context of a Service (or Object) Request Broker, such specialised software is also known as *middleware*, which may be considered another spatial metaphor (because it is in a middle layer, between applications and operating systems) or projection from the blend of layered architecture.

Figure 26 shows that what would have been a direct relationship between two objects in a piece of software, when the role of an intermediate broker is added, the two objects must communicate through the broker. This way, all possible differences of language, style or whatever it may be, the intermediate software will solve the conflict. Such as with real brokers, the new middleware has to provide a list of different types of services.

This is the description of the blend, in which the description of a real world broker is detectable. The frame of this last source mental space projects much structure, so with only one word (broker), all the frame is automatically transferred giving an immediate organisation to the description of a group of applications that otherwise would require extensive commentary to give it a meaning.

We know that when both input spaces come from apparently widely different specific conceptual domains, the result is typically judged to be highly figurative. As this concept of Object Request Broker has a relatively long life of entrenchment, we are not aware of that feature of being, in principle, *highly figurative*. From the point of view of our everyday language of computer developer this expression is, on the contrary, highly literal. Moreover, when a new concept appears in the field of commercial software development, we try automatically to understand the meaning of apparently foreign concepts without even noticing the fact that it is a blend, which would be taken as highly figurative.

Let us see another example of this highly figurative language used in software development: the Java sandbox feature.

4.15 The Java Sandbox

We have already seen two examples, one taking structure from the domain of architecture and other from the domain of commercial brokers or stockbrokers. Now we have an example of blend importing structure from the domain of children playing in a box of sand.

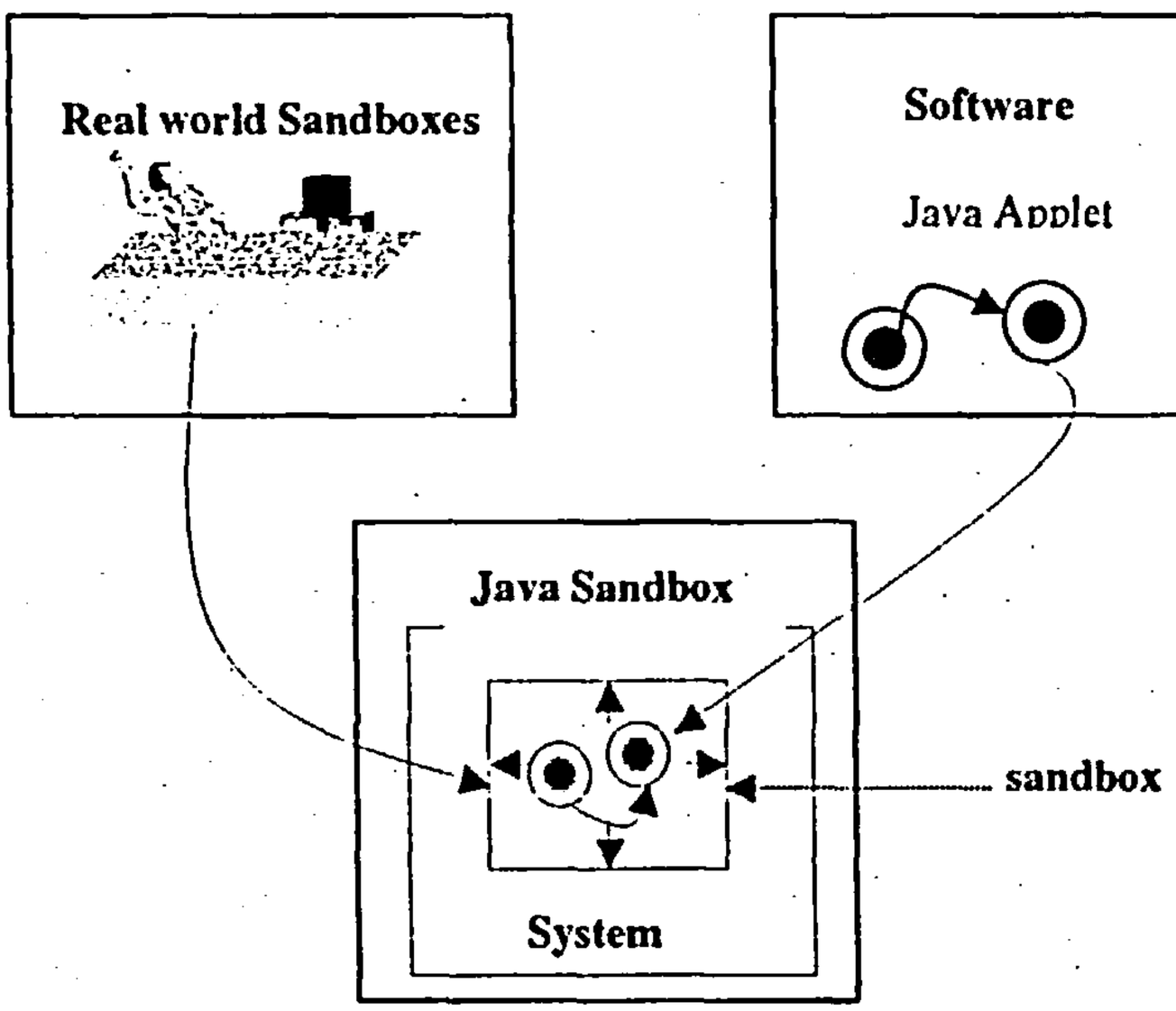


Figure 27 The Sandbox Conceptual Integration

The source input activity is that of playing without dangerous effects or without going to a lot of trouble as a consequence of such a gambling. Such a goal is got by means of a box (a container with limits), so the consequences of the children's actions are limited to the extent of the box's sides. The potentially annoying children's actions are associated –in the blend- to the consequences of executing an imported applet.

> "The sandbox: Java's security allows a user to import and run applets from the Web or an intranet without undue risk to the user's machine. The applet's actions are restricted to its 'sandbox', an area of the Web browser dedicated to that applet. The applet may do anything it wants within its sandbox, but cannot read or alter any data outside its sandbox. The sandbox model is to run untrusted code in a trusted environment so that if a user accidentally imports a hostile applet, that applet cannot damage the local machine". (Fritzinger and Mueller, 1996).

What has been achieved with the blend is to give a coherence to a set of characteristics of the Java language in other to guarantee the security of environments where downloaded applets are executed from the Web.

Once again, the blend would appear as highly figurative, but the fact of using more and more figurative language in the domain of software determines an acceleration of the process of generative entrenchment. The domain of software development (commercial products and definition of standards) determines an increasing use of figurative language as a literal one. There is no even an intermediate period when such new expressions are considered as figurative, they are immediately associated with the 'objective' world of software and so considered as literal.

4.16 Application to the Flex Project.

The description of HIC includes a spatial metaphor (*lean back* and *lean forward*) to indicate a home –or entertainment- orientation (coming back home) or a workplace –or work-orientation (going to work), but even if this metaphor is not essential for the blend, it is nevertheless very useful for conceptualising the future device, as we will see later in this section. The spatial metaphor –the device is not placed in *a fixed position*, neither *back* nor *forward*-, suggests that there are two equivalent solutions:

- a moving around device
- people move along the house but can access a *distributed* device at different locations

The choice of one of both solutions is likely related to cost considerations, not to technological issues.

We can observe that the way used to conceptualise the new device is by means of a metaphor: it is a physical object that is neither *leaned back* nor *leaned forward*. This is a description that might apparently have been made saying that the new device is neither a TV set nor a PC. An interesting question would be why we prefer to speak of leaned back or leaned forward objects in place of more concrete things like a TV set or a PC.

This is an interesting issue since fact that a *figurative language* may be more specific (or precise) than a *literal language*. If we employ concepts like TV sets or PCs we focus on current functionalities of such devices, but leaned back or leaned forward devices means devices that are oriented towards entertainment or workplace activities, with current or future functionalities or some equivalent –future- devices used with the same intention. The spectre of devices that may be placed in a leaned forward position is larger than actual PCs, hence the concept is more adequate because it is more precise and -paradoxically- broader.

The role played by figurative language is similar to the role played by variables in mathematics. To say that one is speaking of a *leaned forward* device is similar to say that we are speaking of devices *X*, where X is indicates the set of computer-based devices used in the workplace to help us with most of our everyday job activities. The difference is that while using X as a variable we have to indicate the properties of elements that belong to a given set, when using "*leaned forward*" we are declaring the properties at the same time we are defining the set of devices.

There have been some concepts included -usually employed in the software domain- that are figurative language:

Information space
Dimensions
Navigation (via intuitive multimodal interfaces)
Inter-person awareness
Datamining
Search engine
Web agent

Some of these features of an *infotainment* device are already included in current TV devices (video text) and the differences between information and entertainment is sometimes difficult to acertain.

The metaphor of data *mining* is an interesting one because it is in line with our proposal of thinking of requirements as *fabrications*. Data mining offers another figurative way of thinking of data –and especially of information- as something to be *elaborated*, to be *extracted* from a mix of valuable and uninteresting material. The first opposition between data and information suggests that the later is an elaboration of the former. Data is the individual invoices generated from all the sales produced during a month. A first elaboration is the total amount of sales produced in the month which is much more significant information that could be compared to sales of previous months and so on.

The difference between *searching* and *data mining* might be the degree of difficulties in finding what is a significant information for our purpose. We may *search* for concrete occurrences of some type of data. In data *mining*, the final information may not be as concrete as when we are searching information. While mining for something that might be interesting, we may change our goal many times in function of previous findings. Data mining is also associated with *fuzzy searching*, as one can start searching something that is not clearly defined from the beginning.

The process of data mining brings structure from the mental space of sifting, filtering, and separating some special nuggets from the rest of the ground. The difference is that in real mining, we know in advance what we are searching for, while in data mining this might not be the usual case.

Chapter 5

Conceptual Integration

5.1 Figurative Language is frequently a form of Conceptual Integration.

The examples we have presented so far are only a minimal list of the use of figurative language in Human-Computer Interaction. The intention was to show that what is known as *figurative* or *intuitive* reasoning is more the rule than the exception. This type of intuitive concepts is very useful in our everyday activity of software development, even if it could be shown that the same is valid in all human activities in general.

The list of metaphors is uncountable, the following is only a sample of metaphors:

Construction: architecture, foundation, platform, frame, framework, hook, front-end,

Manufacturing: Pipeline, tool, toolkit, toolbox, artifact, interface, process, package

Business/Organisation: client, agent, broker, procedure, server, export, import, contract

Office: Drawers, files, folders, papers, paper clips, stick-on note sheets, attachments

Social activity: Queue, pool, session, checkpoint, police, control, grant, authorize, utility, garbage collection, host, protocol

Human values: trust,

Biology: taxonomy, inheritance, tree, branches, leaf, web, virus,

Deck of cards: card, piles,

Documents: Books, chapters, bookmarks, figures, newspapers, sections, magazines, articles, newsletters, forms, bind,

Geography/Urbanism: Domain, landmark, pathway, place, site, location, region, realm,

Travelling/Sports: navigation, surfing,

Most of pattern names are also highly figurative: *Factory*, *Bridge*, *Decorator*, *Façade*, *Proxy*, *Chain of Responsibility*, *Interpreter*, *Mediator*, *Observer*, *Strategy* and so on. As Gamma et al. (1995) points out, finding good names has been one of the hardest parts of developing the catalogue of patterns. It is not a coincidence that good names –in the context of pattern discussions- are, at the same time, highly figurative. The description of the problem, the solution, the consequences, the results and trade-offs of applying the pattern constitute the *meaning* of the named pattern.

5.1.1 Complex blends based on multiple metaphors

When Johnson (1994) tries to explain the metaphoric word *application* (*to apply: to use something for a given purpose*) as "a set of instructions stored in memory", he is using three words with metaphorical origin: instruction, store and memory. But the real interesting observation that the author makes is about some interesting examples of mixing metaphors for a given concept, as we have already illustrated for requirements.

He points out that "highly abstract things are made solid in metaphor, and metaphors are often used in inconsistent way", showing simultaneously the use of a metaphor where abstract things (concepts) are made solid (objects). This is a key issue when treating the problem of using metaphor in computing science: the inconsistency.

The question would be whether this inconsistency is a consequence of informal conversation or discourse or whether such an issue is illustrating some deeper symptom, the symptom of a difficulty not easily solved in terms of a usual conceptualisation in the field of computing.

The first example cited by Johnson explains that

"IBM plans to announce its long awaited repository, or central data dictionary, for Systems Application Architecture (SAA) by early 1989 at the latest, according to industry consultants. IBM expects the repository to emerge as a storage facility for vital information across operating systems. But in its first implementation, the dictionary will act largely as an underlying foundation for a computer-aided software engineering (CASE) systems..." (Margolis, 1989).

Johnson adds that

"a repository in the familiar world we live in is a container or a place where we place physical objects. A repository, we might be surprised to learn from this passage, is a 'dictionary'. It is also, not surprisingly, a 'storage facility' and finally a 'foundation'" (Johnson, 1989, p. 100).

The second example includes the following passage:

"The press conference held here last week was to showcase HP's Precision Architecture, a reduced instruction set computing (RISC) environment the company is positioning as its platform for the future" (Martin, 1989).

The inconsistencies vary from equating *architecture* with *environment*, at the same time that such architecture –or environment- is *positioned*, the same way we *position* a product in a commercial or marketing strategy.

The answer to this frequently detected phenomenon of multiple-metaphoric use for describing computing concepts is that computer science is a new species of science with a high degree of abstraction. As we deal with highly abstract and arbitrary realities (usually called 'virtual') we are obliged to give names to a very large number of things that may not have obvious designations. "Computer discourse is replete with creative, colorful, and sometimes

whimsical nomenclature with only limited scope (i.e., it is used in one system or document and not recognized industry wide)" (Johnson, 1994, p. 97).

When we read that a repository is a dictionary, a 'storage facility', and a 'foundation' we immediately recognise multiple metaphors where each one contributing with a set of features that, when joined in a blend, they collectively describe the abstract *thing* we are intending to describe. So, the usual difficulty of constructing a viable computer discourse is a real difficulty, based on the problem of *choosing* metaphors but not derived from the fact of having contradictory metaphors.

Regarding the inconsistency pointed out by Johnson, there is no such a thing since we have to read the list not as a heterogeneous set of literal meanings but as a conceptual integration. From this new point of view, we have each mental space contributing with its own frame to the conceptual integration:

- **Repository:** a place where things are stored and can be found
- **Dictionary:** a book in which words are listed alphabetically and their meanings, either in the same language or in another language, and other information about them, are given
- **Storage:** the putting and keeping of things in a special place for use in the future
- **Foundation:** a support in the ground for a large structure such as a building or a road. The base of a belief, claim, idea, etc.

Once all frames have been stated as above, we clearly see that a piece of software can be a *container* to store other pieces of software (things); that these things are *stored* in a special place and order (a physical device is also needed); that these things can be *accessed by a given order* (for instance, alphabetically) like in a *dictionary*, and things have a description –or *definition*- in terms of the same or another *notation* used for constructing such things. Finally, this new piece of software will be the *base* upon which other applications will be constructed, that is, they will use the repository to perform their functions.

The blend receives partial structure from all input spaces, but we do not need to think that the new conceptual integration has to be consistent with each input mental space. It is the *referential* –literal- conception of language that produces the effect of considering the emergent blend as inconsistent or contradictory with those input spaces. The conceptual integration has produced a *new* concept, even if it is based on the raw material of the input spaces. So, it is not the supposed relationship to other pre-existent concepts that make useful the conceptual integration, but its suitability to the purpose at hand: to describe a new software application, which may be described using a new concept based upon a set of different metaphors.

As conceptual integration is something we continuously apply without being aware of it, it would be a reasonable decision to apply conceptual integration in a conscious way. In particular it would be interesting for our goal to analyse a particular method –top-down- using the previous analysis.

5.1.2 Conceptual Integration offers new features to be used in Software Design.

Another interesting example of conceptual integration is that of *computer virus*. When analysing the conceptual system underlying the blend, we can observe that the system is based on an integrated schema. This schema corresponds to the generic space we have already seen

when describing the general structure of a blend and it is listed here in the following statements:

- "x is present, but unwanted; it comes in, or is put in, from the outside; it does not naturally belong;
- x is able to replicate; new tokens of x appear that have the same undesirable properties as the original x;
- x disrupts the 'standard' function of the system;
- x is harmful to the system, and hence is harmful to users of the system;
- The system should be protected against x; this might be achieved if the system were such that x could not come into it, or if other elements were added to the system that would counteract the effects of x, or eject x, or destroy x." (Fauconnier, 1997, p. 19)

The blend implies that we, in place of considering the analogical properties between computer programs of a species and biological organisms, treat such computer programs as viruses. We don't just say that the harmful program is "like" a virus. We go ahead and *call* it a virus, this is the important fact: a metaphor –unlike an analogy- says A is B and not A is like B.

As a consequence of calling it a virus, we continue using the health vocabulary and apply it to the target domain of computer programs:

Infections
Spread
Contaminated
Immunity
Healthy
Vaccine
Disinfectant
Safe

Once the blend is made, we would not necessarily have implemented the computer program called a virus. Even if we have already implemented such a program, we are not just conceptualising an already given domain in a certain way, we are actually building it so that it fits the mapping. The blend leads, manages, push the implementation of new finding programs, counter-programs, and blocking devices that will fit the generic conceptual specifications of the health/computer analogy (op. cit. pp. 20-21).

The important fact is that the blend offers a wide range of ideas for design. As Fauconnier points out, in many cases the categorisation goes through: "the categorisation is viable and guides the technical work with some success. But note once more the high-level schematic nature of the mapping: at a lower level, the technicians will no longer be using the biological analogy- they will be relying on domain-specific knowledge about computers" (op. cit. p. 21).

The blend guides the technical work offering new ideas to the design. In this sense, the blend is a repository of open-ended solutions for developers. As such, this source would be maintained in its informal state in order for it to produce all the features –or functionalities- the new system might require. The idea of linking requirements to realisations (refinements) of them –traceability- may be applied to the blend, so that frames of the source domain(s) may be detailed and also traced to realisations as well.

5.2 Mappings as morphisms

The work by Fauconnier and Turner about blending –or conceptual integration- has caused some other authors -in the field of HCI- to begin to publish their own, in particular Joseph Goguen (Goguen, 1999, Malcom and Goguen, 1998).

Such as pointed out by Goguen,

> "a user interface can be considered as a representation of the underlying functionality to which it provides access, and thus user interface design can be considered a craft of constructing such representations, where both the interface and the underlying functionality are considered as (structured) sign systems".

Thinking of a user interface (UI) as a representation of a previously existent functionality, may be consistent with some specific developments (like the first versions of Windows, an upper layer upon MS-DOS) but not with a new development, in which both the UI and the functionality are simultaneously developed; so the UI can not be considered as a representation of the underlying functionality.

In order to study the mappings between the underlying functionality and the interface (semiotic morphisms), the author proposes to use a mathematically precise theory of semiotics, which has been called *algebraic semiotics*. But one surprising issue is (in the opposite direction of the criticism made about the use of metaphors in user interfaces, as expressed in (Malcom and Goguen, 1998)) the affirmation that the "job of user interface designers is to build good metaphors (representations, translations, etc)" (Goguen, 1999, p. 1).

Once this inconsistency has been assimilated, we can go on and try to analyse some of the goals of algebraic semiotics. One of this goals is to get a *calculational approach* to user interface design, in contrast to what Turner and Fauconnier (1995, p.202) explain when they consider the blend as "not constructed by union or intersection of the inputs; it is not a skeletal or fixed mock-up of a few elements from the inputs, but it has a life of its own in the sense that it contains structure *that is not calculable* from the inputs and that it can be developed, once it is set up, on its own terms". (Italics added by me. M.I).

We have here the main contradiction between Fauconnier and Turner's ideas and those of Goguen. After Goguen, it is very important the fact that morphisms could preserve structure ("structure preserving morphisms are often at least as important as the structures themselves" (Goguen, 1999)) while in the words of Turner, "in metaphoric mapping, for those components of the source and target domains determined to be involved in the mapping, do not violate the image-schematic structure of the target, and import as much generic-level structure from the source as is consistent with that preservation" (Turner, 1991, p. 274).

We see that in a metaphoric mapping we have to:

1) Determine those components of the source and target domains to be used in the mapping
2) Not violate the image-schematic structure of the target and
3) Import as much generic-level structure from the source as is consistent with that preservation

In order to illustrate this issue, we are going to see in detail –from the whole and complex blend involved in the *desktop metaphor*- only the partial blend that corresponds to the trash can.

5.3 Blends imply emergent structure

So, if we are not to violate the image-schematic structure of the target importing while at the same time using as much generic-level structure from the source as is consistent with this preservation, it is evident that the resulting sign (represented by the trash can icon) will usually perform a transition from an empty state to a half-full state, but will never achieve a 'full' state. This is a consequence of the image-schematic structure where *there is no limit to the amount of objects to be deleted.* This virtual state could be reached as a consequence of an exceptional operating system and hard disk conditions, but it is not an *inherent trash can state*, that is, it is not a state resulting from the blend The obtained result is just the opposite.

This example clearly shows that a blend should not preserve structure in general –this structure preservation would imply the preservation of structure from both input conceptual domains. Moreover, the idea of importing structure from a sign system –such as expressed by Goguen- implies that the 'signified' of the source domain is clearly stated. But what conceptual integration has taught us is that meaning is a complex process in which there is a construction of new additional meaning when new links are established between mental spaces.

Some structure is not even 'objectively' contained in the target domain as we naively would think. The structure of 'no limit to the amount of objects to be deleted' is emergent when contrasting both domains. Without this opposition between the target and source domains, we would have never worried about the total amount of files to be deleted as a *restricting limit.* The only limit would have been the whole set of objects. The idea of limit derives from the finite capacity of a real trashcan when taken as a source domain, from projecting structure from such an input space.

This structure projection is the image-schematic preservation, but it has to be established in each individual case, as the structure of the target domain is in turn transformed when linked to the source domain, as we have seen in our description. The emergent structure cannot be anticipated and is not deducible from the input spaces.

In summary, the feature of a 'limited capacity' that would have been projected from the source input mental space, is 'disabled' by the second input mental space since in the second there is no notion of limit for the number of folders or documents to be deleted. But *disabled* is not meant to indicate here any type of calculable operation, it is only an indication of a motivated restriction.

Another emergent structure is, for instance, dragging icons with the mouse. This operation belongs to neither moving objects on a desktop nor giving standard symbolic commands. It is a consequence of having the operation for moving an object onto another in order for the latter to contain the former. As we have -in the desktop interface- an integrated context with its own coherent structure, it could happen that, while moving an object onto another, we accidentally release the button of the mouse. In such a case, we would have to produce an aborted moving, but the resulting operation must be preserved to guarantee the integration of operations. The new emergent operation is a *pure dragging* operation.

It could be argued that the drag operation is already present in the use of the mouse. As the device used to interact with the user interface may vary from a wide range (joysticks, balltrack, a finger we move on a surface –is the finger dragged or moved?- and so on), it would be considered in function of the original device used when the computer-based sign was primarily designed.

> "From an "objective" point of view, this activity is totally novel—it shares no physical characteristics with moving real folders, and it is novel even for the traditional user of a computer who has issued commands exclusively from a keyboard rather than from a mouse. Yet the whole point of the desktop interface is that the integrated activity is immediately accessible and congenial" (Fauconnier and Turner, 1998b).

We have only shown two of the input conceptual spaces of the blend. In order to illustrate the complexity of projections between the domains involved in the blend as well as the transformations –derived as structure projection from each other- which have been affected by the morphism. We agree with Fauconnier and Turner in their statement that the blend contains structure *that is not calculable* from the inputs. We think this is a fundamental argument against the idea of using any calculable approach for studying conceptual integration.

Goguen (1999, p.32) is aware of the complexity of the task he is involved in. As he explains,

> "it is a very difficult problem, about which little information is directly available. It is however clear that no simple algorithm based on just the structure of the sign systems involved can be used to compute meanings, because even for the simple blend of two conceptual spaces, selection among the manifold possibilities is governed by external factors in a very complex way, crucially including the *values* of the person doing the blend".

As we think the formalism proposed by Goguen (category, ordered category) has a limited usefulness in the case of blends, we propose here the use of an alternate approach that is closer to our subject, that of patterns. As an illustration of the Goguen's findings, when using his formalism to discover blends, it may occur that the resulting blends be objects like boats for transporting houses or amphibious houses. This type of combinatory could be amusing for "some mad engineer" –as Goguen points out- but unlikely useful when trying to obtain a blend that helps us in the designing process.

Finally, such as stated by Goguen "a user interface for a computer system can be seen as a semiotic morphism from (the theory of) the underlying abstract machine (what the system does) to a sign systems for windows, buttons, menus, etc." (Op. Cit. P .5). It could be inferred that Goguen's interest is to study the relationships between two sign systems already defined (on the one hand, the abstract machine; on the other, windows, buttons and menus), while when we speak of blends we are intending to study how we could establish a new sign system - with windows, trash can, etc- as a blend from different domains. Apparently, Goguen's goal is a semiotic morphism to be applied to two sign systems with signs well established, with meaning already formalised, in which case the idea of mapping appears as more adequate, but in such a case, we would not have blends anymore.

5.4 Optimality Principles of Conceptual Integration

In any conceptual integration network there is evidence of some principles, which are satisfied to a certain degree: *Integration*, *Web*, *Unpacking*, *Topology* and *Good reasons*. From all these Optimality Principles -stated by Fauconnier and Turner (Fauconnier and Turner, 1998b)- that a blend can meet more or less well, we think that for computer-based signs both Integration and Topology are especially important.

Integration: *The blend must constitute a tightly integrated scene that can be manipulated as a unit. More generally, every space in the blend structure should have integration.*

It is not a tightly integrated scene that of putting folders and documents on the trashcan, the same way we put floppy disks on the trashcan. There is no integration as the effects of dragging different items to the trashcan are also different.

Topology: *For any input space and any element in that space projected into the blend, it is optimal for the relations of the element in the blend to match the relations of its counterpart.*

The topology principle is not satisfied when we use a trashcan to eject a floppy disk. When using a real floppy disk device -or other type of devices- there is usually a button to get the floppy disk –or tape- ejected. What we need is something equivalent to pushing a button, even if such an action would be performed –on the user interface- metaphorically, for example dragging the floppy disk to a floppy disk device.

Web: *Manipulating the blend as a unit must maintain the web of appropriate connections to the input spaces easily and without additional surveillance or computation.*

The application of this optimality principle is useful when considering blends such as the Buddhist monk story. What happens in the blend –the same Buddhist monk walking up and down simultaneously- maintains appropriate connections to both input spaces: the Buddhist monk begins at dawn to walk up a mountain and the day after the same Buddhist monk begins at dawn to walk down the same mountain.

Unpacking: *The blend alone must enable the understander to unpack the blend to reconstruct the inputs, the cross-space mapping, the generic space, and the network of connections between all these spaces*

An important aspect of the Unpacking principle, is that it is the basis for our tracing elements from upper degrees to lower degrees of abstraction. So, in order to guarantee the traceability between different artifacts built during the process of software development, we have to take into account the unpacking of each of the blends.

Good reason: *All things being equal, if an element appears in the blend, there will be pressure to find significance for this element. Significance will include relevant links to other spaces and relevant functions in running the blend.*

5.5 Conceptual integration as patterns

In this Chapter, we prefer to use the idea of *pattern* as this concept is closer to the idea of image-schema. The first scholar to see the relationship between image-schemas and patterns is Mark Johnson, who points out that when understanding 'schema' more loosely than he does it is possible to extend the list of image-schemas he provides. He adds that "Christopher Alexander, for example, has developed an account of some 253 recurring 'patterns' that influence our experience...Most of Alexander's patterns are not image schematic, in my sense, and would require a somewhat different kind of analysis than the ones given above" (Johnson, 1987, p. 126). This point shows a first relationship between patterns and image-schemas and suggests us the possibility of representing blends using patterns.

But there is another reason for choosing patterns as a tool for designing computer-based signs from metaphors and conceptual integration. In spite of the criticisms patterns might have generated, we prefer to use such a schema for representing the computer-based signs as our purpose is to use them in a more flexible way.

Some authors (Gartner et al., 1998) include this cognitive process in a broader schema for cognitive models:

- **Domains** (Goals/principles/reasoning associated with recurring situations; serve as explanatory structure for expectations regarding a situation, as in 'sorting activities'
- **Frameworks** (Domains with additional context information, as in 'sorting mail' activities
- **Cognitive Maps** (Frameworks that are oriented towards wayfinding; finding one's way through a problem, as in 'sorting mail when address is incomplete'
- **Patterns** (Detailed, very context specific instantiations of frameworks, as in 'sorting by zipcodes' (p. 10)

In this sense, we use the word pattern with an intermediate meaning, midway between patterns and cognitive maps. Even if we have adopted the classical formulation, it is evident that we are not dealing with classical classes and associations, but with the vagueness of conceptual integration, derived from an initial metaphor (as the desktop metaphor). Even if we are going to use this concept as explained, it is worth using the classical structure for pattern definition –'name', 'context', 'problem' an so on- as this structure forces us to increase the precision of the conceptual integration development process.

While image-schemas are structures that are constantly operating in our perception, bodily movement through space, and physical manipulation of objects (op. cit., p. 23), patterns are recurring structures we use in our professional life, that is, they are used in the context of more abstract contexts. They are conceptual configurations that frequently appear in the context of our everyday experience as –for example- software designers.

The introduction of patterns in software design means making apparent some unconscious schemas used in the design activity. The use of patterns currently extends from programming to analysis through process definition. The central discovery with patterns is that most of our mental activity is a recurring one which only changes the degree of granularity of phenomena to which patterns are applied. We propose here to extend the usual schema of patterns in order to be able to represent *conceptual integration* or *blends*.

Encompassing cognitive processes to which they apply, patterns are defined at different levels of granularity and they may be combined in nested structures to form -as Christopher Alexander proposed- a language of its own. But design patterns, in general, are based on already defined software constructs such as classes, objects or other.

In order for a pattern to be able to capture conceptual integration, it has to be defined as a network to mimic the network of conceptual integration the pattern will represent. What we are proposing in this Chapter is to generalise such patterns as to include other type of mechanisms, which could be used to design new constructs. The mechanism of conceptual integration implies "a variety of constructions involving analogy, metaphor, and hedges set up multispace configurations with source, target, generic, and blended spaces that project onto each other in several directions" (Fauconnier, 1994).

5.6 A Pattern for Conceptual Integration

5.6.1 Pattern Name

RunningTheBlend

It is interesting to observe how the authors that proposed the conceptual integration approach have simultaneously introduced the –computing- concept of *running* the blend. It is important to point out here that running in this context means to have a progressive in-depth relationship with the blend, trying to get some response back from the analysis of the elements involved in the conceptual integration. The aim is to get a consistent blend, assuring at the same time the integration of all elements. The iteration schema derives from the fact that we have to perform many roundtrip journeys, in short, we apply the backtalk concept in order to have a reflective conversation with input mental spaces (Schön and Bennett, 1996). "One form of judgement in which I'm particularly interested is the kind that I call backtalk, where you discover something totally unexpected –'Wow, what was that?' or 'I don't understand this', or 'This is different from what I thought it would be- but how interesting!'. Backtalk can happen when the designer is interacting with the design medium" (op. cit. p. 176).

5.6.2 Context

Metaphors are good at suggesting a general orientation, but not good at accurately encoding precise semantics (Nardi and Zarmer, 1993). This is a quite acceptable position, considering that a metaphor is a very general and vaguely defined mapping between two different domains: one is well known and the other the less so and requires further investigation. The question is that, in the design process, we are looking for new and helpful ideas that can contribute to our final product. What we have to do is transform the original metaphor into a refined conceptual framework that could be implemented as new constructs (computer-based signs, with handling, permanent and transient features). The metaphor is part of the needed raw material, the new constructs -blends- are the final products.

Regarding the precise semantics that metaphors lack, we need to take into account that, at this point of design "is not precision that counts, but the fertility of the concepts" (Heisenberg). The next steps of design will refine and increase the precision of semantics.

On the other hand, we usually have a creative metaphor, which is the base of new signs, but such computer-based signs are to be built on a set of other input spaces. Based on that set of

spaces we will get a conceptual integration that will be compatible with some optimality principles: *Integration*, *Web*, *Unpacking*, *Topology* and *Good reasons*.

5.6.3 Problem

So far we have neither been given conceptual tools nor guidelines about how to apply a given metaphor in a disciplined way. Once a new metaphor is conceived, it is just applied in an ad-hoc way using mostly handicraft methods. Apart from the fact that methods used are handicraft, we usually apply them in a way that we are not conscious of, similar to *driving-a-car* operations. Activity theory establishes a difference between conscious (actions) and unconscious (operations) behaviour while we perform in a given activity. As the activity of conceptual integration is one used in our everyday life (reading a newspaper, thinking about a given personal problem or political situation, difficulties at the workplace, etc) we are not usually aware of the mechanisms employed in such conceptual integration. This is why we think it would be useful to point out the unconscious mechanisms used in conceptual integration in order for us to use them in a disciplined way.

As the mechanisms of conceptual integration are part of our deep cognitive processes, it is not possible to describe them as an algorithm or procedure that we could follow mechanically in order to get a given goal. But even a partial description of the interactions derived from an example of conceptual integration would be useful in our discipline of Human Computer Interaction.

5.6.4 Forces

We are considering, on the one hand, a mainly creative process based on a new metaphor which has to be explored and developed. The original idea of the metaphor –as most of creative processes- could have been produced spontaneously, in a non-predictable way. On the other hand, this creative process has the usual characteristics of handcrafting and non-predictable results.

5.6.5 Solution

When applying Conceptual Integration (or blends) to design User Interfaces, we could detect a set of input spaces to which apply partial blends, using different couples of spaces. This procedure might become a long –and bothersome- one and eventually not quite practical for our purpose of applying it as a procedure for User Interface design. Instead, we propose to apply two main Conceptual Integrations: the first based on the source metaphor space ("the interface is an office" that is, the dektop input space) and the space of operating system commands. The second belongs to a lower level of abstraction -the design level- in which we refine the analysis concepts by applying a new blend.

5.7 The Analysis level blend

The first conceptual integration belongs to an abstraction level -which we propose to call the analysis level- where we find the main concepts derived from considering the desktop blend.

The steps needed to defining the new blend space are:

5.7.1 Detection of input conceptual spaces

The first step is detecting the input conceptual spaces involved in the conceptual integration. In this case we can have two input spaces:

i) Real Workplace Desktop

ii) Computer Commands

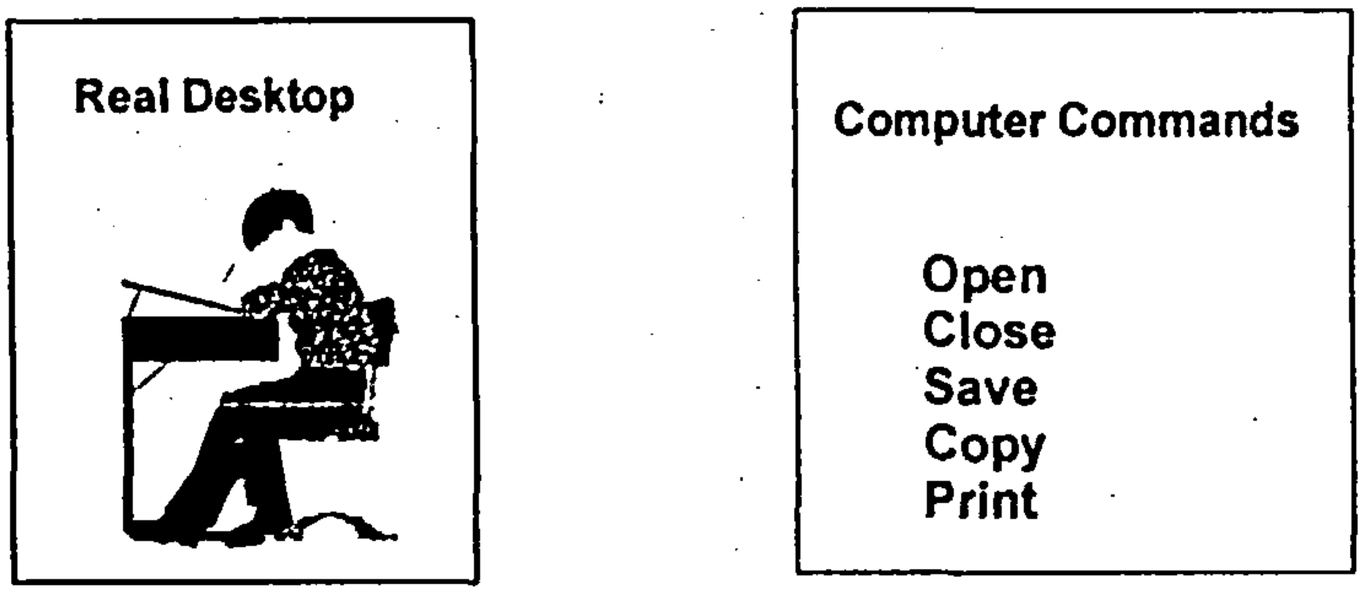


Figure 28 Input Conceptual Spaces

5.7.2 Frame description for each conceptual space

As each of the input spaces has a different organising frame, a detailed description of each one will be valuable for the current blend. The description is the source for detection of individual elements of one space that will be projected (or not) to the conceptual integration. As the real workplace desktop is the source input space of the foundational metaphor, a careful detailed description will be indispensable.

Real Workplace Desktop

The frame of the real workplace desktop has elements like folders, documents, and trash cans. It could be added other items like a calendar, an agenda, a telephone book and so on. But the selected list of elements in the original frame used for the MacOs graphical interface, there were only folders, documents and a trash can. In such a frame, we organise our documents into folders, so we have to *open* a folder to get a document; we *write* documents and then we *archive* them into folders (put them in a *safe* location) and so on. We can *copy* groups of documents, so while the original is in one folder, the copy is in another folder.

Computer Commands

The frame for commands may exist based on previous versions of the operating system or previous OS's. It does not matter if the commands have been previously defined or will be simultaneously developed with the graphical interface. The abstract machine is available and provides commands like write, delete, open, close, copy or print. These are the usual

commands used in previous operating systems and allow the definition of a frame for operations that will be executed as a consequence of interacting with the signs.

Sometimes, one of the two input spaces is not a previously conceptualised space, that is we have not equivalent computer-based concepts like in the case of computer -operating system- commands. Consider the case of a calculator (or a hi-fi device or an external device of the computer), which represents an autonomous device without an equivalent set of commands in the computer. In such a case, we may apply the concept of *simile*, which establishes an isomorphism (a metaphor always establishes a partial mapping between both input spaces) between the device's operations and those of the represented device. When using a simile to represent a calculator, for example, we decide to represent a standard calculator with most of the keys and operations used in such real electronic devices. When using similes there is no emergent structure, as the concept of isomorphism implies the analogy (in terms of structure and behaviour) between both elements: the real device and the simile.

5.7.3 To select the main operations of concepts defined in the previous step

For example, we can associate to a folder operations such as "open", "to put into another folder", etc. It is interesting to observe that some operations –like opening a folder- are derived from additional sub-blends (applied on the conceptual spaces of manipulative actions performed with external devices –mouses, joysticks, trackpad- and the graphical representations of the main concepts we are dealing with -folders, documents, trashcans-).

From the frame of real world objects, we project some additional operations: for example, *to select* an element, that corresponds to an intentional goal. Before moving any object, I have to intentionally focus my attention on it or to grasp it with my hand. So, there is a *selection* of the object I will act upon (move) and it is only after this selection has been made that I can move the object.

5.7.4 To detect new concepts and operations emergent in the blend

Associated with the *open* operation applied to a folder and considering the constraint of a limited space available in the screen, we need a new space where to show documents and other folders contained in the opened folder. The need for a new space lead us to create a new concept –it might be the *window* concept, with new operations associated to it- where to show the content of the opened folder. Considering this issue in detail, we can observe that this new emergent concept derives from a new sub-blend applied to two different input spaces: one is the blend we are currently considering in –the desktop- and the second is the space of buildings. In this second space (or frame) we see windows through which it is possible to observe many different scenes.

The new concept –the window- needs in turn the definition of new operations. As the constraint of limited available space in the screen might prevent to display all elements contained in the folder, we need to define an operation to *modify the size* of the window. But this solution might be not enough to show the complete content, so we may be forced to create new operations *to browse* the content of the window horizontally and vertically. When analysed in detail, these new operations derive from applying another sub-blend to two input spaces: the space of computer based windows and the frame of human vision, where a person moves her head to have a greater visibility, to increase her field of vision. This head movement

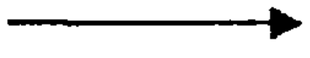
has been projected into the *window* blend as an operation to browse the window horizontally and vertically.

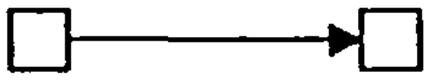
5.7.5 To iterate the process with each sub-blend

Trying to follow the complete set of sub-blends involved in the design of User Interfaces might be extremely bothersome. In order to follow the procedure in an ordered way, we must consider that for each new element and/or operation, we may need create new blends to which we will apply steps 5.7.1 to 5.7.4 above. This lead us to an iterative process which will be applied so many times as necessary until the main conceptual integration –the desktop- is stable. It must be taken into account that this stable condition for a blend is a relative one and the whole process may imply months or even years of continuous re-working.

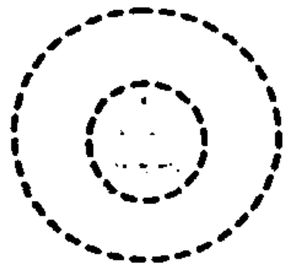
5.7.6 To represent the analysis elements and operations

In order to represent analysis elements and operations, we propose to use image-schema graphical representations of the sort:

Link: 

Source-path-target: 

Container (for any analysis element): 

Centre-periphery (for a general analysis structure): 

These graphical representation are neither icons nor computer-based signs, they are only schematic representations that must to be refined –completed- as icons or computer-based signs.

For example, the folder –as an analysis element- may be represented as a container:

Eventually, after completing the design level blend, it must be assigned a graphical representation more suitable to the idea of folder we are dealing with:

The idea of separating both levels –analysis and design- is a good solution to the problem of allowing reasoning at a conceptual level without the need to be constrained by technological considerations. In order to consider the main features of the folder –operations and, possibly, emergent structure- we do not need to have a definitive graphical representation of the concept.

5.8 The Design level blend

The second Conceptual Integration belongs to the design level, *in which we refine* the analysis concepts by applying a new blend. It is interesting to observe that even both concepts *–to constraint* and *to refine-* may be considered as the result of applying conceptual integration. The new design blend results from taking the new concepts and operations obtained in the analysis blend and a new space, that of *computer-based signs*. This new space involves such issues as states, colours, sizes, etc. as we have already shown in Section 4.8.

The needed steps for defining the resulting blend space are:

5.8.1 To consider each analysis element in order to assign graphical features

This step aims at assign graphical representations (with different forms and colours) to analysis elements. To do so, it will be necessary:

1) To select one element from the analysis blend, for example a folder.
2) To consider all possible states associated with the folder.
3) To map graphical representations or colours to each state (this might mean to change the colour of an image or icon). For example, the possible states of a folder are closed, opened o selected. Sometimes, the state of *open folder* may imply the use of a different image. In other cases –as in the case of *selected folder-* we can use a different colour to indicate it. Observe that the state of *selected* derives from a sub-blend using as the first input space that of the analysis blend and the second input space (or frame) that of visual field or that of manipulative actions, as indicated in step 5.8.3. A selected folder might correspond to a glance at the folder or to the folder in our hands (not yet opened).

5.8.2 To consider each operation in order to assign actions we must perform

To each operation applicable –for example- to the folder detected during the analysis blend, (open, select, move) we need to associate actions we might perform *using an external device* like a mouse, a joystick or a trackpad. This is also a sub-blend applied to the space of elements' operations and the actions performed using an external device. When executing this new sub-blend, we can produce new concepts not included in any of the input spaces. For example, as we have constraints when using mouses –we can press down the left or the right button, sometimes there is only one button- we might need to associate to the operation –*open* a folder- a new action, that of using a *double click*. This is a new meaning not available in any of the input spaces, where we only had "to open a folder" or "to press the left button". The new operation appears in the blend, into which we project the concept of *operation*, and the action of performing a double-click.

5.8.3 To iterate the process with each sub-blend

The same way we proceeded in the analysis blend, we might need to apply Sections 5.8.1 and 5.8.2 iteratively to each new sub-blend we may get when performing such steps.

5.8.4 To represent the design elements and operations (to create the computer-based signs)

In order to represent the design elements, we may use icons defined using:
graphical design principles
usability criteria
user interface standards

Sometimes, we may find that analysis and design graphical representations have an identical appearance, but what must be taken into account in such a case, is that they are concepts belonging to different abstraction levels. In this sense, it is usual to speak of the metaphor or icon as equivalent to the design computer based sign. For example, it is usual to speak of the trashcan metaphor (or icon) as *responsible* of contradictions and ambiguity, instead of speaking of a design problem or a bad design decision when designing the trashcan *computer-based sign*. Assimilating both levels (analysis-metaphorical and design-sign) we may deduce erroneous conclusions about the usefulness of metaphor. In the case of the trashcan used to eject floppy disks, what is to be criticised is a design decision and not the use of the desktop metaphor.

The trash can computer-based sign

Besides the two general features (*select* and *move*, applied to any computer-based sign of the desktop metaphor), we can recognise two specific ones: open the trash can (to see its contents) and empty it when we are sure its content is no longer useful to us.

We propose to use matrices in order to include more specific items regarding the blend. The design level blend is shown this way:

	**Empty**	**Delete all**	**Click on an option**	**Menu**
	**Select (focus)**	**(Subordinated to the next)**	**Click on the sign**	
	**To put in**	**Move to**	**Drag and drop**	**Document, folder**
	**Open to see**	**List items**	**Double-click**	**All objects contained in the trashcan**

Figure 29 Table for Representing a Blend

After performing this blend, we can make some considerations in relation to the context:

As we have already assigned the manipulative operations click, drag, drop and (let us imagine) double-click (which has been already assigned for opening the trash can as there is no order in the blend), we have to imagine a new operation. One possible solution would be a triple click, but for each additional operation we would include another click and it does not appear to be a reasonable design solution. So, for example, the decision for using a menu option is taken. The important issue here is that *there is a difficulty in the blend* and it has to be solved. The difficulty derives from the fact that there are only 2 or 3 elementary actions that can be used with the mouse and we have an increasing number of operations to be assigned to them.

Another aspect is the concept of a menu, which we have introduced here as a *natural* idea. In fact, this is also an evolution in the process of conceptual integration and it was used by the Macintosh designers, but not by their predecessors who designed Star. Such as David Liddle points out "We did not call any of the mechanisms in the Star menus. Apple always called them menus; we never did. But the idea was there nonetheless, tied to the objects on the screen. Instead of being in command space, where you were always asking, "Okay, what's the next command?" you sat there with a restful screen full of icons. You pointed to one of them, and then you hit properties or options, and you were presented with all the operations that you could do with the object, as well as its characteristics". (Winograd, 1996).

The operation of putting something in the trash can is derived more or less immediately from the real world image-schema: we take an object and put it into the trash can. The equivalent with our signs would be to grasp and bring (to *drag* is the equivalent given the restriction of having only two dimensions) the object to the trash can. When we reach the trash can, we release (drop) the object. The same restriction of being in a two-dimensional space determines that we drop the object *on* another and not *into* another; a three-dimensional representation of objects would allow us to put, for example, a document *into* a folder. A solution provided with the lasts versions of MacOS allows the possibility of moving a document onto a folder, but after remaining on the sign for a while it opens automatically and permits the dragging of the document *into* the folder or repeating recursively the operation in another folder contained in the first one. Another evolution of the conceptual integration.

This approach is used repeatedly when interacting with a computer system: in order for an object to interact with another, we drag the first object onto the second. This drag and drop operation will have a different meaning depending on the objects involved in the interaction. As we can see currently in our user interface, when we drag and drop a document onto an application sign, the application execute by opening the document we have put onto it.

As we have already shown at the beginning of this chapter, in this blend we can detect that the trashcan will never fill up completely. This consequence is derived from the fact that the underlying operating system command implies moving the object from one location to another, and if there was enough space to store the document (or folder) before the move, there will be enough space after the move as well. This feature is projected from the frame of computer commands and restricts the potential projection from the frame of real world trashcans. It is something nobody could have anticipated since it emerged when making the blend. That is why there is no a graphical representation for a full state of the can.

For the 'open to see the content' feature, it is evident that the graphical representation of the half-filled trash can does not allow us to see anything inside it. After using the *open* operation,

it would be necessary to enlarge the same graphical representation (a big trashcan with enough capacity to list its contents) or to choose an alternate graphical representation. As the *window* sign is also available for other purposes, it is a good candidate to be selected as a viewer for the trash can's contents. So, this is the solution that has been chosen along with a general solution so as to see the inside of other objects-containers (folders): a window as a view of the inside of a container.

This analysis would have continued thoroughly until the sign has been completely defined at a design level. This complete definition may need additional models in order for the sign to be fully specified. For example, additional state transition diagrams would be very useful as each state and transition has to be mapped with different graphical representations. In general, all the artifacts offered by UML (Unified Modeling Language) are available to create the set of diagrams necessary for a complete modelling of the sign. Once the complete set of operations is stable and has successfully passed the test of optimality principles the moment for the modelling to begin has arrived.

Using UML, each operation could be captured as a Use Case. At this point begin a classical software engineering task, but –very important- *once the conceptual integration is finished.*

5.9 A Use Case for specifying the trash can blend

The use case for *empty* the trash can could be represented as follows:

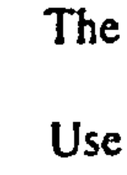
Use Case: Empty the trash can

Pre-Condition
The trashcan is not empty

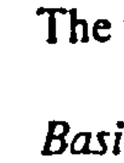
Basic Path
1. The user selects an option to empty the trash can
2. The system shows a message about the trash can's content and asks for confirmation
3. The user confirms the empty operation
4. The systems empties the trash can

Post-Condition
The trashcan is empty

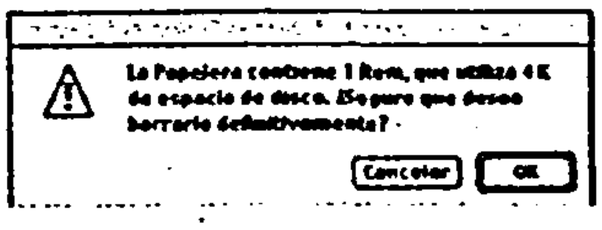
Initial graphical representation



Final graphical representation



Message shown by the system



The process of signs construction continues until all signs have been completly specified (and modelled using UML). Once all signs needed for a given graphical interface have been specified, we have to apply a set of Optimality Principles to them.

5.10 Why the Mac trashcan computer-based sign has design problems

5.10.1 The trashcan: metaphor or computer-based sign?

In order to discuss both principles in relation to the trashcan, we will suppose that we have also taken the same design decision as the Macintosh designers, so the trashcan is also used to eject a floppy disk.

This decision is considered by many authors as a problematic solution. Let us see what some authors say about it.

Malcom and Goguen (1998) write that "there are dangers to using metaphors in designing interfaces, the most notorious example being the use of the Apple Mac trash can to eject floppy disks from the drive". This comment is particularly odd coming from two authors with a background in metaphor and blend. Paraphrasing Goguen's comment (1999), there is a danger of throwing out the baby of metaphor with the dirty bathwater of ill-designed computer-based signs.

We have already analysed –in Chapter 4- this symptom: the fact of attributing to metaphor a problem that has nothing to do with it. If we have to imagine some reason why the Macintosh designer made this decision, it is likely that they thought there was a reason to economise on signs: as the icon for the trash can was already available, why not to use it for other purposes?

Historically, this way of using a module for including a mix of functions has been a solution frequently used by developers, until structured programming and design recommended the rule of *cohesion*. And even if the graphical interface was designed at the end of 70' (when the rules of structured programming and design were quite consolidated), the use of graphical signs was introduced for the first time and it was more difficult to perceive cohesion in them. In any case, it is a design decision and is not dependent on the underlying metaphor, otherwise the whole set of signs used in the graphical interface would have had the same problems and this does not seem to be the case.

Tim Rorher has an interesting paper about metaphors, image-schemata and HCI (1995) in which he points out that "this feature of the Macintosh desktop is so quirky and counterintuitive that a computer lab on my campus has a sign on every Macintosh which explains that 'in order to eject the disk, the user should drag the disk icon onto the trash icon and drop it' ".

This is not the only design problem with the MacOS, as the following story that occurred to me can testify. A friend of mine phoned me and was very worried since she had just been displayed a *bomb* message on her Mac screen. I explained to her that in such a case she had to restart the computer and the application and usually there was no resulting problem. As we continued talking she was increasingly worried until the moment I asked her why she continued to be so nervous. I discovered that she was thinking about the bomb as a problem that in due course augmented its danger! The projection of the frame of the bomb as something more and more dangerous as time goes on provoked this -for me- strange reaction, which I did not understand at first.

A similar reaction is described by Rohrer: "My initial response was to assume that the user's discomfort was irrational, fleeting and would be assuaged by a little conditioning practice from which the can ascertain that ejecting a diskette does not delete information. Despite practice however, many users still reported discomfort at putting a disk icon into the trash, sometimes even to the extent of choosing a drastically different procedure -shutting the machine down- to eject the diskette automatically" (op. cit).

5.10.2 Optimality principles applied to the traschcan

In terms of these optimality principles, we can say that the decision of using the trash can to eject diskettes, is a failure to satisfy both Integration and Topology, derived from the fact that the floppy disk device is not something belonging to the desktop input space. Rather, it is something belonging to the computer's external devices, and it would have been better represented as a simile.

For example, there are two issues that fail to satisfy both Integration and Topology:

Topology

One ejects the floppy disk to keep it rather than discard it.

If we have applied the topology principle, we would have designed an icon to represent the floppy disk driver and, when dragging a floppy disk to that icon, it would transform itself to represent the ejection of the floppy disk. So, in place of using the computer-based sign of the trashcan, we would use a *simile* to represent the floppy disk external device.

Integration

The usual dragging operation of one icon to another results in the first icon to be contained in the second, but not so when using a floppy disk dragged on the trashcan.

But the main feature is that we move something to the trash can in order to empty its contents later, that is to *delete* all objects there are inside. This characteristics of the sign is not compatible with the principle of integration, whereby the same operation would have a coherent behaviour and not two different behaviours depending on the object we move onto the icon (the trashcan).

In order for the sign to be in accordance with the integration principle, it has to be split into two different computer based signs: the trash can and the floppy disk device. Each one is concerned with the type of objects it can handle, the trash can will handle folders, documents

of different types and applications. The main characteristic of this type of objects is that they exist in a memory device (a hard disk, a Ram memory or floppy) even if they can be transferred to other type of medium (printing, for example).

The floppy disk device will only accept floppy disks in order for them to be ejected. The simile substitutes a button that would be placed in the real world floppy disk device, as it occurs in most of PC computers in which the ejection of floppies is done manually.

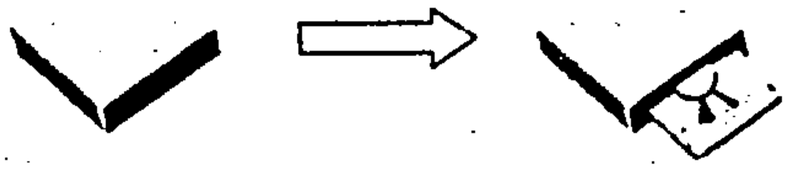


Figure 30 Similes used to represent the floppy disk device

Once this design decision is supposedly taken, it becomes clear that there is no problem with the desktop metaphor. I suppose that now there is so much history (more than 15 years) with the trash can 'problem', that people of Apple would not change their mind as this characteristic has become a *trademark* one.

Each design decision bound to a graphical interface has to be extensively tested with a group of users in order to determine if our conceptual integration works correctly in the context of other computer based signs. But this is an additional question no to be considered here.

5.11 Conclusion

In summary, it is worth including here the extremely useful yet extensive summary by Turner about Conceptual Integration (blends):

"Blend in [projection] has the following general principles:

The blend exploits and develops counterpart connections between input spaces.
Counterparts may or may not both be brought into the blend, and may or may not be fused in the blend
The projection from the input spaces is selective
Blends recruit a great range of conceptual structure and knowledge without our recognising it.
What has been recruited to the blend can be difficult to discover.
A blend may have many input spaces.
Blending is a process that can be applied repeatedly, and blends themselves can be inputs to other blends.
Blends develop structure not provided by the inputs.
Blends can combine elements on the basis of metonymic relation.
The recruitment of a conventional metaphor to the blend is in general partial, selective, and transforming.

> Inferences, arguments, ideas, and emotions developed in the blend can lead us to modify the initial input spaces and change our views of the knowledge used to build those input spaces." (Turner, 1996, p. 83)

The application of conceptual-integration patterns as a network of input spaces and a blend space would allow the avatars of different concepts to follow that will be combined in new design constructs. The proposed schema follows the model applied to explain how meaning develops from a given sentence or paragraph applied by cognitive semantics scholars. The new types of patterns proposed will be able to connect partial structure from different input spaces and to give the design basis for all blends obtained from the original spaces. These gradual transformations also reflect the transition from informal to formal description of design elements through intermediate -semiformal- stages.

5.12 Application to the Flex Project

In Flex we have two concepts that are candidates to be considered as blends:

5.12.1 The first concept is that of *infotainment.*

The first symptom we observe in this concept is the name itself: *infotainment*, an integration of *information* and *entertainment*. This is a good clue of the evidence that we are facing of a blend. In the description of the HIC we read that the new device –HIC- "is neither a lean back device as the traditional TV nor a lean forward product like a PC, but a kind of moving around device where users can get and provide information while they are occupied with other house doings". This is another clue, as the new artifact is of the type "neither..nor", so it is a new kind of device not comparable with other types of available artifacts.

Supposing that our device will be a blend from different input spaces, we have to show such input spaces. Both traditional TV and PC evoke two input mental spaces from which some features may be recruited:

> i) The input mental spaces of working with a PC and the type of interactions derived from such activity. We need to type in commands, to click or double click with a mouse, or moving a joystick, or via touching, to read the information shown on the screen, etc. The activity of working with a PC implies the use of hands to communicate with the computer, as the usual interfaces are textual or graphical.
>
> ii) On the other hand we have the input mental space of traditional TV, which allows for hearing and seeing information while we are occupied with other household doings. The main point here is that *we would not need to constantly use our hands* to interact with the device, only when we change the channel or increase the volume of the TV set. Another aspect of the mental space is that we can make remarks –out loud - about the TV news, to the device in the form of dialog.

The requirements of the new device show that it will used for helping with other home activities, such as cooking or maybe with some type of do-it-yourself activities. From this new mental input space we can recruit additional features:

> iii) Other housekeeping –or entertainment- activities like cooking and do-it-yourself usually require us to constantly use our hands. In such circumstances we need someone else to help

us to search or read a recipe in a book or to tell us how to prepare a particular recipe. So, it would be essential that the new device be able to recognise spoken commands in order for us to continue with the current activity while asking for specific information.

So, we have at least 3 input mental spaces from which to recruit structure for the new device: it should offer the possibility of a classical PC interface (keyboard, mouse -or equivalent- and a monitor), the image and sound quality we require from a TV set, and the added spoken interface to leave our hands free when they are needed for other activities.

We have created a blend from which it is possible to make some inferences, and the main two are: the *mobility* of the artifact (or distribution in many locations of the house) in order for us to be able to use it in different contexts, and the mix of traditional and spoken interaction in order to leave our hands free when needed.

5.12.2 The integration of written and spoken inputs for the device is derived from the previous blend.

This integration is another candidate to be considered as a blend. The important fact is that both types of input have to be integrated in only one *blended message*. This new blend will recruit structure from the space of traditional computer interface -desktop-, and from usual conversations.

The input mental space of speech offer us examples of usual conversations including speech acts. In usual conversations we use some sentences that are considered as speech acts, that is, the equivalent of an act, of a command to be executed by somebody else. The equivalent of these speech acts will be all the commands used to interact with the new artifact; commands like the following:

Turn on, Start, Switch on, Close, Stop, and so on

From the same input mental space we can recruit some sentences that have no meaning for a machine. Such sentences have a meaning only for humans, even if they are very frequently used only as a pure formality. Such sentences are expressions like:

Please, I would like, I want to, and so on

So, in the blend we have phrases that are apparently the same as in usual speech, but that have lost their meaning because some of these expressions have no meaning at all when communicating with machines. This is a first example of emergent structure in the blend and they have been labelled 'prephrases' and 'postphrases' to indicate that they are idle, void structures (they are placed *before* -pre- or *after* -post- some *phrases*); they intend to express politeness -something completely useless with a device- and included to retain the fictional aspect of *human conversation*.

There are other phrases that may be used but are useless as well:

Me, this, the, a, to, switch to, and so on

Some of them are contextual information (*this*, *the*, *a*, or *to*) not necessary because they are deduced from the context; while others are supplementary information meaningless to a device (*me*) and others are already implicit in the sentence being employed ("*switch to* next channel").

5.12.3 Reverse projection

Applying the Unpacking Principles (one of the Optimality Principles) to a blend means projecting from the blend to the input mental spaces. The blend is part of the requirements and is complemented with additional statements like this:

> "The markets for information access to home shopping, home banking, video-on-demand, web-access, traffic information, etc. are expected to explode within the next two years" (Esprit Project no. P29158 Documentation).

For example, home shopping is stated as a requirement, which will determine a given service of the HIC device. If we speak of home shopping, we are using an underlying assumption: *people prefer shopping from home*. When performing a study for this type of project, it will be very useful to get a list of all assumptions derived from a given requirement.

In this project this type of validation has been applied to the assumption of home shopping. In (Report on household interview, T1-1, 1999/09/15, Version 2, Final) there are some paragraphs regarding the issue:

> "They almost never order take out food so there's no pizza offers hanging on the shelves in the kitchen"

> "Cindy couldn't imagine herself shopping via the device. She works in a bookstore and it's her experience that the contact between customer and staff is essential. She likes it both as customer and as staff. Also she likes to see and feel the things she buy"

It might be deduced that home shopping will be a function only used by a percentage of all potential users of HIC. But this as well other assumptions may be difficult to validate as all market trends are dynamic and may transform as long as new services are offered by the device. Moreover, all services offered on the Web are constantly changing, so if a service no longer has demand it will disappear automatically. This is only an example of a requirement/assumption pair that must be verified.

Other assumptions are obvious, such as:

> Users have TV sets
> Users have PCs

But this assumption has been explicitly included in the Description of the Flex Project:

> "The HIC addresses the niche of trend setting families who are open to changes and who already have PCs and audio-visual equipment"

Chapter 6

Experientialist Design

6.1 Background

In this Chapter we will apply most of the concepts of Experientialism to the activity of designing a new User Interface, in this case for the FLEX Project (See Appendix A). Our aim is to preserve most of the terms used in relation with Experientialism, even if these concepts may be restated in terms of previous HCI concepts.

The Experientialist approach has been used by other authors, among them by Lund and Waterworth (1999). They point out that the experientialist approach rests on the fundamental premise that *to design HCI is to design the conditions for possible users' experiences*. In the traditional approach, They continues, "the metaphor is part of the interface. This need not be the case with experientialism since, by this account, metaphor is everywhere". We think that it would be interesting to analyse this statement in detail, in order to state our own position regarding the role of metaphor in the interface. The fact that metaphor is pervasive is an unanimous consideration, but referred to language –spoken and written- and literature in general. Is the same consideration applicable to user interfaces in general?

6.1.1 First and second generation interfaces

We think that the first and second generation –maybe even early third generation- of computers had an interface that hardly could be considered as metaphorical. The small display where one machine instruction appeared (Univac SS90 or IBM 1401 and 1440) was simply an isomorphism between the real register or accumulator containing data and the equivalent display based on coloured lights (generally red). On the display it could be read, for example, 25 0002 1000, which was the representation of a ten digits instruction indicating that we were loading –operation code 25- to the accumulator the content of the location 1000 modified by an index register 2.

When computers evolved through more sophisticated languages, the interface allowed writing commands in terms of the underlying operating system's language. In this last case, it was possible to tell the computer, for example, "(to) delete (the file) Orders", having put in parentheses some additional words that allow a reading closer to natural language. This type of structure was the traditional *verb + object* syntagm, but there were also other variants. In the Star user interface (Smith et al., 1983), for example, the authors recommended a particular order between command and object, which later became very popular, and took the form *noun + verb*. The advantage of this reversed order was that when the user has selected an object, the designer knows which actions are legal at this point (Andersen, 1997, p. 91).

Independently of the chosen form, if we detect some metaphors in such commands, it is simply the consequence of using a language already full of metaphors, but not the fact of having created them explicitly for the interface. In the case of *deleting a file*, it could be considered that we are employing a figurative expression based on a metaphor: the fact that we are erasing the file the same way we delete a drawing -drawn with a pencil- using a rubber. This new meaning might have been taken from the domain of reading, recording and deleting

magnetic tapes and transferred to the domain of magnetic media in general. Note at the same time the metaphorical projection from the traditional *reading* concept.

6.1.2 Metaphors and modern interfaces

So, we consider that the creation of the graphical interface and, in particular, the desktop metaphor is the origin of uncountable new metaphors, of having placed metaphors everywhere, in terms of Waterworth. In Chapter 5 we have seen that the design of computer based signs by means of conceptual integration usually implies the use of metaphors. What is really pervasive is the use of *computer based signs* or, in experientialist terms, conceptual integration (blend). Although many computer-based signs are based on metaphors, there are many other that are not –at least directly- based on them. For example *icons*, in general, are used because their meaning emerges from a likeness between expression and content (two important features of signs). So, the usability of icons derives from a direct analogy with the object it represents, which makes the sign easily understandable. But we must recall that analogy is not a metaphor: in an explicit analogy we have a structure of the type *'a is like b'* while in metaphors, which are implicit analogies, the structure is *'a is b'* (Presmeg, 1997, p. 267)

A well-known example of conceptual integration, which is not based on a metaphor, is the case of complex numbers. Complex numbers are produced as a blend of two already existent domains of mathematics: real numbers and analytical geometry. Another example of a computer-based sign that is not based on a metaphor is the use of a graphical thermometer –or whatever measure device- in order to show temperatures of a given device connected to a computer. As we will see, this last example is a *simile* or explicit analogy, a modern version of old displays where it was possible to show some measurements made by special purposes devices (thermometer, barometer, counter, etc.).

6.1.3 The interface as a space of computer-based signs

What we can say is that a user interface is a space where we place a mix of different computer based signs, some of them are blends -conceptual integration- based on metaphors but other do not. Other signs are *similes* of other real objects. But the most important characteristic of computer-based signs is that they communicate meaning to the user –as all signs do- and sometimes this meaning is derived from a –or is experienced in a way that supports it, according to Lund and Waterworth- metaphoric projection of image schemata.

The important fact regarding the communicative aspect of signs is which is the sender and which the receiver. Some authors (Andersen, 1997) consider that computers could not be said to posses anything analogous to a human language and consequently could neither be sender nor receiver of communication. Then, in this communication through computer-based signs there must be a third party involved, namely the interface *designer*. Lund and Waterworth state the same idea when they point out that "the traditional interface designer is primarily a communicator of mental models".

This point of view also corresponds to the objectivist conduit metaphor, whereby the sender has a content –maybe a mental model- that he wants to transmit to the receiver, in order for the user –the receiver- to reconstruct the same mental model. It is an idealised scenario where all happens as it was intended. But reality is quite different and, as we know, communication is a quite difficult phenomenon. Even Andersen (1997, p. 333) recognises the issue when he points out that, "a system developer is in much the same situation as other producers of symbolic

material like authors or play writers: they can hope that their intentions will be realized on the opening night, and if they are talented their hope may often come true –but in principle they can never be sure".

Another question about the sender-channel-receiver approach is that, when applied to a literary piece, for example, there are many complex problems without solution. Why - through centuries- some drama like Greek literary pieces have been continuously changing their meaning, so that they may perfectly be adapted to contemporary times? We cannot imagine the Greek author foreseeing all the future avatars of his drama and then including all this content into the literary piece. That is why some French structuralist literary critics (Barthes, Soller and others) of the '70 have elaborated a theoretical approach to consider the text as something autonomous, with a capacity of establishing its own rules and then to be considered in its own terms.

Even if there are profound differences between a Greek drama and a graphical interface, the previous analysis show that we do not need to include a sender –the interface designer- as something essential to understand the communicative function of the interface. It is not a question whether the designer has correctly sent a given mental model but how a computer-based sign facilitates the task of inferring its meaning.

6.1.4 The meaning of computer-based signs

In a general sense, we can say that not only the experientialist designer is primarily a creator of user experiences (Lund and Waterworth, op. cit.) but *any* graphical user interface –whether based on metaphor or not- allow user experiences. Then, the main question would be: what type of experience we are speaking of?

When we consider that a computer-based sign is experienced in a way that supports, for example, metaphoric projection of image schemata, we are indicating that the meaning of that sign may be easily inferred from such a projection. We are in face of a *cognitive process* –metaphoric projection- aimed at deriving the meaning of a computer-based sign, the same way that "we make sense of all the other experiences of our daily life through unconscious projection of bodily image schemata" (Lund and Waterworth, op. cit.). The key element of the previous sentence is to *make sense*, that is, to produce sense or meaning.

This is the *experience* of the user and the experience intended by the experientialist designer. We must take into account that, in addition to propositional comprehension, understanding is an evolving process or activity in which image-schemata (as organising structures) partially order and form our experience and are modified by their embodiment in concrete experiences (Johnson, 1987). We must recognise that the user from his job or workplace might have derived additional experiences. In the cognitive process we have an interaction between the image-schemata that partially order and form our experience but, at the same time, these image-schemata are in turn modified by their embodiment in concrete experiences. The important point, though, is that all these *experiences* are aimed at producing meaning trough a cognitive process.

We propose to speak of a computer sign as being *intuitive* and say that at least one of the meanings of the word *intuition* refers to something that is experienced in a way that supports the metaphorical projection of image schemata. When we put something in a central position we are *intuitively* understanding that pattern in terms of the centre-periphery image schema and

therefore interpreting that the central position means 'important', 'central', 'main', etc. It is interesting to observe how the central position in terms of spatial ordering has the corresponding 'central' figurative position in terms of importance.

When Waterworth (op. cit.) shows us a group of stacks of information references organised along an arc, all the issues he takes into account about the centre-periphery schema or the verticality schema have been used to build a complex computer-based sign with stacks represented as labelled drawers (containers). We can easily infer the meaning of the sign because the schemas upon which it is based indicate that some –central- stacks are more important than peripheral ones; or that highest stacks contain more information references than lower ones. Even if we are offering the user an opportunity to experience such metaphorical projection from image schemas, we need to take into account other aspects as well. These additional issues might be how to show the information contained in the stacks (double-clicking on the stack?), whether is it possible to move a stack to the periphery (dragging it?) and so on, until the computer-based sign is completely designed.

Supposing that the main goal of the interface –like in the previous example- is to offer the user an experience based on image-schemas, this fact by no means eliminates the need to design the interface in terms of computer-based signs that could be blends, similes or icons based on image-schemas, metaphors and projections. Then, we need some guidelines and to define some concepts in order to increase the efficiency when designing computer-based signs in particular and interfaces in general.

6.2 Some previous definitions

We will introduce some previous definitions of the terms, which will be used in the rest of the Chapter:

i) **Conceptual Space:** The set of elements –computer based signs- we can see on a given screen or window. The idea of calling this entity a *space* derives from a frequent spatial metaphor (mental space, blended space, input conceptual space for a blend) that is useful when we need a boundary or container to include a set of concepts. We call it conceptual to indicate the level of abstraction at which we are considering the interface elements that will gradually populate the space: conceptual integration or computer-based signs. Usually, the set of elements of the conceptual space belongs to just one category, for example all the elements of a given window.

ii) **Category:** A set of elements that have been grouped in terms of a given goal, intention or objective for a given activity (i.e., foods to be eaten while on a diet, virtual devices to be used in the HIC application, etc). Observe that we are speaking of non-classical categories (not necessarily determined by a group of shared attributes). A category is composed –in his turn- by a group of other categories, blends or similes.

iii) **Blend:** A user interface element produced by a set of input mental spaces. The new element (the computer-based sign) has partial structure from all input mental spaces and a new, emergent structure, of its own, as well. In a blend, we have –in the general case- structure deriving from several input mental spaces. We have already seen in Chapter 5 the design of the trashcan with a set of input mental spaces bringing their own structure and how new structure emerged in the blend.

iii) **Sign:** An already existent blend used in everyday activities, like traffic, buildings, etc. These signs are frequently used in the user interface, as when using arrow heads as a means of showing directions. In the case of signs, it is evident the work performed to find out what is an essential element in the input mental space to be represented in the sign. So, the intention is to *minimise* the number of elements to be represented from the input mental space of the real life things:

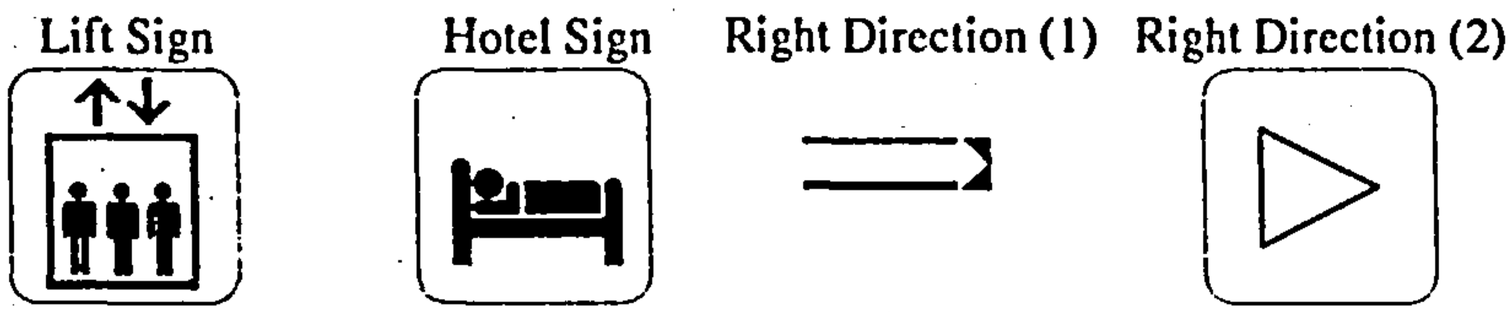


Figure 31 Usual signs and blends derived from usual signs

In the sign for a lift, we can see that there is a group of people into a box –container- and the two arrows indicate an up and down movement respectively: If we have to define a lift, we should very likely use similar words to give its meaning. In the case of a hotel sign, a person on a bed –probably sleeping- might suggest the idea, but it is mainly the context –usually a highway or a road- that determines the precise meaning of the sign. It is clear that the usage of a sign makes its meaning more and more precise.

iv) **Simile:** Sfard (1997, p. 345) explains that analogy, or simile, makes a comparison between two already constructed concepts, even if this comparison does not leave our understanding of either the target or the source unchanged. We propose to use similes for *representing already existent devices* –calculator, CD-player, remote-control device, etc- to be consistent with the definition: comparison between two already constructed concepts. The main characteristic of a simile –as in most analogies- is that there is an isomorphism between functions from both mental spaces, as shown in the following example:

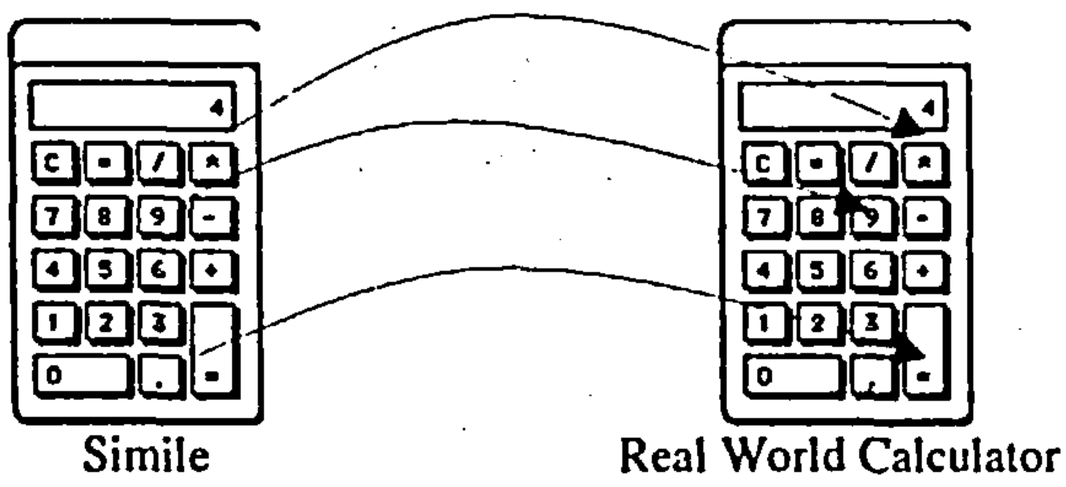


Figure 32 A Calculator Simile

6.3 Structuring the conceptual space

We consider that, on the one hand, the main limitation of experientialism is the high level of abstraction of the concepts it uses and, on the other hand, that the theory of computer

semiotics takes into consideration signs that have been already conceived. As both approaches aim to provide theoretical tools for the analysis of meaning at different levels of abstraction and it appears that both are complementary, we propose to divide the process of development of the interface into phases the same way we proceed for developing software systems in general. This division has the advantage of considering the developing process for the interface in a way that is consistent with the process of developing system software: an *analysis* where it is built a conceptual model and a *design* where the conceptual model is refined to build a logical model. So, we can consider two main phases in addition to the phase of requirements capture:

- Analysis of the interface
- Design of the interface

6.4 Analysis of the Interface

This is the phase where to elaborate a conceptual model of the interface. This conceptual model is composed by a set of conceptual spaces (a conceptual space may sometimes match that of a screen, sometimes that of a window, sometimes that of a blend) each one built on the basis of metaphors, image-schemas, blends, projections and so on.

In order to bring some structure to the conceptual space, we might choose between different frameworks taken from experientialism. A first framework is based on image schemas, as they constitute the form of representation common to perception, memory and semantic meaning. An additional interesting point is that image schemas have an inherent spatial structure, as a recent work about Conceptual Spaces shows (Gärdenfors, 2000). As some researchers like Langacker (1987), Lakoff (1987) and Talmy (1988) have studied the spatial and dynamic structures of image-schemas, Gärdenfors proposes that a conceptual mode based on geometrical and topological representations deserves at least as much attention in cognitive science as other approaches like the symbolic and associationistic. As we have already seen, image schema like *container*, *source-path-goal* or *link* have, in general, a direct geometrical representation associated to them:

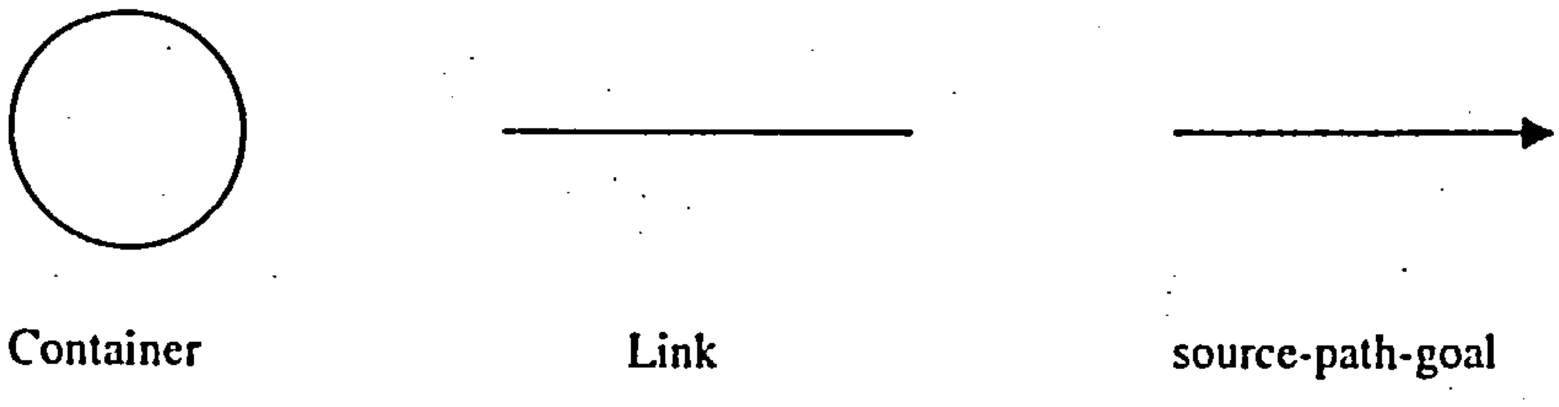


Figure 33 Geometrical representation associated to image-schemas

Gärdenfors proposes a more precise account of what constitutes an image schema, therefore in his theory, image schemas are often just *topological structures*, and a container, for example, is a closed border that separates space into 'inside' and 'outside'. An object such as a cup may count as a container even though it is not physically closed.

Another important fact about image schemas is that they provide a level of representation intermediate between perception and language. This fact is expressed by Johnson, for example, when he points out that image schemas are non-finitary, analogue structures. It is a way of categorising at an intermediate level, **without the need of using linguistic expressions.** That is why a representation based on a *source-path-goal* image schema, for example, is usually considered as *intuitive*, as it suggests by its own a movement from an initial point to an ending point.

It is evident that we can find a propositional *equivalent* of an image-schema, but the important fact is that meaning in natural language begins in "figurative, multivalent patterns that cannot typically be reduced to a set of *literal* concepts and propositions; and (2) the patterns and their connections are embodied and cannot be reduced to a set of *literal concepts and propositions*." (Johnson, 1987, p. 5). We may infer that most of the *intuitive* characteristics of image schematic representations derive from a complex web of nonpropositional schematic structures that emerge from our bodily experience.

The second framework we will use is built on categories (concepts or propositions). It means that in this framework we will use groups of categories to represent on the one hand actions or activities we will perform on entities and, on the other, groups of objects which will be the support of those activities. As could be seen, the first category corresponds to concepts expressed as *verbs* (or verbal phrases) and the second category corresponds to concepts expressed as *nouns* (or nominal phrases). The main difference, when we consider both frameworks, is that in the second framework we use basically a *linguistic* representation, even if concepts are represented as graphical signs. Blends are complex constructions that might also be used as general signs, even if building blends means complex cognitive processes.

It is evident that selecting an activity and an object we act upon, is equivalent to form a sentence with a verb and a noun. These sentences are *basic stories* we construct to represent the interactions with the system. The question is how we choose an adequate group of signs in order to represent both categories: activities and objects.

6.4.1 Image Schema Framework

When structuring the abstract space based on image schema, the main activities could be represented using such image-schemas as path, link, or container. The advantage of this type of representations is that they are intuitive enough as they represent schema we are continuously using in our everyday life. A classical example shows a path between two containers, suggesting an object moving from the source container into the target container:

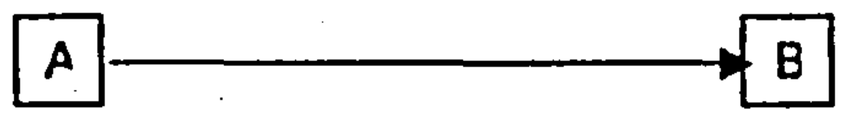


Figure 34 The source-path-goal image-schema

This image schema fits perfectly the representation for a communication between two agents, A and B, in which the first agent sends messages to the second one. The use of

different graphical representations to show the agent B –different fill colours, for example- might mean that the container has or has not already received any message from the source. A different colour would suggest, in such a case, that an event has occurred: the sending of a message.

6.4.2 Frameworks based on Categories of Objects and Activities

When structuring the conceptual space based on categories, both objects and activities could be represented using groups of signs. The advantage of this type of representation is that we get an approach flexible enough to build user stories as a mix of activities applied to objects, with the limitations of some activities not being applicable to some objects in a given context. This representation might have a structure as the following:

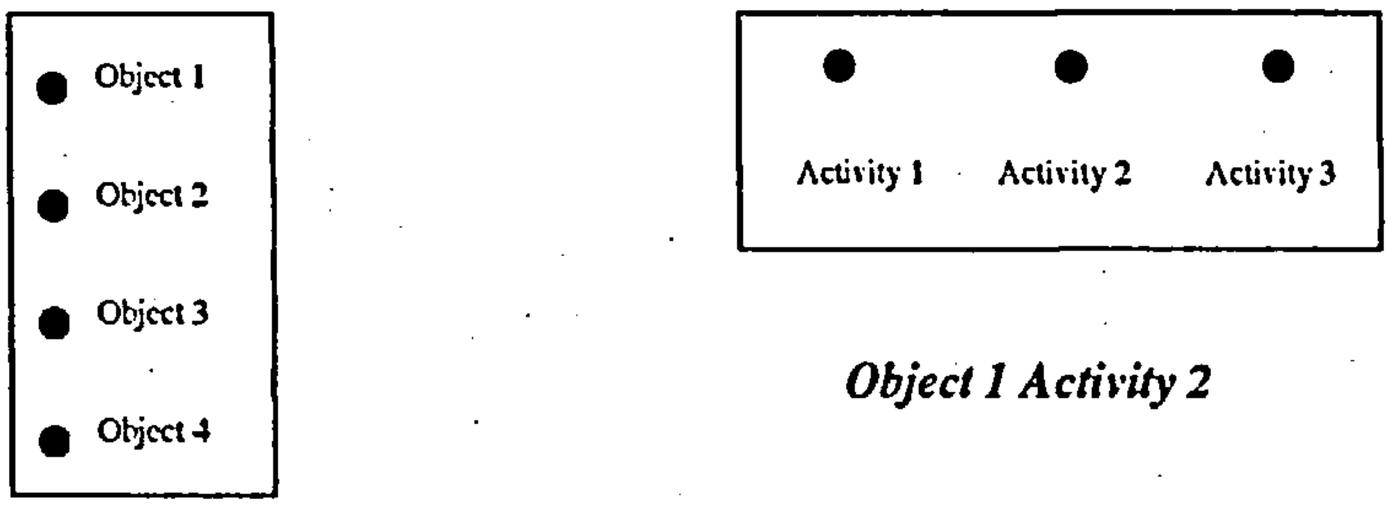


Figure 35 Association of objects and activities

The sentence *Object 1 Activity 2* has been built clicking the signs *Object 1* and *Activity 2*. This linguistic –or propositional- representation may be complemented by means of some blends added to the abstract space. For example, one or more of the objects may be blends, which might have been built using a set of image-schemas and metaphors. The new element –as the classical trash can- play the role of any other object represented in the set of general objects. This type of approach is frequently used in some interfaces (Mac OS and Windows) where it is possible to select the trash can –an object- and then to empty it -an activity- by means of an option of a pull-down menu.

6.5 Design of the Interface

In this phase we propose to use the approach of *computer semiotics* in addition to other approaches that could contribute to a good interface design, such as Cognitive Psychology or studies about visual perception (Gibson, 1986), graphical design (Tufte, 1983; 1997), technical constraints to graphical design (Siegel, 1996; 1997), usability engineering (Nielsen, 1993), standards about interface development (ISO/IEC standards) and so on. Computer semiotics has the advantage of being complementary of the experientialist approach, as both are based upon a semantic perspective taken at a coarse grain (experientialism) and at a fine grain (computer semiotics) point of view. What experientialism and computer semiotics have also in common is their capacity to be applied to the workplace and the organisation in general as we did in Benyon and Imaz (1999) and as Andersen (1997) analysed for the case of computer semiotics.

We have seen in Section 4.8 that computer-based signs have three types of features: handling, permanent and transient. According to these features, Andersen stated a classification of computer-based signs based upon the combination of features possessed by the sign and whether the sign can perform actions with function to features of other signs (1997, p. 216):

Interactive signs
Button (signs)
Actor signs
Controller signs
Object signs
Layout signs
Ghost signs

We will give a brief summary of the classification in order to use it during the design phase.

Interactive signs

An interactive sign has permanent features like e.g. size and shape, and during its lifetime it can change transient properties like e.g. location and colour, this changes being functionally dependent upon its handling features. In most cases it can perform actions that change transient features in other signs.

When using an interactive sign, often the effect on the other sign is not dependent upon the mouse cursor, but upon an intermediary sign manipulated by the cursor. This produces the feeling that the intermediary sign, and not the cursor, is the tool we handle. A typical example is the box in the scrollbar, which can be moved by means of the mouse cursor, and in its turn makes the "page" move in the opposite direction (Fig. 8)

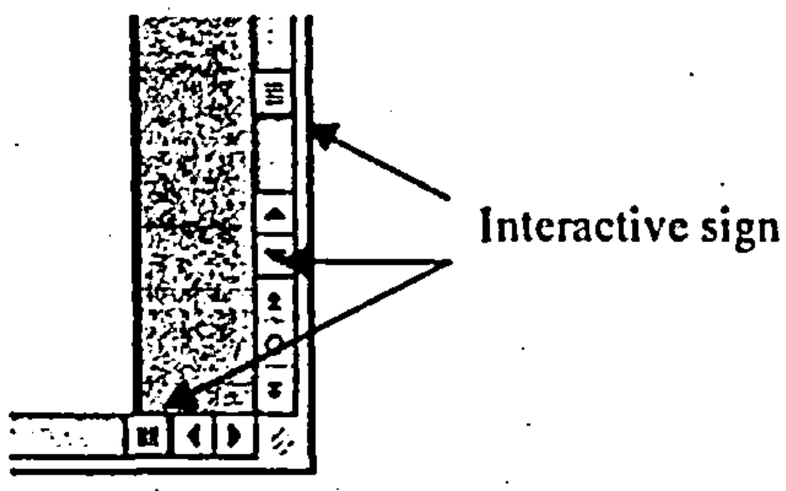


Figure 36 An interactive sign

Button (signs)

A button is an interactive sign in which the transient features are very simple. When clicking a button, we can perceive a short inversion. As these signs have their transient features so simple, they call for a special designation.

Most toolkits have a set of functions activated through a button, as when we use Word:

Set of buttons to Open a File, Save the current file, Print the file and so on

Figure 37 Button signs

Actor signs

These signs are able to change to change position and/or shape on the screen and to modify other signs, but they cannot be influenced directly by the user, although they may adapt their behaviour according to the way the user manipulates his interactive signs. For example, early versions of Macintosh Word (version 3) had an actor to create a Table of Contents. Once clicked the button that sets the actor in motion –the Table of Content Maker- it cannot be handled by the user, even of it can be stopped. The actor has its own permanent and transient features (a clock, the verbal sign –"repaginating"- and a group of changing numbers. Last versions of Word only show the clock as transient features to indicate that it is working and we need to wait until it finishes the current task.

Controller signs

Some signs change other signs although they do not change their own visual appearance. In Word, the rectangle enclosing the document (the "paper") is a controller, since it changes the cursor when it moves across the border.

Object signs

Object signs possess permanent and transient features, but not handling features: they cannot influence other signs, but can themselves be influenced. Usually, an object sign denotes a work object, like in Word the sign representing the document (the "paper"). This object sign can be modified by means of an interactive text cursor whose permanent features include shape (I) and whose transient features include location (it moves as long we introduce characters).

Layout signs

Layout signs lack transient and handling features, and have no function vis-à-vis other signs. They serve as mere decoration and are quite similar to conventional paper based signs. For example, a trademark logo included in a web page is a layout sign.

Ghost signs

Ghost signs are signs that lack both permanent and transient features. As they are not represented by icons or other identifiable graphical element, they cannot be manipulated directly. In some games, there are hidden traps; they cannot be seen, but they cause the protagonist to fall down if he steps on them.

6.5.1 Transforming a blend from analysis to design

A blend may be a complex concept whose structure becomes from various input mental spaces. In the blend we may choose a partial mapping from a given metaphor and additional features from other mental spaces but the blend has even its own emergent structure. The construction of the blend can then be represented as a network of spaces each one contributing with some features.

Passing from analysis to design means transforming the blend in terms of computer-based signs. We may get another complex structure represented by a network of signs, but this new network has nothing to do with the original one, which is a network of mental spaces.

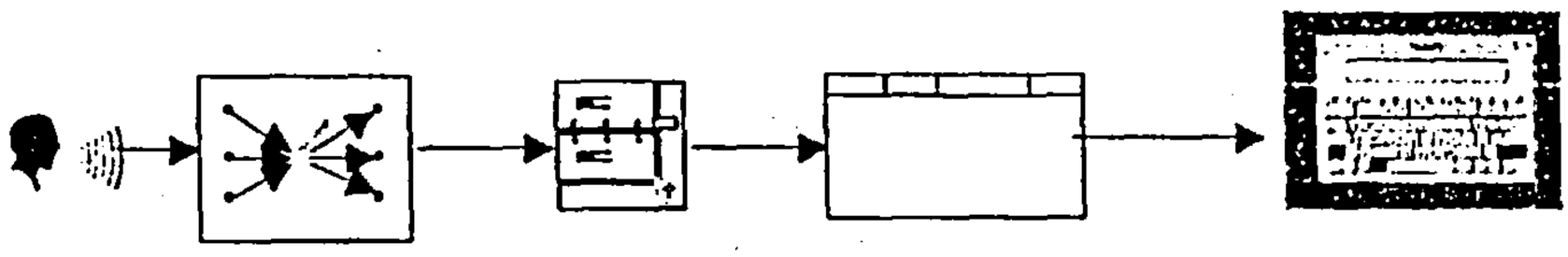


Figure 38 Network of computer-based signs

Once the set of signs have been designed, the next step is to decide where to place them: in the same conceptual space (screen) as a new window, in the same window with another sign, as an isolated signs, and so on. The location of signs in the conceptual space may be assigned using an image-schema in order to give them a more clear meaning. In the case of the message board sign we may place the device containing all input and output messages using the centre-periphery image-schema in order to assign it the central role of such a device.

6.6 The Home Information Centre (HIC)

As explained in Appendix A, the product Home Information Centre (HIC) is a "device for multimodal access to information and services relevant for the everyday life in a family". The project for developing such a device (Flex) will "provide the software for supporting intuitive navigation in and visualisation of information and flexible queries for information, as needed for the realisation of a HIC".

There was an old version of the interface that allowed the access to some of the devices (TV, CD, Hi Fi, etc) by means of a PC, which integrated an equivalent of a remote control.

6.6.1 Analysis of the Main Current Conceptual Space

At the moment of evaluating the Flex User Interface in terms of the Experientialist Approach, the initial Conceptual Space shows elements from two different categories: the Windows 95 Operating System and the HIC Application. It has the following look:

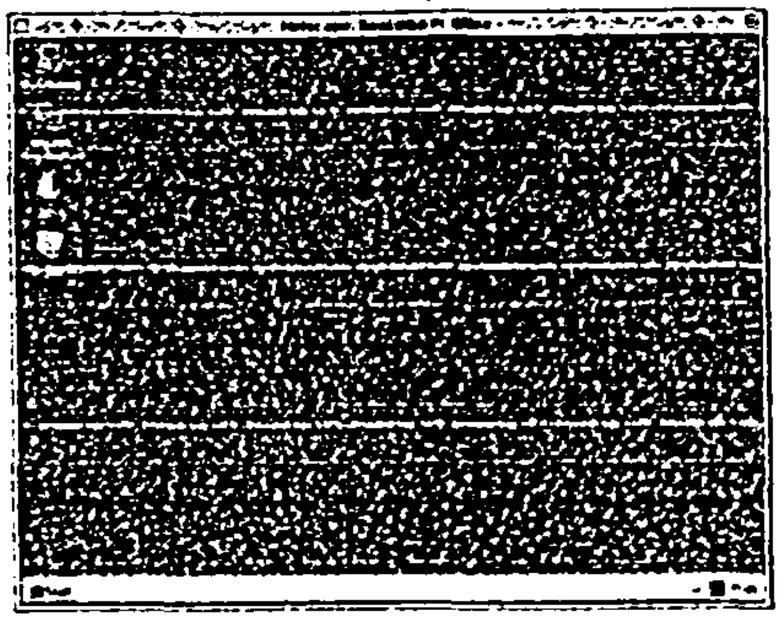


Figure 39 The Window User Interface

When approaching to the right margin –using a feature of the OS itself- appears a Remote Control Simile. The decision of including elements of two different categories -OS Windows 95 and HIC- is in line with the idea of getting an extension of the PC Office: integrating TV and Audio devices. This conceptualisation of the new elements in terms of extension is coherent with the inclusion of the Remote Control Simile as a hidden device, which can appear when needed.

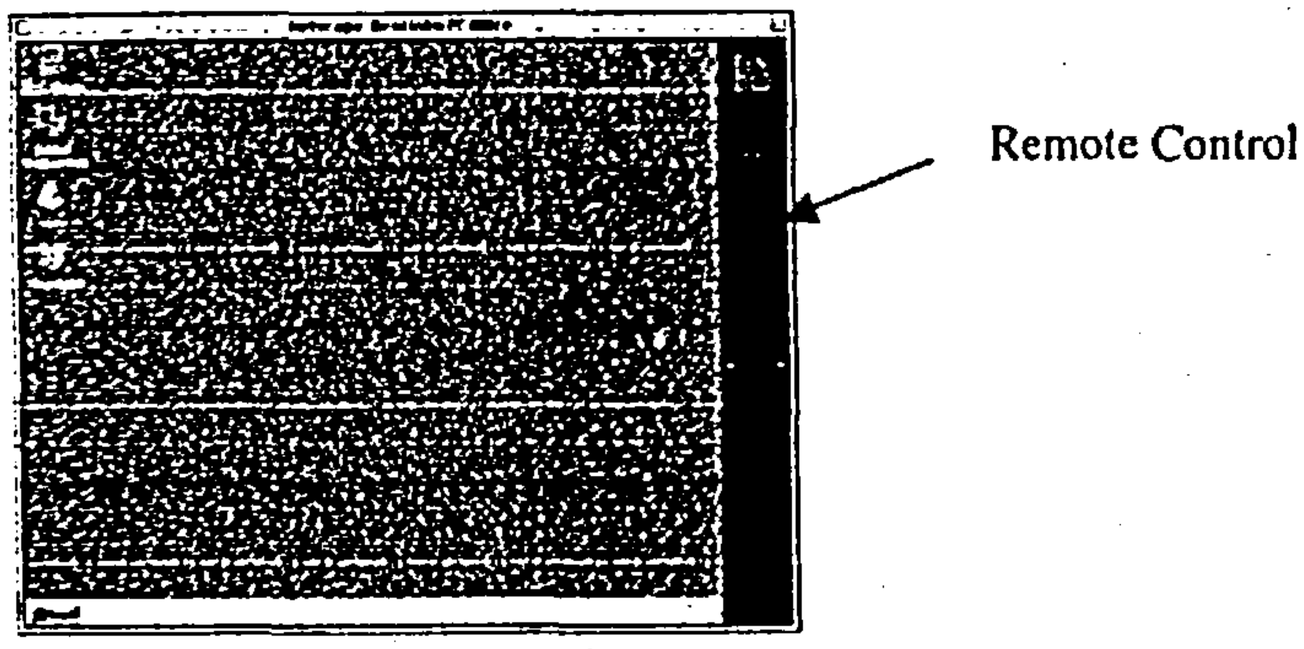


Figure 40 Integration of the Remote Control in the Windows Interface

In this User Interface, once the Remote Control Simile appears, it is possible to control various devices –all those that have been connected to a centralised physical Remote Control- such as TV set, radio or CD reader. The Abstract Space is conceived in terms of a centre-periphery image schema, in which the main category corresponds to the PC Office and the peripheral category to the Remote Control Simile (hidden).

If one wants to see a TV program, one selects the program and begin to see it on a special window of the general abstract space (PC + Remote Control Device).

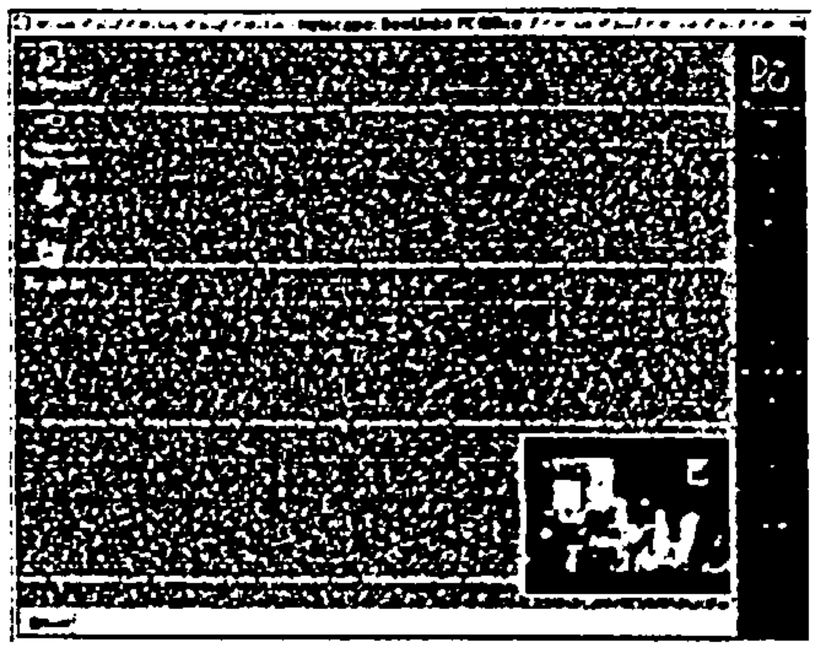


Figure 41 Seeing a TV program in the Windows user interface

6.6.2 New Conceptual Spaces for the HIC

The same way that the initial conceptual space for the integration of TV, radio and CD into the PC Office would have the following aspect:

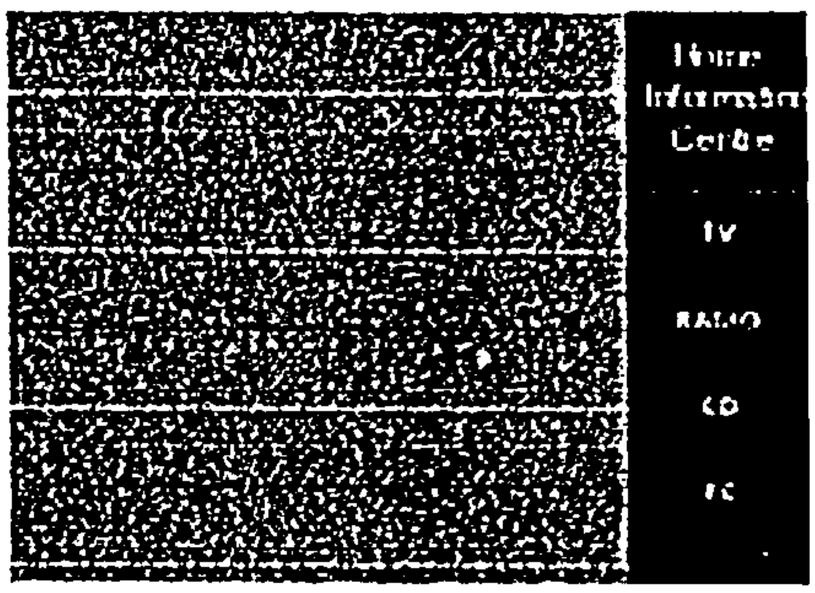


Figure 42 The initial conceptual space of the PC Office

The new device is a larger blend in which there are other features not included in the original integration. These new features contemplate aspects such as content providers, internet, verbal communication and so on. The conceptual space is an abstract representation of the blend: we must include all the features of the device as this initial screen is the *visible* form of the device.

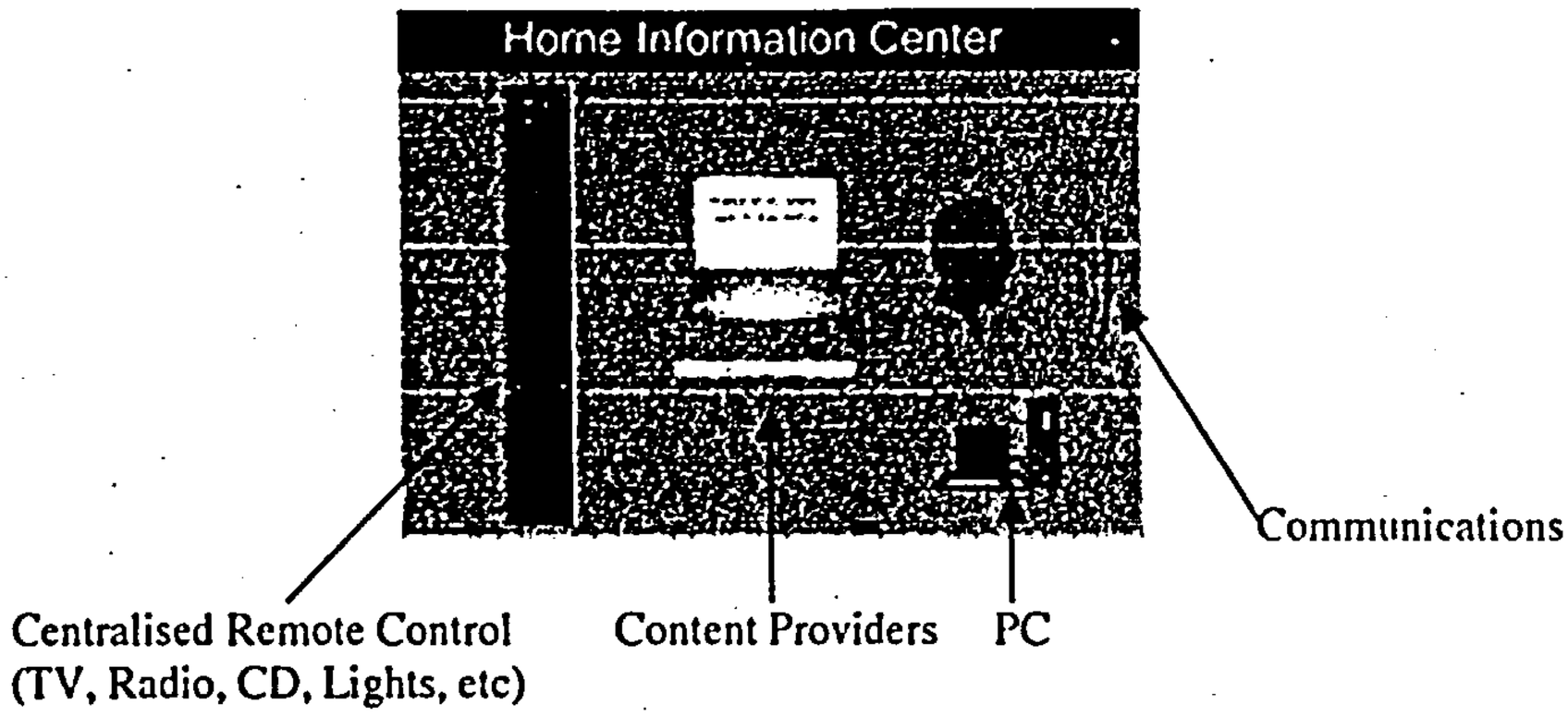


Figure 43 The proposed conceptual space for the HIC

The organisation of this initial conceptual space may be left to the (main) user preferences, so that the image schematic distribution will follow the central interests of the house's users. This initial conceptual space may also show some important facts like urgent messages by showing, for example, the sign of sound waves (Communications sign) in the form of animation or changing the usual colours. Note that all icons used here to represent signs and blends are only provisional, suited to visualise the concepts we are talking about, and they need to be designed and completed in function of additional considerations like colours, form, and suitability to the concepts we are treating. So, they are only used to allow a concrete discussion of ideas and not to propose them as the best candidates to the conceptual space.

6.6.3 Capturing Requirements for the Message Board

After clicking the *Communications* sign in the previous conceptual space, we will be presented a new conceptual space, that of all possible ways of communication between the user and the HIC. In order to analyse the Interface, we will discuss one of the scenarios, that of Message Board (scenario 05) of the Flex Project (Appendix A).

The sequential reading of scenario 5 provides us a list of possible interactions between the user and the HIC. The number in parenthesis refers to the reference in the scenario description:

1) Ask for messages (1)
2) The HIC identifies the user (2)
3) Inform the user that there are new messages (3)
4) Forward messages to another location or user (4)
5) To record verbal message (7)
6) Finishing a verbal message (8)
7) To take a snapshot from the user (9)
8) To attach a snapshot to a message (10)
9) To set a priority to a message (11)
10) To send a message (automatically or asked by the user) (12)
11) To translate a verbal message (13)
12) To acknowledge that a message has been sent (15)

13) To answer a phone call (18)
14) To record a phone message (19)
15) To forward a phone message by e-mail (20)
16) To announce a phone message (23)
17) To phone (24)
18) To write a message (26)
19) To record a verbal message (27)
20) To display a message on the HIC (and/or on the mainroom TV) at a given time (28) for a given period (29)
21) To active the CD-player at a given time (30)
22) To select a CD for the CD-player to play it the next time it is automatically activated (30)

This list may be categorised using a set of concepts (mainly objects of communication and devices): messages, images, internet, TV, CD-player, phone and keyboard. Sometimes, we may use a combination of objects of communication and devices in order to create new categories, as when we treat *phone messages*, for example. Applying these criteria to the above list, we can create the following categories:

Sending and receiving e-mail messages
Receiving (and sending) phone messages
Taking a snapshot (or a video)
To record and listening verbal messages
To control a given device

And, for each category, we can include the corresponding interaction from the list:

1) Sending and receiving e-mail messages
   - Ask for messages
   - Inform the user that there are new messages
   - To write a message
   - To set a priority to a message
   - To send a message (automatically or asked by the user)
   - Forward messages to another location or user
   - To acknowledge that a message has been sent
   - To attach a snapshot to a message
   - To forward a phone message by e-mail

2) Receiving (and sending) phone messages
   - To answer a phone call
   - To record a phone message
   - To announce a phone message
   - To phone

3) Taking a snapshot (or a video)
   - The HIC identifies the user
   - To take a snapshot from the user

4) To record and listening verbal messages
   - To record a verbal message
   - To finish a message

- To display a verbal message on the HIC (and/or on the mainroom TV) at a given time for a given period

5) To process a verbal message
- To translate a verbal message

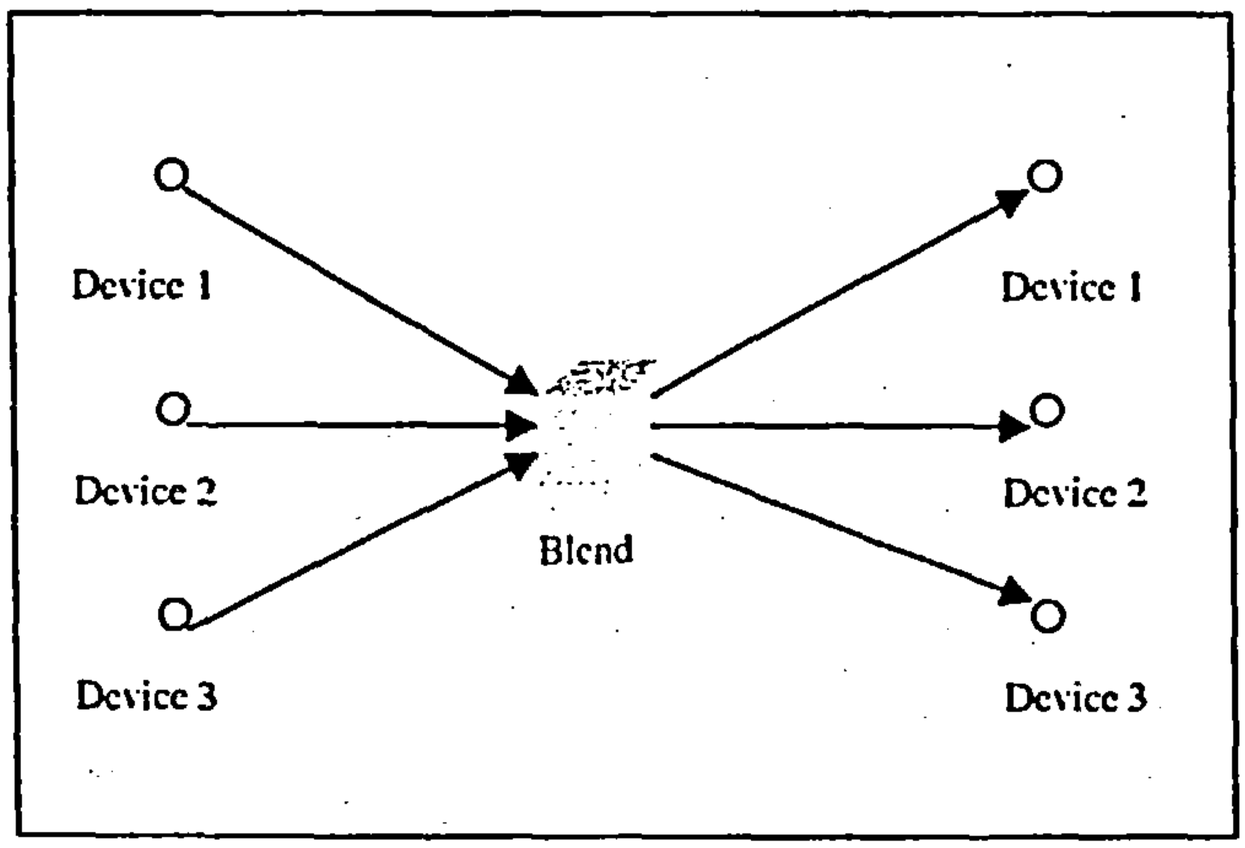
6) To control a given device
- To activate the CD-player at a given time

6.5 Analysing the Message Board Conceptual Space

After having categorised the main interactions we may proceed to an initial solution to the conceptual space of communications (message board) in which we will introduce a set of blends. We will present two main solutions based on the frameworks defined in sections 6.4.1 and 6.4.2: image schematic framework and framework of categories.

6.5.1 Framework based on image schemas

A solution based on image schemas might be based upon three of them: source-path-goal, container and centre-periphery. In this design we may use a set of source-path-goal image schemas to represent different ways a message may come in and a set of similar image schemas to represent ways a message may go out. The second image schema is used as the base to build a blend which is connected to input and output source-path-goal image schemas. We use the third image schema –centre periphery- to place the blend at the centre of the design in order to fix the attention of the user on this design element in which all messages –input and output- are stored.



The message board conceptual space

Figure 44 A first draft of the Message Board built with image-schemas

Selecting the goal of the path may be the indication to send a message to the specific media (e-mail, voice message –recorded-, TV set or CD player)

6.5.2 Analysis of the Blend

In the conceptual space of the message board there is only one (central) blend and some input and output devices. In fact, we must consider that all entities used in the analysis –in this case input and output devices and a container for messages- are, in principle, blends. So, the analysis we are going to apply to the container may also be applied to any input and output device.

We propose to build the blend using a metaphor: the device is a rolodex.

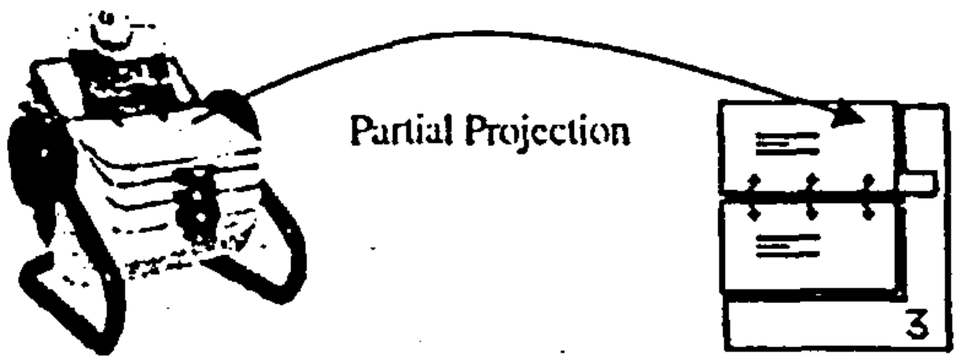


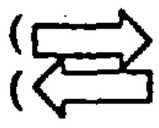
Figure 45 Projecting from one space to another

We need to decide which is the *partial projection* from the metaphor. In this case we can choose for example, the set of cards, the movement from a card to the next (previous), the action of passing from a given category (flap) to the next (previous) one and the search for a set of cards containing a given word.

An additional element of the blend might contain the projection from the input mental space of hand manipulations –going forward and backward to look new cards- and this element may look like this:

Figure 46 Blend of hand manipulation

Note that the blend of *hand manipulations* is a mix of two types of signs (see the next section about signs). On the one hand, we have the right and left arrows ("-> and <-) whose meanings may be deduced using a metaphorical projection from the source-path-goal image schema. The metaphorical projection is implied when we are assimilating a path –a physical movement- to a sequence of cards that pass successively in front of our eyes or to the transition of water from a solid state to a liquid one. Such metaphorical projection is easily applicable to different situations and this is the reason why these arrows are so frequently used as an everyday sign (streets, buildings, tape players, etc) and very frequently only using the head of the arrow. In this context –the Rolodex- the meaning may be immediately inferred as:

()Right arrow – next card and
()Left arrow – previous card

On the other hand we have the top and down arrows, whose meaning is not immediately inferred as in the case of the right and left arrows. A direct projection from the path image schema may infer that they are used to indicate an *up* or *down* directions respectively, but this has no sense in the context of the Rolodex, so we need to assign a new meaning to these arrows.

In a real world Rolodex we use flaps to indicate the change of category. For example, the usual case is an alphabetical ordering of cards so that at the beginning of each category -a set of cards beginning with a given letter- there is a flap with the name of the category, say "B". So, the up arrow could be used to go to the next category and the down arrow to indicate previous category. The blend might look something like this:

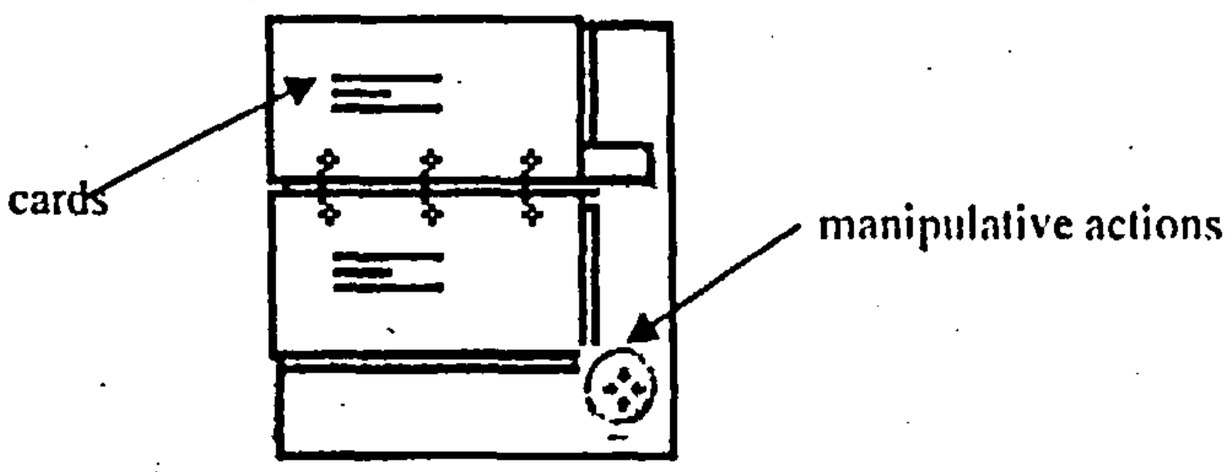


Figure 47 Enlarged Blend: Rolodex with manipulative actions

It is possible to continue developing the blend by including elements from other –or the same- mental spaces. We could imagine that the activity of searching for cards containing specific data might be *included in the blend*; the scope of the search may be the category where we are reading a card or the whole set of cards (all categories). In terms of image-schemas, it is evident that we will need two containers: one to introduce the search criteria –set of words that must be contained in the cards- and other to receive the list of cards that match the search criteria.

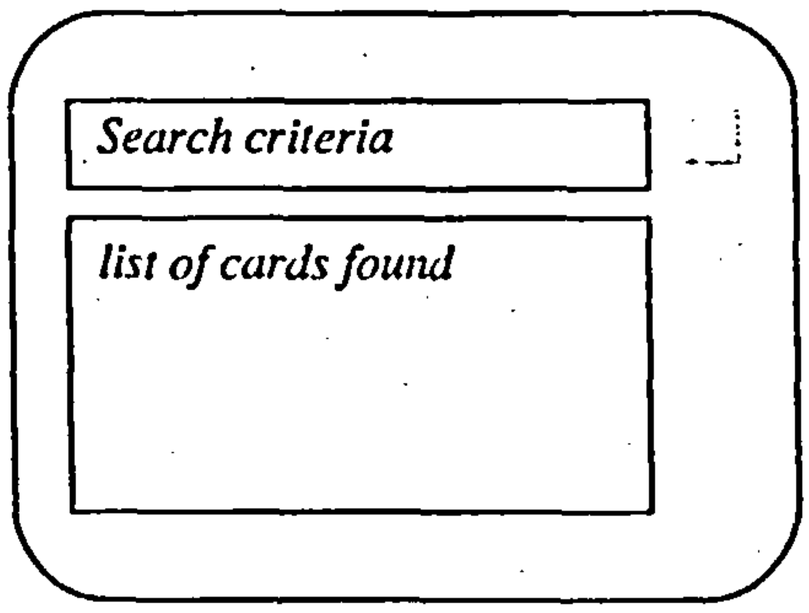


Figure 48 Blend: Rolodex with searching capability

6.5.3 Framework based on categories

The same message board may be structured using signs in order to build propositions as combinations of such categories. In order to indicate that a message has been received through device 1, we need to change a feature of both *receive* and *device 1*. A similar approach would be used to send a message using device 2, for example. In such a case, the user would touch the sign send and device 2. As can be seen, the solution based on the image schematic approach is more economical in terms of user's movements, as touching an output device implies the action of sending. The equivalent in terms of categories would mean using two sets of devices: one for input and the second for output.

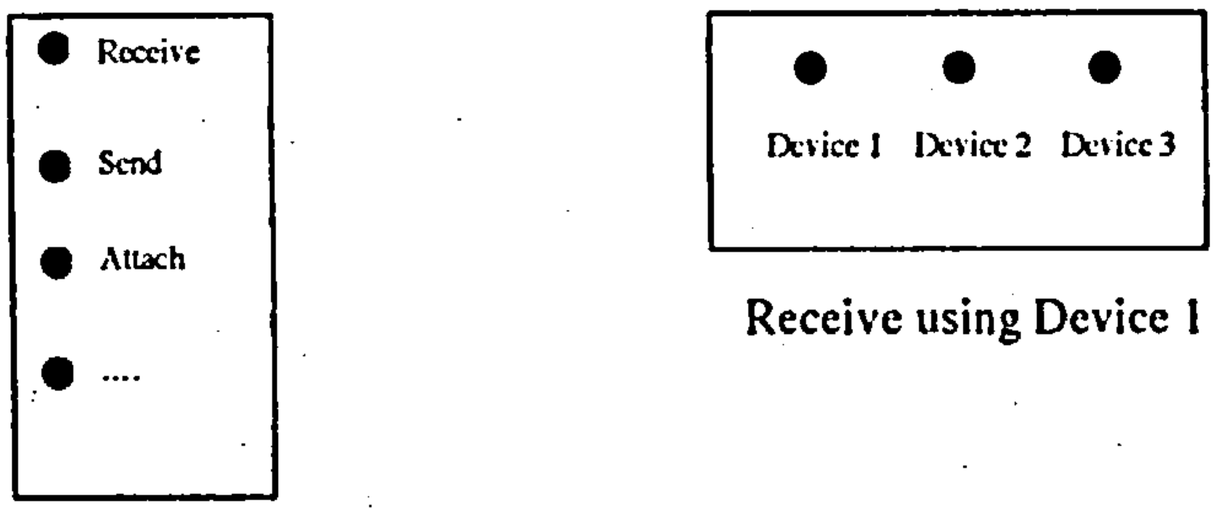


Figure 49 Building sentences by choosing objects

6.6 Designing the Message Board Conceptual Space

6.6.1 Sending and receiving e-mail messages

We are now in condition of specifying the main interactions for this group of activities in terms of interactions with blends and signs:

Inform the user that there are new messages

Changing a transient feature of the sign –for example colour- representing one of the input devices, could indicate the arrival of new messages. This change of feature may occur simultaneously to a group of devices, indicating that messages have been received from different media.

Ask for messages

Touching the sign representing the blend –for example the rolodex-, we get an increased form (zoomed) of it. This transformation -from a reduced form to an increased one of the sign- occurs using another image-schema: the link. The first sign –probably an icon- is associated to a second one, which has an increased complexity and may be built –in turn- as a blend. This new sign allows us to navigate using the group of arrows to select the next (previous) message or to select the next –or previous- category of messages (from e-mails to phone messages or local verbal messages). The increased version of the Rolodex may look like this:

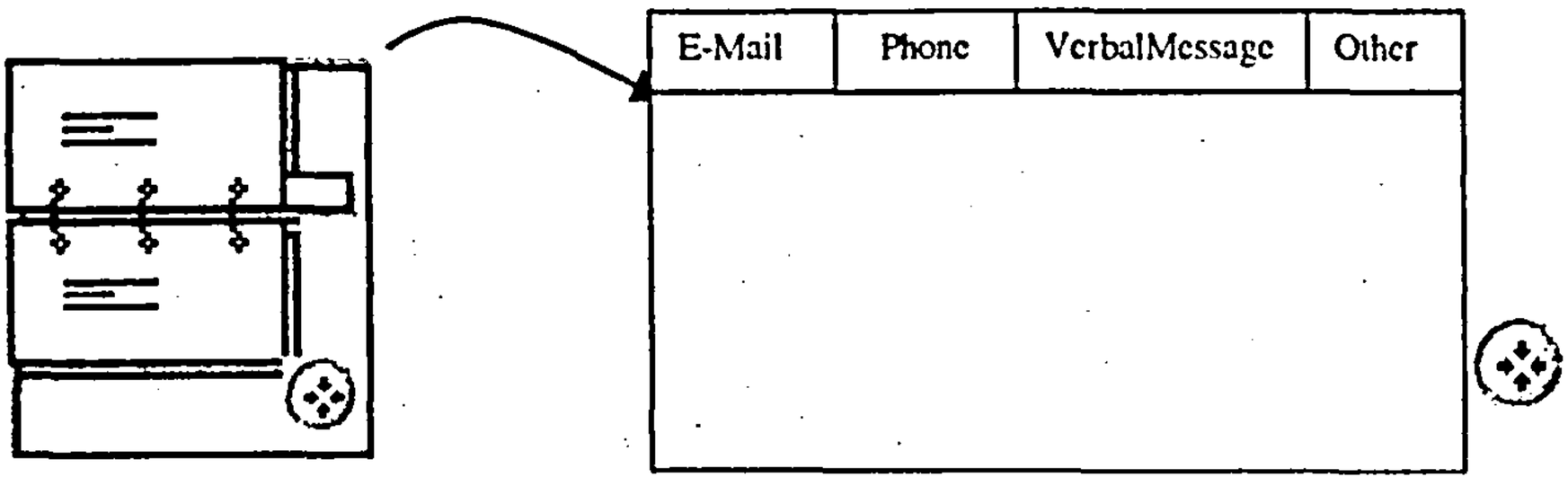


Figure 50 Rolodex showing messages –Flaps indicating categories of messages

We may think of the Rolodex as a single –conceptual- blend (at the analysis level) which has been designed as a network of different signs. During the analysis, we may conceive all the features the blend will have, even if these features will be distributed into different signs. The first sign –the icon- is used to indicated the arrival of messages, while the increased sign is a more complete representation of the blend, allowing the reading, writing and searching of messages.

Touching a given flap (button signs) we access the named category –for instance, verbal messages-, while being in a category, touching the 'right' arrow (control sign, as it changes the content of the *card* sign) we access the next message and so on. The *up* arrow indicates the *next category*, as if we touched the corresponding flap.

To write a message

In order to do that, the user has the choice of using the real keyboard, or a software keyboard –presented as a simile in a new abstract space.

We may consider that this new sign is a part of the same blend –the Rolodex- which appears when touching, for example, a button of the previous sign indicating *Write*. It is possible to expand the sign of the arrows or include an independent button sign.

To set a priority to a message

After writing the message, it would be possible to touch the message window in order to select a given attribute to specify or modify, for example, the priority. After touching the message window, we can get a new space: for example, a pop up window on the keyboard where to set the priority.

After selecting the priority, we can move to a higher level abstract space (*up* arrow) –that of the message board- in order to:

To send a message (automatically or asked by the user)

Sending a message will be performed touching the icon representing the *device* we want to use to send the message.

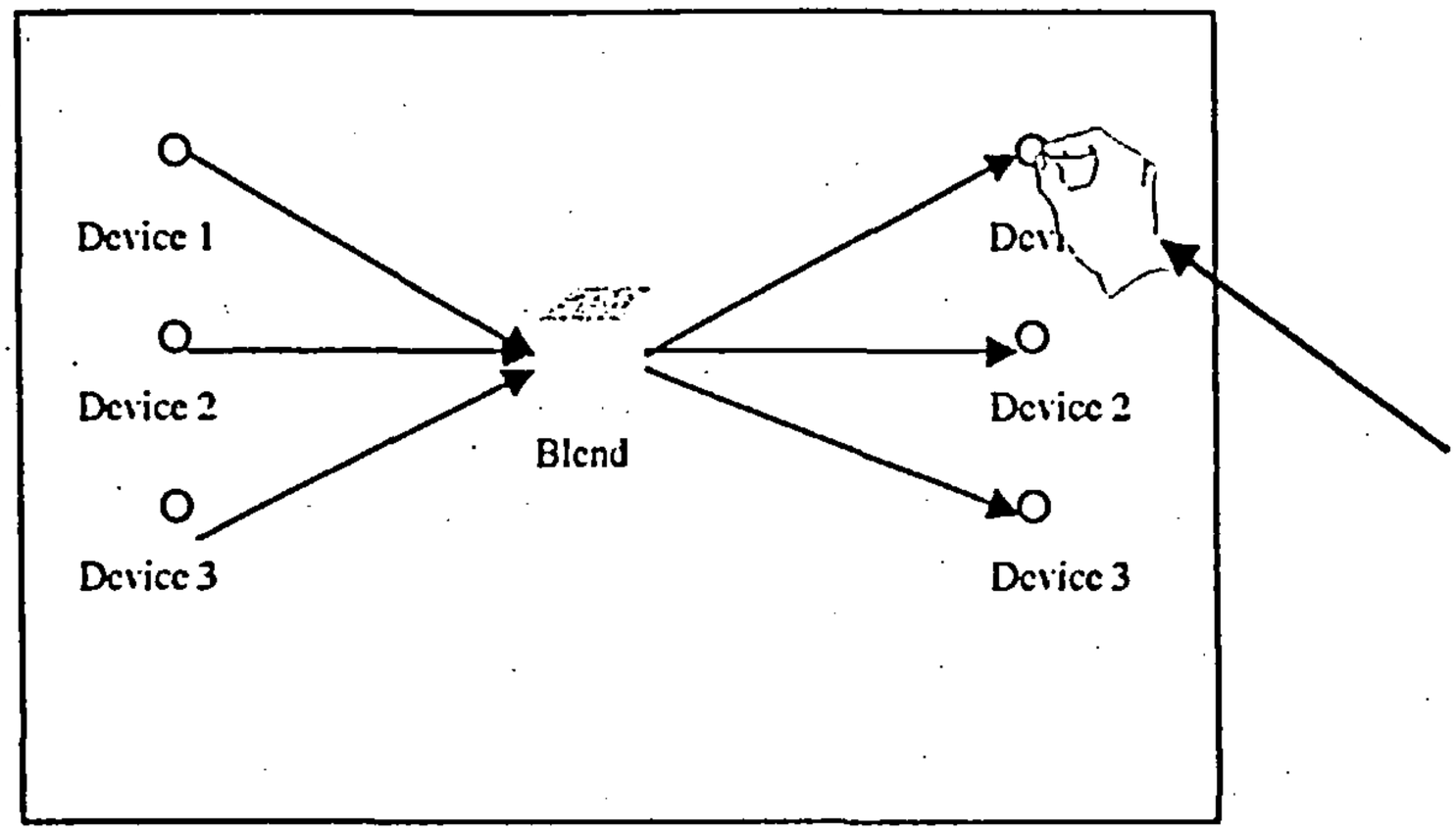


Figure 51 Sending a message by touching the target device

After touching the letter icon, the increased mode of the blend will appear and it will be possible to select –from a list- the user or location to which to want to sent the message. It is possible to have a given location pre-selected in order to send it more easily. For example, after touching the letter icon, we may touch it again in order to use the preselected destination and send the message immediately.

Forward messages to another location or user

This case is identical to the previous one. The only difference is that the message to be sent has been previously received and selected.

To acknowledge that a message has been successfully sent

This information may be presented to the user by means of call-outs, the same way we use to draw comics. So, it is a way of *anthromorphising objects* by projecting vocal capacities, as we usually make with other objects. This information may appear for a period that would has been previously selected and may be changed by using a set of preferences

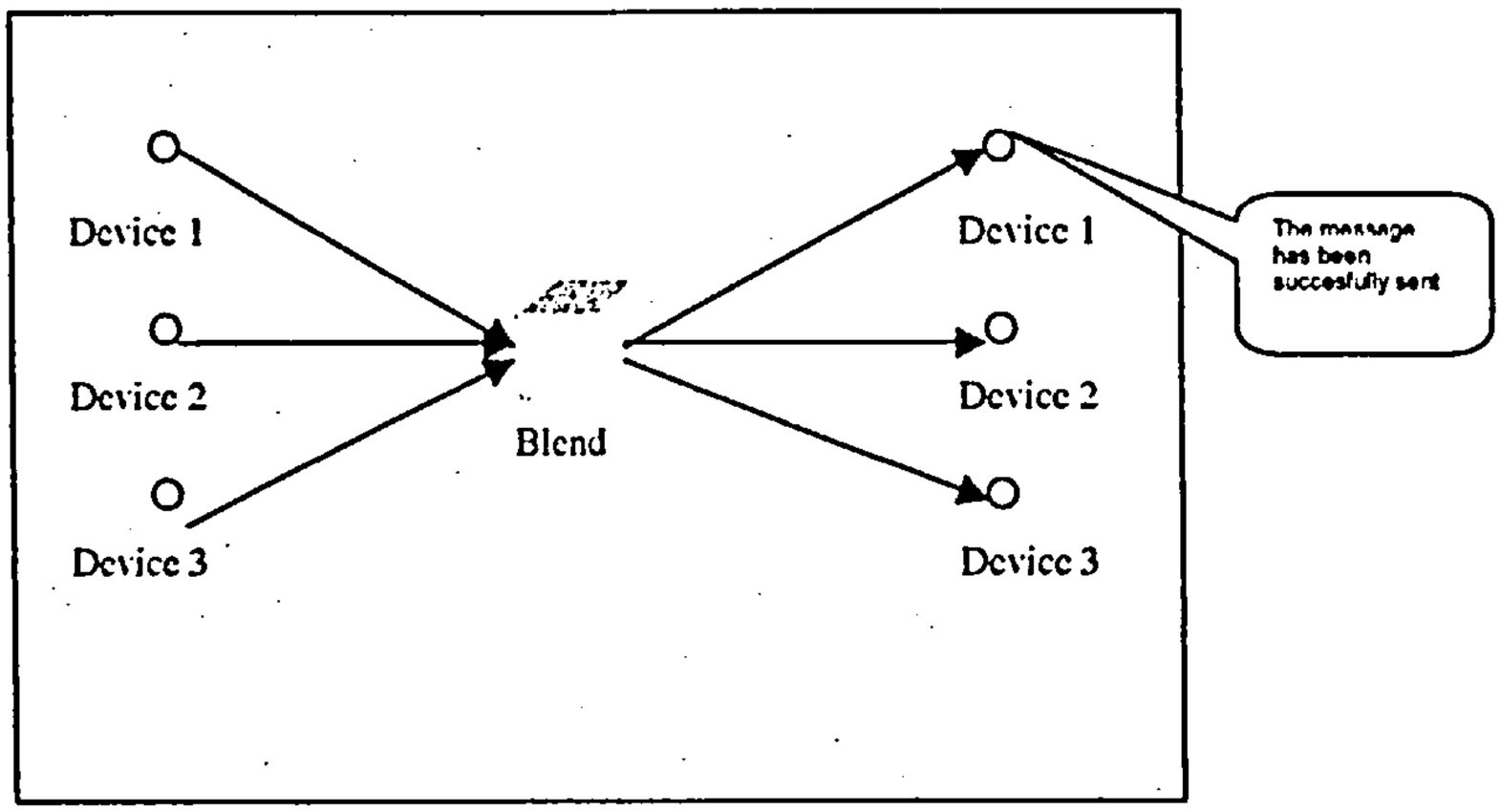


Figure 52 Acknowledging a message has been sent

To attach a snapshot to a message

The snapshot might have been taken automatically by the HIC. To take it manually, it is touching the video camera that we get it. If we have previously selected a message –by touching it in the list of messages in the increased mode of the rolodex blend- the snapshot will automatically attached to it.

These are only some of the features needed by the Message Board, set here to illustrate how the method of Experientialist Design may be applied.

6.7 Conclusions

The main conclusions we may derive from this Chapter is that the pragmatics of Experientialist Design are summed up in two phases -*Analysis* and *Design*- and two approaches –*Designing by Image Schemas* and *Designing by Categories*. In order to face up the task of designing user interfaces we may validate the result by using the *Optimality Principles*, in particular *integration* and *topology* principles.

Analysis

During the analysis phase we produce the conceptual model of the interface, which is composed by a set of conceptual spaces. A conceptual space may sometimes match that of a window or sometimes that of a blend; in general, a window may be composed by a set of blends, so that, the design of a conceptual space has to be done in an iterative an incremental way. In order to structuring a conceptual space, we use metaphors, image-schemas, blends, projections and so on.

Take into account that when we say, in this context, 'a window' we are referring to an already defined *blend* that has been used for a long period and has its own *personality*. Using the UML jargon, a window is a *stereotyped* blend (a blend of a given type and then defined in a meta-classification), and that is why we speak of 'windows and blends'.

Design

In this phase we propose to use *computer semiotics* in addition to other approaches that could contribute to design a good user interface (Cognitive Psychology, usability engineering, studies about visual perception, graphical design, technical constraints to graphical design, standards about interface development (ISO/IEC standards) and so on).

Computer Semiotics and Experientialism are complementary: both are based upon a semantic perspective taken at a coarse grain (experientialism) and at a fine grain (computer semiotics) point of view. What experientialism and computer semiotics have also in common is their capacity to be applied to the workplace and the organisation in general as we did in Benyon and Imaz (1999) and as Andersen (1997) analysed for the case of computer semiotics. In this sense, we can say that Experientialism studies meaning in general, while Computer Semiotics studies meaning of specific entities: computer-based signs.

Designing by Image-Schemas

We may use two ways of structuring the conceptual space: by using image-schemas or by using categories. We employ the general concept of '*designing*' as such frameworks may be used in both phases: analysis and design. An important fact about image schemas is that they provide a level of representation intermediate between perception and language. This fact makes image-schemas very useful designing tools, as they allow categorising at an intermediate level, without the need of using linguistic expressions. That is why a representation based on a *source-path-goal* image schema, for example, is usually considered as *intuitive*, as it suggests by its own a movement from an initial point to an ending point.

Designing by Categories

The second framework we may use is based on categories (concepts or propositions). In this framework we use sets of categories to represent activities we will perform on entities, and objects which will support those activities. The first category corresponds to concepts expressed as *verbs* (or verbal phrases) and the second category corresponds to concepts expressed as *nouns* (or nominal phrases). The main difference with the first approach –designing by image schemas- is that here we use a *linguistic* representation, even if concepts are represented as graphical signs.

The Life-Cycle of a Blend

An important fact is that conceptual integration is not an event that occurs in a given period of time. It may start with some elements of the integration but persist for some months or even years, the way it actually occurred with the desktop conceptual integration. The following paragraph shows how this evolution of the conceptual integration progresses:

"In later systems, such as the Macintosh and Windows, people did strange things with icons, such as using them to represent an application program. The user should never need to operate directly on programs. An icon should represent a document of some kind—you double-click on it, and you are automatically in the right program for dealing with it. That was not what happened, because the later designers were retrofitting the Star's concepts over existing ideas. In the Macintosh, they just missed it; in Windows, they were retrofitting it over DOS." (Winograd, 1996, pp. 23-24)

Optimality Principles

Integration: *The blend must constitute a tightly integrated scene that can be manipulated as a unit. More generally, every space in the blend structure should have integration.*

We have designed a message board as a blend. We consider that it fits the integration principle as the whole (the input messages coming from different devices, the output messages going to other devices and the rolodex) constitute a tightly integrated scene. We can see the states and content of input/output devices, to read/hear/see the incoming messages in the main -central- blend (the rolodex) and so on. Each channel is aimed at transmitting messages, even of different types.

Topology: *For any input space and any element in that space projected into the blend, it is optimal for the relations of the element in the blend to match the relations of its counterpart.*

We may observe, also, that each input space has been projected with all its elements (an input device, a connection and a final device receiving input messages; the central blend -the rolodex- where there is a new message to be sent, a connection and an output device). There are no problems with the elements of the blend, as each one matches the relations of its physical counterpart.

Chapter 7

Designing Interactions through Stories

7.1 Scenarios and Use Cases

A number of techniques have been adopted to facilitate the gathering and representation of contextual information, the most prominent being 'scenarios of use'. Scenarios are narratives describing what people do when engaged in particular activities (Carroll, 1995), although how scenarios are actually used varies widely. Scenarios might be based on in-depth ethnographic studies (Nardi, 1995) or on focused collaborative sessions with clients, users and managers (or what is now called, in general, stakeholders). Other authors (Preece et al., 1994, p. 462) present the difference among a scenario (*a personalized, fictional story with characters, events, products and environments. They help the designer to explore ideas and the ramifications of design decisions in particular, concrete situations*), a snapshot (*a single visual –often cartoon-like- images which capture a significant possible interaction*) and a storyboard (*a sequence of snapshots which focus on the main actions in a possible situation*).

On the other side of the HCI/Software Engineering divide, object-oriented (OO) methods of systems development advocate the adoption of 'Use Cases' (e.g. Jacobson et al, 1992). The Use Case approach has been used by its proponents, at least for the last 20 years, some time before the expanding phenomena of Object-Orientation. In a general sense, Use Cases are not an object-oriented construct. It is a semi-formal way of describing interactions between users and systems, so it is in line with the tradition of *scenarios of use*.

Jacobson describes how use cases should present a 'black box' view of the system. A use case model defines the system's behaviour and is developed alongside, and orthogonal to, an object model. Jacobson employs a graphical representation showing the interaction between entities outside the system ('actors') and the use cases, which are inside the system. When an actor uses the system, the system performs a use case and hence the use cases describe the complete functionality of the system.

At a given stage of the Jacobson's development method (Object-Oriented Software Engineering), the author has constructed an interesting blend: To conceptualise use cases as classes. So doing, he has introduced use cases as another object-oriented construct, and UML was able to use only one modelling element, that of class. The use case is a stereotyped class, thus permitting a uniform and integrated view of object-orientation.

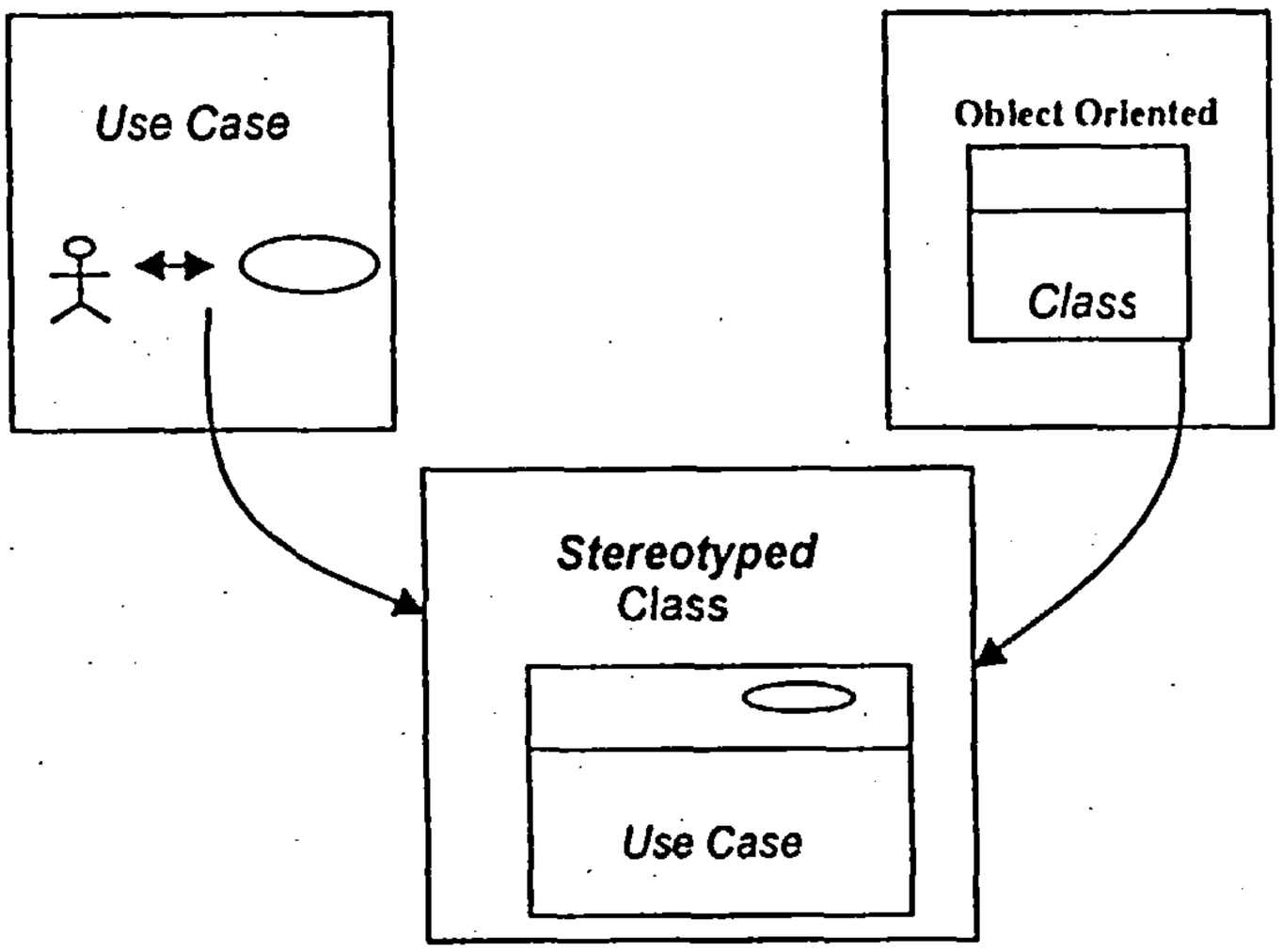


Figure 53 Use Case as a Stereotyped Class

As a Use Case is a bundle of interactions organised by a SOURCE-PATH-GOAL image-schema (that is, a script, one of the Idealised Cognitive Models in Lakoff's terms), in the blend each action corresponds to a method of a class. The method invocation is the user's action to each reaction of the use case. It is evident that there is data encapsulated in the use case as it is capable of maintaining multiple states. This blend receives structure from both input spaces, so we can think of a given instance of a use case (a scenario) as an *object*. As a consequence, when we use a Automatic Teller Machine in order to withdraw some money, this operation is the interaction of two objects: the user (we) and the withdraw operation of the ATM!

Once Use Cases have been conceptualised as classes, they receive in the blend a more formal structure and then they may be given a more formal representation of interactive situations than scenarios. The question remains however: how can we move confidently and consistently between these two representations? Moreover, scenarios are themselves more structured than the original 'user stories' that might be gathered during a work place study or systems analysis. In this Chapter, we describe how such transitions may be accomplished.

The origins of scenarios and use cases are user stories. User stories are examples of storytelling, and storytelling is a constant mental activity of human beings. User stories are the raw material through which we organise idealised cognitive models (ICM), and the ICM derived from a group of stories are Scripts. Script is the common ICM to scenarios and use cases, taking into account that use cases have gone through another conceptual integration and have achieved a higher level of formalisation as classes.

We are constantly constructing small spatial stories and projecting them onto other stories. The capacity to categorise is closely associated to small spatial stories in which we partition the world into objects. This partitioning into objects depends on the typical stories in which they appear: we catch a ball, sit on a chair, take a drink from a glass (Turner, 1996). Categorising is associated to stories, as we do not just recognise a collection of particular objects involved in particular events, but also a set of objects that belong to *categories* in events. In their turn events belong to categories as well.

Thus, human capacities to categorise, to tell and understand stories and to project some stories onto other stories, provide us some cognitive foundations to the concept of requirement. To capture requirements is a way of constructing stories about the workspace, and we can move from such informal stages –user stories- to more formal ones –use cases-.

7.2 Stories

7.2.1 Stories structure

The simplest story structure includes an agent, an action, and an object. In this basic abstract story there is an animate agent that performs a physical action on an object. Moreover, in a general sense, we can consider this basic abstract story to be an *interactional* story: the agent interacts with an object. Conceptualisation of reality derives from interactions with objects: interactions are the basis of stories and of the categorisation of objects involved in such interactions. What experientialism adds to these stories is the fact that our "understanding of social, mental and abstract domains is formed on our understanding of spatial and bodily stories" (Turner, 1996, p. 51).

As there is a human agent in the story, we can observe additional aspects like intentional acts. If we watch someone sitting down in a chair, the sitting involves an agent's (human animate) act with *intentionality* and an object: the chair. Prototypical actors are human beings and many animals: they are capable of self-moving and of sensation. When we observe a small spatial story where an actor -other than ourselves- behaves in certain ways, we immediately project features of animation, intentionality and agency onto it from stories in which we are the actors.

7.2.2 Including User's Stories

Based on this experientialist approach, we can consider requirements as a collection of user's stories. Some authors have proposed that patterns can capture these stories (Beck, 1999). Beck's proposal aims at a 'business to feel ownership of and take responsibility for the care and maintenance of 'the requirements'...I want business to feel free to add new requirements, and add new detail to existing requirements, as development progresses'.

Beck's proposal implies a strong bet on an organisational transformation. Traditionally those who lead and control requirements have been people of the development group. This means that there is a distributed knowledge between developers and users: users are asked about their requirements, requirements are captured, elicited and constructed. This collaborative activity may be transferred *to the business*, as Beck intends, but this is a long-

term project during which the organisation has to accomplish a real transformation in the sense of maturity and self-learning.

Another proponent of patterns (McBreen, 1999) explains the main difference between a User Story and a Use Case: 'A Use Case is very precise and attempts to completely formalise all of the requirements relating to a particular interaction with the system. A User Story gives a specific example of what the results of the interaction should be. A specific, concrete example is more accessible than the more abstract formats that are often used for a Use Case.' He proposes a technique based on a collection of cards on which the stories are captured, each one with its own name.

We agree with McBreen in that a concrete example is more accessible than more abstract formats. What we intend to propose is that both elements are necessary: user stories and use cases. Goguen (1996) speaks of degrees of formality in requirements information, ranging from informal to semi-formal or formal information. Use case is an intermediate stage through others that lead to the last completely formalised stage of programming. We think of software development as a gradual process from informal data (*wet*, in Goguen's terms) to more and more formal levels of representation (*dry*, as opposed to wet). If the goal is to make this process a gradual one, we need semi-formal mechanisms of representation. Moreover, what use cases show is the abstract structure of user's stories, so they are described in everyday language, the same language as the user's. There are no additional dangers, as when we use a specific notation that users could have difficulties in understanding.

7.2.3 What do User's Stories offer?

Beck's proposal about users' stories involves a name describing the story and one or two paragraphs describing the story. The importance of these users' stories is that they capture the intentionality of users and this is essential when trying to understand users' activities and context. Often, the intentionality is richer than the interaction allowed by the system.

There is a new proposal to capture this user's goals as *Business Use Cases*:

"An important issue I have come across with use cases is the difference between what I call business and system use cases. For example, consider the style sheet functionality found in most word processors. With system use cases, you can say that the use cases would include scenarios along the lines of 'define a style,' 'change a style,' and 'move a style from one document to another.' However, all these scenarios reflect things the user is doing with the system rather than *the real goals the user is trying to achieve*. The real business use cases might be described with terms like 'ensure consistent formatting for a document' and 'make one document's format the same as another.'

This dichotomy between the business and system use case is not present in all situations. For example, the process of indexing a document is pretty much the same whichever way you think of it. However, where business goals and system interactions do differ, it is important to be aware of the difference." (Fowler and Scott, 1999). (Italics added)

This proposal is in line with theories such as the Activity Theory, as it is in the context of a workplace activity (business or organisational process is another term for the same concept) where we have to search for goals.

This usually occurs because when designing the interaction with the system we unconsciously include in the blend the technological constraints or usual interface solutions. In design activities we are unconsciously and continually creating blends.

It is useful to capture a degree of informality in such a way that context can be represented in a suitable way. Context maintains its advantages of flexibility and source of new design solutions.

These stories are part of real requirements. Traditional HCI systems design has pointed out the deficiencies of stories: "*as we've noted, stories are not particularly accurate. Stories are like pearls, layers of gloss accreted around an irrigating grain of reality. They exaggerate both the travails and prowess of the teller. Thus, it would be quite unwise to rely on stories as the only source of information*" (Ericksson, 1995, p. 50). In the same line of criticism, other highlight that "*Cognitive scientists have developed many concepts about how to represent the expressive discourse of procedures and tasks. One key concept is that the surface form of discourse is not complete in essential ways. The surface form of most discourse, especially but not only stories, leave out motivations, and details of actions and events.*" (Mack, 1995, p. 379). Even agreeing on the idea that thought is independent of language (at least in part), what may be said is that most of the *deep* meaning of discourse –details of actions and events- may be discovered using language.

More recently this trend has been reversed focusing on the central role stories play in learning and design, as when it is claimed that 'knowledge, then, is experiences and stories, and intelligence is the apt use of experience and the creation and telling of stories' (Schank, 1990). In relation with users, they "can understand [scenarios] as stories about their work practices and assumptions. Software engineers can understand [scenarios] in these terms, as well as more procedural versions that resolve higher-level goals into specific actions and dependencies among actions" (Mack, op. cit.). Other authors, out of the HCI field (Turner, 1996) even consider stories as the basis of language: 'With story, projection, and their powerful combination in parable, we have a cognitive basis from which language can originate' (p. 168).

Our main source of requirements then is the set of stories from our users. What has been proposed -by the OO standard Unified Modelling Language (UML)- is to capture such requirements in a special form of stories: use cases. In reality, use cases are prototypical stories, stories that correspond to prototypical situations, in which anybody (any person in a given role, e.g. a category of actors) must act in the same way. Some authors call them scripts or scenarios, but this last word has been maintained in UML to indicate a given flow of events corresponding to a given situation (or condition).

User's stories are related to a given workplace context and technological set-up. What is important in these stories is the intentionality of the user as such intention gives meaning to a group of otherwise isolated stories. This meaning gives coherence to stories by connecting them to activities and objects. Meaning is never local, it is not a deposit in a concept-container, it is a complex operation of blending, projecting and integrating over multiple spaces (Turner, op. cit.). Meaning spreads over a network of spaces, and this network comprises the context of activities.

As we have already shown, categorisation is done using diverse types of ICM. But an important issue in categorisation is that the relevant notion of 'property' is not something

objectively in the world independent of any being: it is an interactional property. And as interactions are integrated in activities, what defines the category is our structural understanding of the activity. That's why some authors have proposed to study some categories *as things to take on a camping trip, foods not to eat on a diet, clothes to wear in the snow*, and the like. As occurs with games –which have only family resemblances- the only attribute foods have in common is the fact that they must not be eaten when on a diet, so it is a strictly relational attribute that defines such a category, not an intrinsic quality of foods (supposing such an expression –*intrinsic* quality- has a meaning). If we try to see that such category is based on foods not high in calories, we may answer that some others attributes have to be added (nutritional properties, balance of components, etc) and, more importantly, even calories is a relational attribute: they are the consequence of the interaction between our body and a given food.

This means our sets of stories with a given goal are the origin of a potential category. The set of stories are structured by the source-path-goal image-schema, and this image-schema gives unity to a group of stories that otherwise would have been isolated stories.

7.2.4 Use Cases as stories from two viewpoints

Cockburn (1996) has found 18 different definitions of use cases. He argues that there are four main issues that can help to define use cases: Purpose, Contents, Plurality and Structure. He proposes the usual definition of use case (shared by Jacobson) as consisting of the following mapping:

Purpose = requirements
Contents = consistent prose
Plurality = multiple scenarios per use case
Structure = semi-formal

Cockburn points out that 'if the purpose of using use cases is to collect user stories, they may be self-contradicting, informal, and have no structure - and still be useful'. However, this contradicts the definition above (which has also been included in UML) in two main aspects; being written in *consistent prose* and having a *semi-formal* structure.

Use Cases are special cases of user stories. They are generalised stories, not stories corresponding to any individual user but to a category of users. They are useful for understanding the possible interactions between an actor (an occurrence of the category of users) and the system we are going to develop. A Use Case categorises a set of stories belonging to two different sub-sets: one set is defined by the user's viewpoint and the other corresponds to the system viewpoint. As Turner (op. cit.) points out, 'in imagination, we can construct spaces of what we take to be someone else's focus and viewpoint. We can, for example, in imagination, take the spatial viewpoint of one of the actors in the story'. The case is that we, as use case designers, take the viewpoint of one of the two actors in the story of interaction between the user and the system.

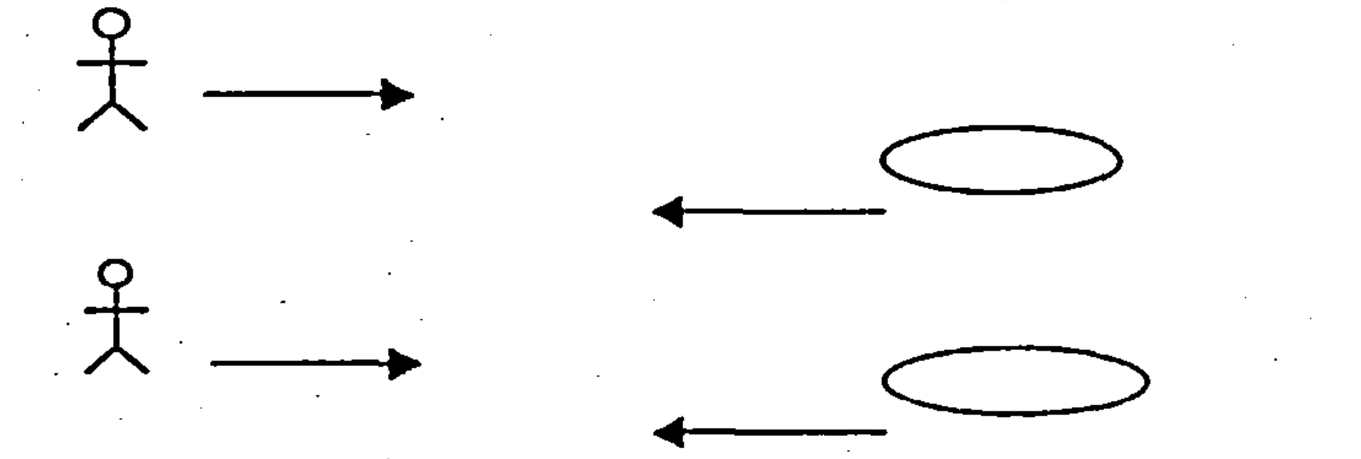


Figure 54 Stories from Different Viewpoints

This reference to a point of view is constant in software design. There is, for example, a differentiation between an 'internal' and an 'external' description of systems that corresponds to the 'what' and 'how' differentiation. The following paragraph by Yourdon, shows how such a difference between inside and outside is constant in systems development: "This may seem like an exercise in semantics, but it's not. If we describe the event from the system's point of view (i.e., from the inside looking out), we might mistakenly identify incoming flows that are not events on their own, but that are required to process some other event. Thus, we always want to describe the events from the environment's point of view (i.e., from the outside looking in)" (Yourdon, 1989, p. 351).

In Use Case descriptions, we are consistently using the point of view of the user. It is the user that will initiate the interaction and will perceive the answer the system will produce. When we use the point of view of the system, it is to realise and refine the interaction in terms of objects that will collaborate in order to obtain the behaviour we need the from system.

Just in the same way that we project agency features on an object, we also project agency features on the system and treat it as another agent acting on objects. Moreover, it is valid to think of use cases as big objects interacting with other objects (users) since the realisation (in UML sense) of a use case is established in terms of a collaboration diagram, that is, a group of objects. In the Use Case, the objects that both agents (the user and the system) act upon are the same.

The interface between both agents is a collection of objects that both the user and system are sharing, so both sub-sets of stories apply to the same shared objects. The shared objects correspond to computer-based signs, which are user interface icons with a set of features. The whole story defines a dialog (conversation) between two actors, user and system, based on the manipulation of shared objects that constitutes the user interface.

7.2.5 Comparing User Stories and Use Cases

We can differentiate use cases from user stories in a number of ways.

First, a user story may have many foci and many viewpoints. In general, these stories may be complex as the changing of viewpoint may imply some future consequences of early actions. In a use case, each sub-set of stories of the use case involves only a spatial and temporal viewpoint (that of the user of the system) and the same synchronous focus: an object of the interface.

The focus may change, but it does so simultaneously for both agents; for a dialog to exist, focus should change in a synchronous way for the user and the system. For example, if the user makes a double-click on an icon, the system has to focus on the same icon which has been double-clicked and perform the operation implied by such a button action. Similarly, when the system shows a window with some question or warning, the user has to focus on such a message contained in the window.

The list of actions in a use case defines a temporal sequence. Sometimes there is a split of a large main story in a set of sub-sets, which correspond to exceptions, groups of reusable stories and so on.

Second, a user story usually contains intentionality and motives. This intention tells us why the user is doing something, it is the meaning of the user action. Use cases, on the other hand, might not say anything in relation to user's intentions. Once the use case concludes, we may be able to infer what was it for. Even if its name gives us an indication of its goal, this might be a sub-goal of the user's intentions. This is the differentiation between business and system use cases.

Third, user's stories may be an individual story standing for a general one. It may contain some design ideas for the designer in addition to desired requirements. The use case is a general, skeletal story: all members of the same category will act the same way. There are no surprises: even exceptions are already included in the stories. These stories tell us how things happen, independently of user's intentions or motives.

7.3 An Example

In order to illustrate how we can better understand user's stories through the concepts of experientialism and in order to illustrate the differences between user's stories and use cases, we provide the following example (taken from (Jeffries, 1999)). It is typical of the sort of requirements that a company may have for a payroll system.

"SPLIT COLA: When the COLA rate changes in the middle of the B/W Pay Period, we will want to pay the 1st week of the pay period at the OLD COLA rate and the 2nd week of the Pay Period at the NEW COLA rate. Should occur automatically based on system design".
For the OT, we will run a m/frame program that will pay or calc the COLA on the 2nd week of OT. The plant currently retransmits the hours data for the 2nd week exclusively so that we can calc COLA. This will come into the Model as a "2144" COLA Gross Pay Adjustment. Create RM Boundary and Place in DE Ent Excess COLA BIN"

This user story has parts that are obscure (the workplace context would give us the elements to have a better understanding of the meaning) but it has been intentionally used because it is a *real* story and can be publicly accessed via the web.

7.3.1 As a User Story

In this example we can see that some issues are general ones (related to the workplace context) and other are specific to certain conceptual spaces.

7.3.1.1 Workplace context categories

These issues are things such as abbreviations (B/W, meaning *biweekly-*, COLA, OT ('overtime'), etc), local artifacts ("2144", DE Ent Excess COLA BIN, RM Boundary, m/frame program, etc), and so on. There is an ongoing process of unfolding the meaning of such expressions, as some of them may be obscure even for co-workers of the employee using the terms/expression.

7.3.1.2 Conceptual spaces

The main conceptual space is that of paying (payroll) which is a good example of blend. This blend could be considered as a classical business rule with two basic stories: "we want to pay the 1st week of the pay period at the OLD COLA rate" and "the 2nd week of the Pay Period at the NEW COLA". The original blend comes from two spaces: the space of paying a period and the space of rates, with a generic space of "paying".

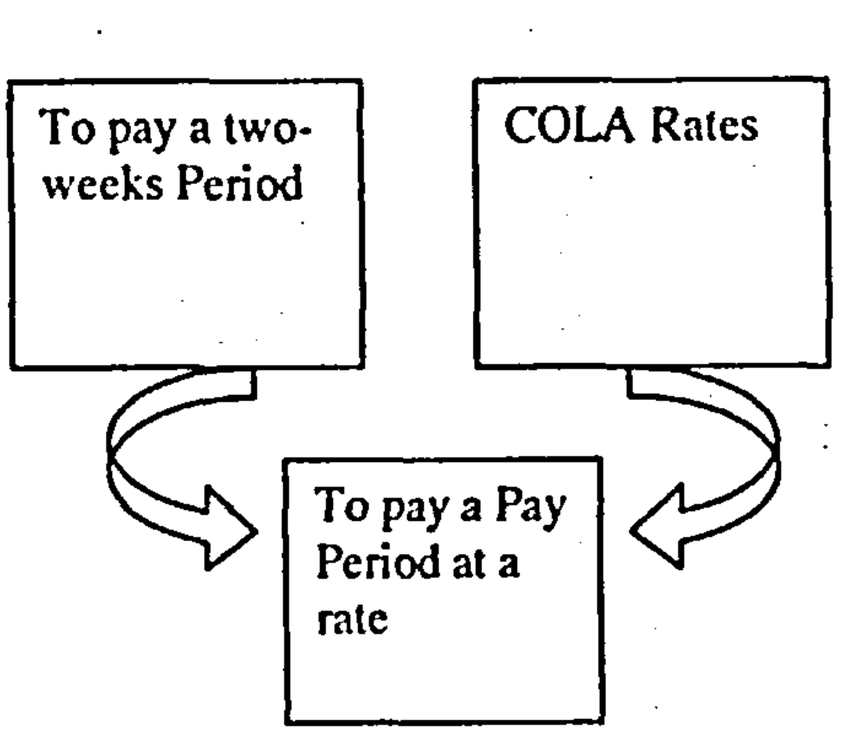


Figure 55 The Generic Space of Paying

It is interesting to observe how the split of one of the input spaces determines the split in the second input space:

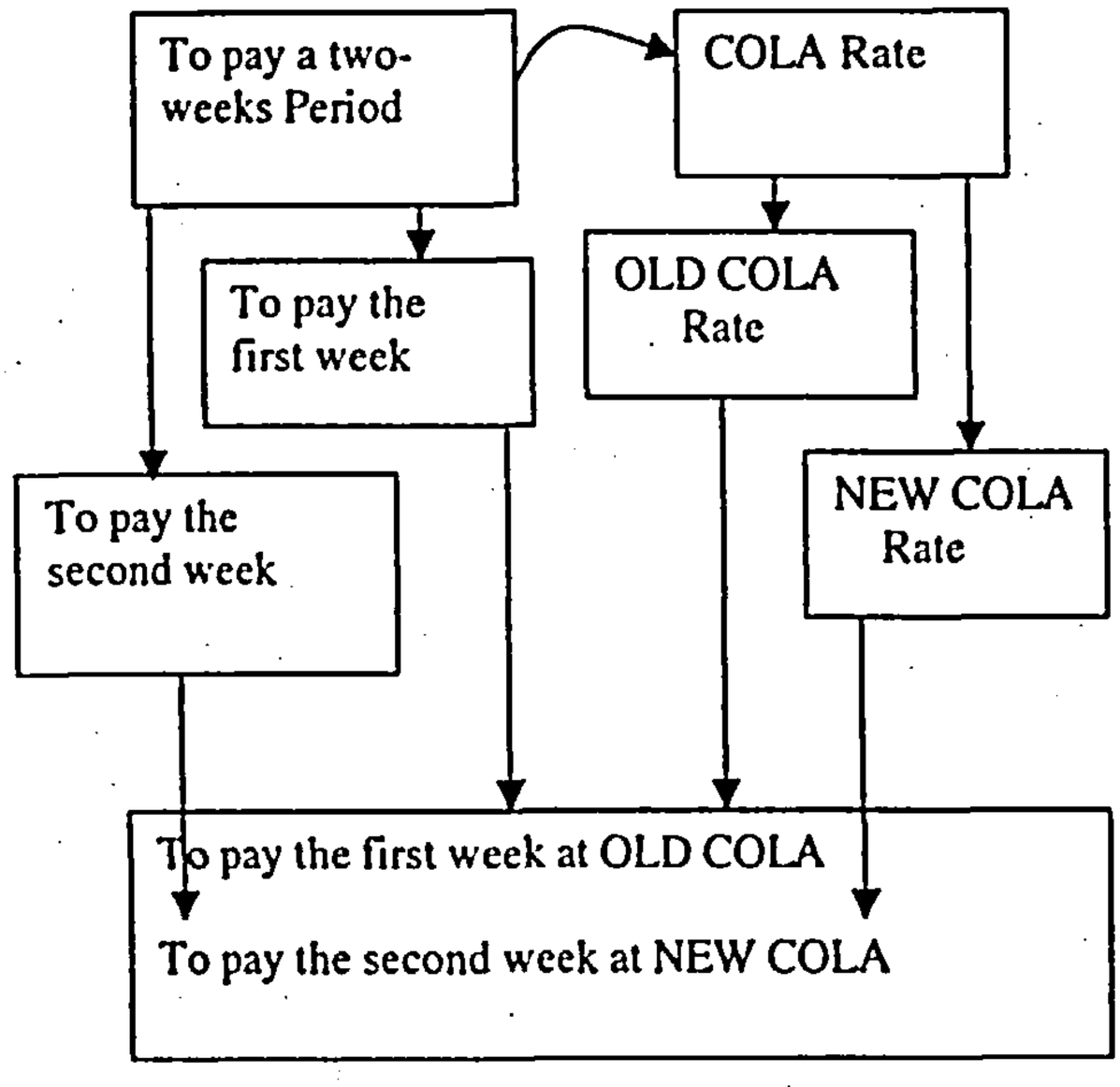


Figure 56 Splitting spaces

The user story originates as a blend in which one of the input spaces is that of rates. Splitting this input space determines a split in the blend. The business rule to apply is then to pay a first week at the old rate and the second week at the new rate. It would have been possible to split the blend differently (for example, calculate the days until the rate change occurred, or calculate a mean rate for two weeks, and so on).

7.3.1.3 Viewpoints

In the user story we witness viewpoint changes. In one case, from the space of paying (we want to pay the 1st week of the pay period at the OLD COLA rate") to that of business processes ("we will run a m/frame program that will pay or calc the COLA on the 2nd week of OT" or "This will come into the Model as a '2144' COLA Gross Pay Adjustment"). In another to the space of communication (or transmission) between geographical sites ("The plant currently retransmits the hours data for the 2nd week exclusively").

There is also some explicit intentionality ("we will want to pay" or "Should occur automatically based on system design") and other intentions expressed in terms of projections from one space (business process) to another (software applications) in the basic story of "Should occur automatically based on system design". In this last statement, intentionality is expressed as a change of viewpoint. The change of viewpoint can go from a current system ("the plant currently retransmits...") to a future one ("should occur automatically...").

7.3.2 As a Use Case

In the user story above we have a core story that we have to transform into a use case. The central story is the blend that could appear in an already specified main use case: Biweekly Payroll. The story has to be translated in terms of such Use Case, taking into account that in the user story there is some additional information that will not appear in the use case.

The translation has to maintain only two viewpoints, that of the user and that of the system, so the remainder of the viewpoints consequently have to be adapted. For example, when the user story points out that 'should occur automatically', we need to observe whether this issue implies some additional interaction to be included in the use case.

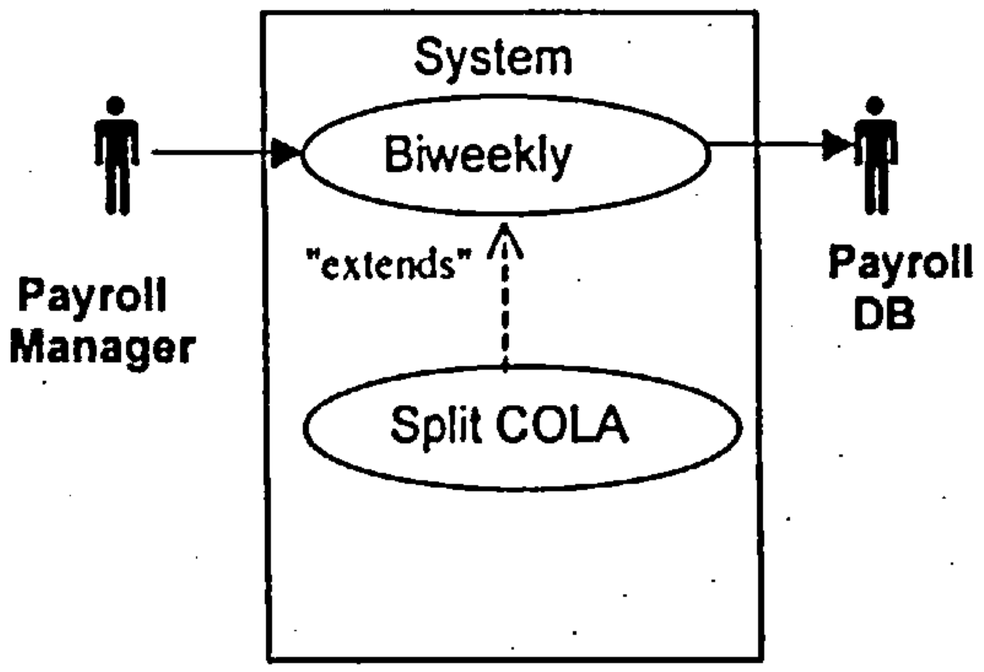


Figure 57 The User Story Transformed in a New Use Case

In the use case Biweekly Payroll we will have a story such as;

1. The use case starts when an actor (Payroll Manager) requests to pay employees
2. The system requests the database to find all employees (or those marked as biweekly ones)
3. For each biweekly paid employee
   If there has been a COLA rate change during the period then Split COLA

And the Split COLA extension will contain the core user story:

If there has been a COLA rate change during the biweekly period then Split COLA

5 Calculate pay of the 1st week of the pay period at the OLD COLA rate
6 Calculate pay of the 2nd week of the pay period at the NEW COLA rate

The user story may be a whole use case, a partial use case (a 'uses' or extend use case, as in the present example), or some rules to be applied to a use case (new calculations). So the concept of use case has an integral conceptual entity that user's stories do not.

Another question concerns the convenience of having two types of stories: why not use user's stories directly to implement the system? This issue refers to another raised at the beginning of this chapter being the need for a gradual transformation from informal to semi-

formal information to get usable models in implementation. We consider the use of Use Cases as quite convenient since this intermediate semi-formal information determines a better understanding between the community of users and that of developers. But the main argument is that Use Cases are organised by scripts, that is source-path-goal image-schemas and that image-schematic structure is projected to the use case. This is the reason why OMG's UML has proposed that "a pragmatic rule of use when defining use cases is that each use case should yield some kind of observable result of value to (at least) one of its actors. This ensures that the use cases are complete specifications and not just fragments " (OMG, 1999).

Some kind of observable result of value is, in terms of image-schemas, a goal that gives meaning to the whole structure.

A user's story has to be integrated within a larger structure that gives a meaning to it in the context of the system. User's stories may give the opportunity for *a business to feel ownership of and take responsibility for the care and maintenance of 'the requirements*, yet the question of getting a coherent method of specifying requirements remains.

Our experience is that semi-formal notation, when learned by users, compensates for the effort of training them in the new diagrams as the communication increases and users may offer new design issues that might otherwise have been overlooked.

7.4 Objects as blends

In a previous article we suggested (Imaz, 1995) that the paradigm shift in software engineering from data-centred to object-oriented methods could be understood by recognising that objects are based on a new underlying metaphor: OBJECTS ARE PEOPLE. There are various considerations about such a metaphor, the first one being the language used to refer to objects: there are responsibilities, collaborations, messages, behaviour and so on. In the design process, and during the conceptual stage, the most general features we can attribute to objects are responsibilities, which are to be refined in terms of operations and eventually as methods.

Another aspect is the use of some methods like CRC (Classes, Responsibilities and Collaborations), which is a kind of object simulation where people assign responsibilities and collaborations to each other and then play the roles of each of the classes involved in the system to be developed.

What is implied in the metaphor is the projected *agency* capacity of objects. Even the name (object) is tricky: We think of *objects* as completely passive entities, without any minimal capacity of performing actions or even behaviour. But real computing objects are entities in which we have projected some of our own capacities in order for them to replace –at least partially- ourselves in a workplace. As most of objects –especially business objects- have the name of entities we interact with in the context of workplace, it is easy to project such a self-animation capacity on them. Maybe *object* is a misleading name and *agent* would avoid some misinterpretations.

7.4.1 Events as movers or manipulators

Turner (1996, p. 46) provides a large amount of examples of everyday expressions in which we observe the projection on events of some capacities of moving or manipulating. Among the

examples of EVENTS ARE MOVERS, there are some that are well known as they are frequently used:

This recession is an *opponent* whose *progress* we cannot stop.

Time *marches* on.

The recession *crept up* on California and delivered an unexpected wallop.

The examples of EVENTS ARE MANIPULATORS –closer to our computing objects, since much of objects' capacities are manipulations- include:

The drought is *strangling* us.

The bad weather this season has *picked our pockets*.

The recession is *spinning us around*.

It is evident that we consider objects as manipulators, taking into account that manipulation means –in its Latin origin- to take or operate with *the hands*.

7.4.2 Objects as blends of stories

The following step was to think of objects as (conceptual) blends from two or more input spaces (Benyon and Imaz, 1999). Now we can streamline this idea by proposing that objects are designed on a blend of stories.

We have said that when we experience a story we project the features of anima and agency on other actors. In a workplace like a bank we can have different processes represented by stories. One possible story is that of a client going to the bank office to withdraw money from his/her account. It is a typical interaction story in which the client asks the human teller to withdraw an amount of money. Other similar stories involving the account could be:

To deposit an amount of money into the account

To print the account balance

We can imagine other sets of basic stories where the object is the same: in the case above there are three stories about the client account. These basic stories may be in different sequences of stories, corresponding to different activities and actors but the same object appears in all these stories as being manipulated by different actors.

If we project all these physical actions on the same object in such a way that the object becomes self-moving or animated, we have a blend from different spaces and at the same time we project the agency from the set of all actors on the same object:

So, we can think of objects as a blend from different basic stories in which the object is the same even if the actors are different and from which we project the features of agency on the blend: on the object itself. A workplace process is a group of activities whose descriptions are basic stories with different actors and objects. Objects –here used as real workplace items- are

manipulated by different actors and undergo multiple transformations as a consequence of such manipulations.

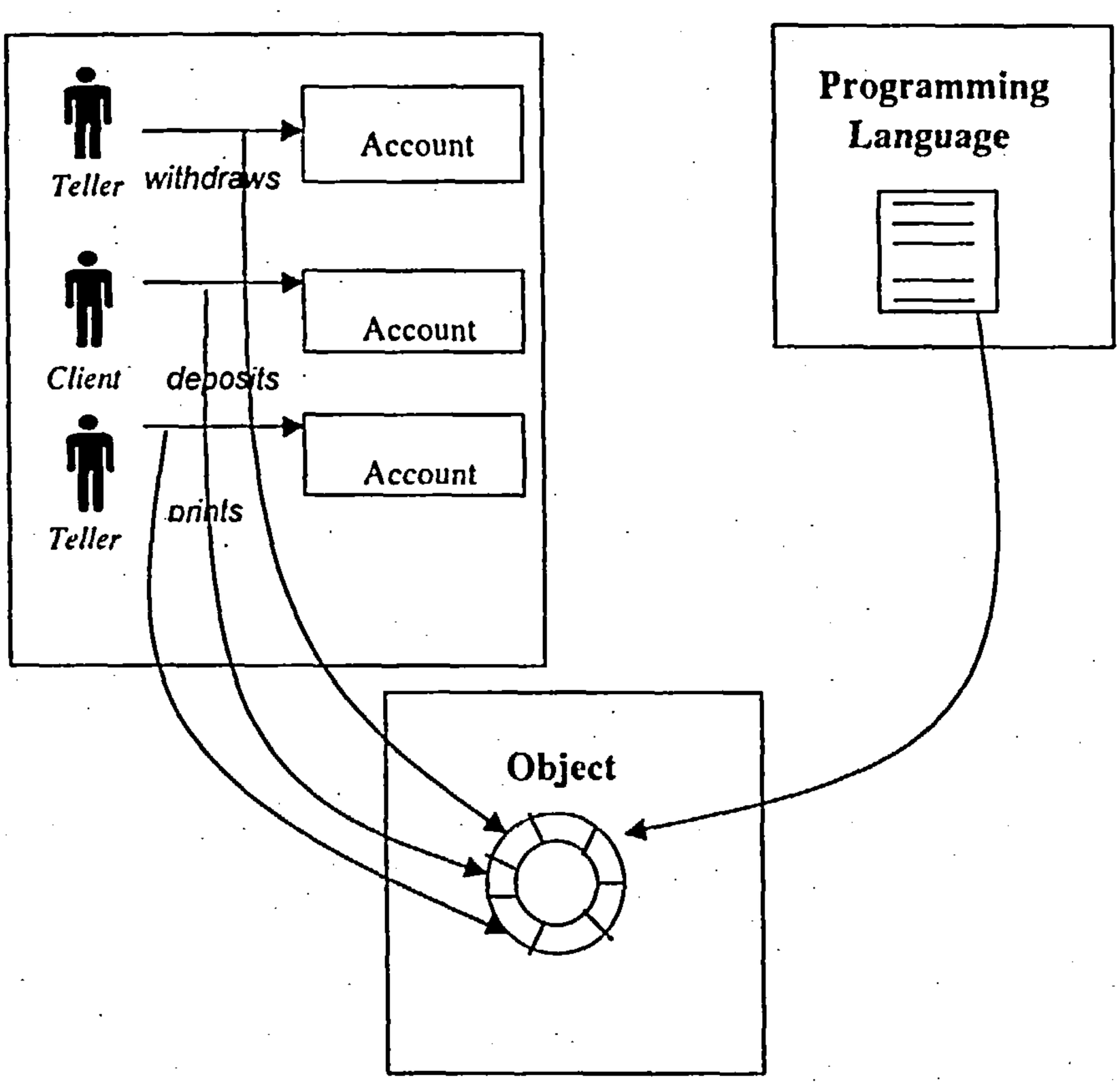


Figure 58 Objects as Blends

We then have a new character, an *object* (in terms of OO terminology), which is a blend from several possible spaces (supposing we unpack the workplace activity into many steps). The projection from the input spaces is selective: we have selected only those stories in which the same workplace object appears (the client account). It is also partial, as we have not included all elements from input stories. The list of stories in which an account may be involved includes:

Creating an account

Introducing account data

Modifying account data

Printing account data

Saving account data into a secure location

Transferring money from/to the account

Deleting an account

The blend develops an emergent structure not provided by the inputs: we can imagine new stories in the blend. The emergent structure is a consequence of interactions the new object will have with another in the new *society of objects*. For example, aside from the manipulative capacity, the object has new features such as the capacity to tell which of the operations it is able to perform by means of *introspection* (another human capacity which has been projected onto objects).

But the features an object receives from the workplace are *not all the possible* features it includes. In the blend there are other features like the kind of object produced by the fact that objects have a given function and this function determines a taxonomy. For example, the stereotypes proposed by UML dividing objects into *boundary*, *entity* and *control* objects are an emergent structure as they represent different kinds of features projected from input spaces.

7.4.3 Kinds of objects

Jacobson (1992) has established a meta-classification of objects in three categories, which has been maintained in UML (OGM, 1995).

7.4.3.1 Boundary objects

These are specialised objects addressed to facilitate the interaction between users and the rest of objects. They are mainly the objects that compose the user interface, and this is a completely emergent structure motivated by the use of the desktop metaphor or, in general, by the use of a graphical interface. These are objects to be manipulated by the user and the system in order to be able to communicate with each other. Historically, human beings have been using machines through interfaces (buttons, levers, gears, etc) so these objects receive such features from the space of machinery, in general and, in particular, the workplace.

7.4.3.2 Entity objects

These objects are representative of the business process or application domain. That is why they are also called *business objects*. Since they represent conceptual-things of the workplace (mainly documents that reflect *business transactions*), they are inherently persistent. Most of the concepts of the business domain are documents reflecting agreements (like an account that represents the agreement between a client and a bank where the client agrees to perform deposit/withdraw transactions while maintaining in the long term a positive balance and so on) and transactions. So there is a need to maintain all transactions recorded. This feature from the business is projected into the blend and objects belonging to this kind are intrinsically persistent.

7.4.3.3 Control objects

We have already shown that most features projected in the blend are derived mainly from stories where the object receives the action from a subject. What is projected is the action performed on the object, and it is integrated in the software object. But these are not all stories in the workplace, because subjects (people performing activities) have to coordinate actions with other co-workers and coordinate actions using many artifacts (documents, tools, etc) in order to accomplish a special transaction.

This coordination capacity is also projected onto software objects so that some special kind of task is to be performed by objects. This is a feature of workplaces, where a group of specialised persons need to be coordinated in order for them to perform new tasks. In a mature environment, where there is a continuous learning process from each other and from the changing environment, such a coordination feature would be unnecessary. Perhaps this learning capacity is to be projected –in the future- to objects.

The richness of features the new blend may have depends on the richness of the stories that occur in the workplace. The stories of the blends correspond to operations or methods (in OO terms) of the Object.

7.5 Application to the Flex Project

The ideas expressed in this Chapter have a good realisation in the scenarios developed in the Flex Project. We have been talking about *user's stories* as stories produced by users. They represent the user's concerns, the activities users are involved in, the information needs in order to efficiently perform their tasks and so on. So these user's stories apply to actual HIC situations in which real activities determine the need for new software systems.

Nevertheless, we attend to other situations where the design of new systems can not be helped by *actual* stories, simply because *there are no such new situations*. This is the case when we witness the design of a *new type of device* like the Home Information Center (HIC), as the new type of device has not yet been used in real situations and hence we do not have a user's direct experience.

In these cases we propose to call such stories, *stories about users* or stories of virtual users. These stories are near literary stories but are imagined to produce the effect of predicting new situations for users and devices interacting with each other.

7.5.1 Scenarios as stories

In Flex the scenarios are built as series of stories. The total set of stories is grouped into paragraphs so they may be better handled and analysed. It is interesting to observe how these stories are not purely descriptions of interactions with the device to be designed. They also represent other aspects of everyday life as we can observe in any paragraph:

"P1. Pia lives with her 8 year old son Patrick in a small town in rural Scotland, Kirkdean. She teaches French at the local school, and although not originally from the area has become very involved in local activities since moving there 3 years ago. Tonight after work the four other committee members of the local amateur dramatic society Pia attends are coming round for a meal to celebrate their success in securing a grant from the local

government authority to put on a summer open-air theatre festival" (Scenario S99/01. First paragraph. What shall we have for dinner)

There are some descriptions that are not directly –apparently- connected to our purposes (to imagine the future interactions with the HCI device): Pia lives with her 8 year old son Patrick, she teaches French and so on. The important fact is that these descriptions –stories- help us to recreate the general context of the activity. But, even more importantly, we don't know in advance whether a given story may or may not be related to our purposes. The whole description –as *wet* information- may be a source of new ideas in order to design the HIC device, as the following examples illustrate:

Pia lives with her 8 year old son: The son is another potential user of HIC
Pia lives...in a small town: To take into account special needs of inhabitants of small towns
Very involved in local activities: Information about local activities, resources, grants...

This is a strong reason to maintain the informal description of scenarios: we may need to return to the source information to consider new viewpoints about the software to be designed. The informal information is not redundant information that must be transformed into a formal information and –once completed- should be discarded. Scenarios are essential to the design –and redesign- phase and have to be maintained in its original form, as user's stories –real ones- or stories about users –imagined ones.

7.5.2 Scenarios transformed into Use Cases

Scenarios –even if maintained in their original form- have to be transformed into semi-formal structures: Use Cases. It is the analysis of the different paragraphs of the scenario that may provide the first drafts for Use Cases. In the Flex Project there is an elaboration of Scenario 99/01, in which there is a search for examples of spoken input. Tracing the scenario, it could be seen at the end of paragraph 3 that "while she [Pia] makes some toast she activates the HIC". As one of the reviewers points out "voice activation of the HIC will be required in hands-occupied situations like this. Need a HIC voice activation Generic Input Method scenario" (Note 3 to Scenario 99/01).

After the Scenario Elaboration, it is postulated that "in order to avoid the situations where the user has her hands occupied with other things, it should also be possible to activate the HIC using speech:

- "Turn on HIC", "Activate HIC", "Start HIC", "Open HIC"
- When the HIC is on standby it may not even be necessary to utter the word 'HIC' –i.e "Turn on", "Activate", "Start" etc. will be enough to activate the HIC.

This is an example of a Use Case, a first interaction between the user and the device in order for the latter to be turned on. This has all requirements needed for a Use Case: the user will get a useful result derived from the interaction: the device that starts functioning to operate with it.

The analysis may be applied to all the paragraphs of the scenario in order to detect all possible use cases. As it can be seen, the fourth paragraph offers us the example of another Use Case, that of selecting items or links on the screen. As "the four other committee members of the...society Pia attends are coming round for a meal", she needs to look for something for dinner and then she will select an item (Recipes) from the screen.

This selection will offer a list of new items (*Recipes by Region*, *Recipes by Food Type*, *Menus*, *Restricted Diet Recipes* and so on) which could be also selected the same way the main selection was made (Recipes). If the selection of *Restricted Diet Recipes* is made (supposing that all selections are spoken input for the device), then a list of names of recipes will appear that satisfy the restriction of being a Restricted Diet.

Once most of the Use Cases have been detected, they may be classified –categorised- by different criteria: by applications, navigations, queries, devices and so on. Some categories –like devices- may be divided into subcategories, and the first out Use Case will appear in the "general" subcategory and in the "start" new subcategory.

Finally the Use Case is specified in terms of a structure of attributes:
Name
Number
Actors
Functionality
Constraints
Comments.

So this is an adequate example of moving from informal to semi-formal specifications of requirements. Informal requirements are specified as stories about users (scenarios) that evolve into use cases, a semi-formal specifications of such requirements. Use Cases are then realised as interactions between objects (supposing an object-oriented implementation) whereby the utterance ("Turn on") is analysed, recognised and then executed as a starting command.

7.6 Pragmatic Considerations

Our aim is to maintain both informal and semi-formal (even formal) information in modelling HCI systems. Both types of information have their advantages, so it will be helpful to have both types of representations simultaneously. Some authors (Ramesh, 1998) point out the importance of pre-requirement-specification traceability, "that is, traceability that addresses the question: Where do requirements come from?" (p. 40).

Observed in detail, this is the central issue. Where do requirements come from?.

In our discussion we have postulated that the origin of requirements are user stories, so if we wish to maintain coherence with such a premise requirements must come from the user stories. This is a complex problem as observed by Goguen (1994) where there are cases where the use of specialised methods –video-based analysis or ethnomethodology- is advisable.

"Ethnomethodology (Garfinkel, 1967) can be seen as a reaction against the 'scientific' methodology of traditional sociology...Ethnomethodology tries to reconcile a radical empiricism with the situatedness of social data, by looking closely at how competent members of a group actually organise their behaviour, and in particular, at the *categories* and *methods* that they use to render their actions intelligible to one another; this contrasts with presupposing that the categories and methods of the analyst are necessarily superior to those of members...Through immersion in data from particular social group (such as stockbrokers), the particular competencies are gradually acquired that enable the analyst to be a sensitive and effective instrument in that domain. In this way, subjectivity is harnessed rather than rejected."

In some cases, where even the best intentions on the users behalf, it would be very difficult to 'elicit' or 'capture' most of the requirements. It is a known problem that, for somebody who participates in a given everyday activity, it would be quite difficult to be aware of all the operations involved while performing the activity.

In general, these are very special cases and the usual workplace context allows for user stories to *contain* most of the requirements (italics are intentionally used to point out the conduit metaphor underlying such expression). In a previous chapter we have stated that fabricating requirements is an open-ended process, so what we are proposing is the specification of requirements in a given state of evolution.

Imagine that we have a previous elaboration phase that collects all user stories (in the case of using ethnomethodology methods The description of user operations would then be considered as part of such user stories), we can then state that it is necessary to use some traceability method between user stories and use cases.

7.6.1 Traceability as mapping

We cannot consider traceability as an equivalent of mapping between mental spaces. Projecting from some spaces onto others is a complex process which intends to produce meaning. Projecting implies a network of spaces and multiple and complex projections between them. So, even if traceabillity is by no means a simulation of projecting between spaces, it reproduces –even remotely- some of these unconscious projection mechanisms.

User stories can be broken down into units, such as sentences. Sentences are units since they describe basic stories with subjects, actions and objects. Sometimes we consider a group of sentences as a unit, the way we consider a group of sentences to be a *transaction*.

As all stories have to be reflected in a document produced by a word editor and we can divide the user stories into units and include a tag to each of them.

In order for this proposal to be useful, we believe that a good solution would be to implement traceability from user stories to use cases. Several solutions could be proposed but the simplest is to create HTML documents in which to include the user stories (as can be seen in Jeffries (1999)) in JPEG or GIF format. These documents would include image maps to link them to other UML documents (glossary for abbreviations; vision for aspects of the new systems to be taken into account; parts of use cases for core user stories and so on) in order to maintain most of the semi-formal user information. This contextual information has the advantage of being a source of new design solutions as well as a reference for a contractual basis.

7.7 Conclusion

In order to develop effective human-computer systems, there is a need to understand and represent contextual information. We propose that to include user's stories in the system model instead of maintaining them in secondary (manual) files or even losing them, is one contribution to this aim. In order to achieve traceability between user's stories and use cases we have studied what could be the possible mappings between both types of representations.

The concepts of story, projection, mental (or conceptual) spaces and blends, derived from experientialism, help us to understand the structure of user's stories and where the concepts have originated. This in turn allows us to develop object definitions based on the shared objects of the different stories with operations corresponding to the actions. Actors in the user's stories become actors in use cases with the use case description capturing and structuring the partial stories inherent in the full user's stories.

Finally, we propose the usual solution of linking user stories -informal requirements- and use cases -semiformal requirements- through HTML documents in which user stories could be maintained as JPEG or GIF images and creating on them mappings zones to other UML documents. UML documents may be as varied as glossary entries, paragraphs in the Vision document, parts of -or whole- use case diagrams and descriptions.

As with many other aspects of software engineering and HCI, it is not sufficient to rely on ad hoc methods of systems design. Bringing the concepts of experientialisms to bear on why a particular design is appropriate enables a more rational, rigorous and user-centred decision to be taken.

Chapter 8

Conclusion

This thesis was an exercise of applying cognitive semantics –or Experientialism- to Human Computer Interaction methods, where we attempted to demonstrate that most of the methodological constructs may be derived from general cognitive processes like image-schemas, metaphorical projection and conceptual integration. We also attempted to show how metaphor is pervasive in all domains of computing, from basic concepts such as instruction, procedure or execution to more elaborated conceptual integrations such as the desktop user interface or layered architecture, broker and so on. Other issues we have attempted to demonstrate are the non-differentiation between literal and figurative language; how we can make a guideline –in form of a pattern- aware of the conceptual integration process –a process that mainly runs in an unconscious manner-, and how Conceptual Integration and Scripting are two essential HCI cognitive tools.

The same Kaptelinin's remark regarding Activity Theory, "Activity Theory is a general conceptual approach, rather than a highly predictive theory" (Kaptelinin et al, 1999, p. 28), may be applied to these early stages of applying Experientialism to HCI. The reason is twofold: on the one hand our unconscious mental processes –like conceptual integration or generating metaphors- are harder to grasp than those considered as conscious ones (as building a system model). Experience, in general, is a mix of conscious and unconscious behaviour as when driving a car and doing operations (unconscious performance) that become conscious (actions) when a breakdown is produced. On the other, both Activity Theory and Experientialism includes the context into their analysis, and a context –a specific situation- is just the opposite of what is usually considered as a classical scientific phenomena: a general and reproducible situation.

8.1 Modelling notation is derivable from image-schemas, scripts and blends

We have presented evidence that most of the HCI notation derives from image-schemas. This is valid for entity-relationship diagrams, data flow diagrams, state transition diagrams, and class diagrams. Object-oriented technology requires additional cognitive patterns, so we can consider that objects –even based on the OBJECTS ARE PEOPLE metaphor- are blends in which we can detect emergent structures.

As objects are more complex constructs mainly characterised by interactions, the role of stories is essential to them. Requirements may be captured through user's stories to build more abstract scripts: the notation used to build models of requirements capture is the *Use Case* construct. This abstraction is not necessarily object-oriented, although Jacobson has made a blend from the input mental space of scripting and the mental space of class diagrams, producing the concept of use case as a *stereotyped* class or specific type of class. The important fact for use cases is that even if they may be considered as object-oriented notation, they may be used in a large context, in modelling business processes and in non-OO developments.

The same idealised cognitive model (ICM) -the script- is applied to other UML constructs, such as sequence diagrams and collaboration diagrams. Both types of diagrams depict object interactions the same way as use cases do; the difference is that use cases depict the interaction

between two coarse grained objects: the user and the system (the part of the system, in fact, specialised in the specific interaction for producing a given result) while sequence and collaboration diagrams represent interactions between individual –fine grained- objects.

8.2 Metaphor is pervasive in HCI: from requirements to design

We have shown how metaphor may be used from a quite general level such as the organisation level, to more narrow domains like the workplace domain or the operational –cognitive- domain where we use HCI notation mentioned in the previous section.

At the organisational level, metaphors are embedded in a larger context consisting of discourse. The idea of discourse means an assembly of heterogenous elements; techniques, artifacts, practices, experiences, fictions and language, including metaphors. Each of these elements may reinforce each other in a mutual entrenchment. A metaphor may be placed at the origin of a given tool having produced the conceptual context in which the tool was thought and built. This is a proposed sequence: *thought* as the use of conscious methods would allow to analyse and design a computer-based sign, and then *built* –implemented- on the basis of the models produced during analysis and design. But the mere presence of the tool reinforces the metaphor that allowed its conception. At a given moment both elements are no longer differentiable, being the generator metaphor and the artifact just one thing: that is why people say the *desktop metaphor.*

As at the organisational level there are contradictory points of view –the organisational point of view and the employees point of view- we have to detect both assumptions in order to design systems that are suited for the organisation as a whole. As Suchman (1995, p. 56) points out "we need to reflect carefully on the kinds of secrecy that surround specific knowledges and experiences of working practice and the implications of making them visible". We think that these kinds of secrecy may not be other than unconscious assumptions that are sheltered behind a set of metaphors.

The organisational level is the main level we have to look out for in order to detect the requirements. Even if we perceive the workplace level as the very place to look over for requirements very frequently, what is really determining the context, the goal, or the scope of a software system is the organisational level. If we really wish to answer the question*: where are requirements coming from?* (Ramesh, 1998), it is in the organisation level that we will find the answer: the network of assumptions frequently based o a set of metaphors.

It would be advantageous, therefore, to detect the set of metaphors that are ruling the business and that are demanding a given software system as we can see in. David Wellman´s statement about work: -"how people work is one of the best kept secrets in America"- and the derived Suchman's proposal for making work visible requires an efficient method in order for it to be an achievable goal. As most of our cognitive processes and social interactions are essentially unconscious, the availability of an efficient tool to disclose the *secrets of work* is a fundamental prerequisite. Even usual ethnographical methods may not allow to understand what is going on in a given situation, as the following sentence reveal: "even the most seemingly unmediated, veridical representational forms like video recordings do not wear their meanings on their sleeves to be read definitively once and for all" (Suchman, 1995, p. 58).

One of the most significant discoveries in cognitive science is that most of our thought is unconscious, not in the Freudian sense of being repressed, but in the sense that it operates beneath the level of cognitive awareness, inaccessible to consciousness and operating too quickly to be focused on (Lakoff and Johnson, 1999). The same applies to our awareness of social interactions since we are inserted in social institutions with a set of social roles, assumptions and activities that obscure the real perception of the underlying governing rules.

The workplace level has two sides: one concerning and integrated in the organisational level. All we have expressed regarding the organisational level may be applied to the workplace level. But there is another –autonomous- side regarding the metaphors and blends we use for designing the required software systems. It is in this context where the use of metaphor provides more additional advantages than more traditional approaches.

There is another important issue being the construction of complex metaphors. Primary metaphors are like atoms that can be put together to form molecules. A great many of these complex molecular metaphors are stable, conventionalised and entrenched for long periods of time. We have seen some examples regarding the repository, which could be considered as a contradictory set of concepts put together. In fact, the repository considered as a container, a dictionary and a platform, implies the construction of such a molecular metaphor.

The previous concept of molecular metaphor is based on the fact that we frequently use multiple metaphors for a single concept. It is not a contradiction but a real need trying to use multiple metaphors in the form of a blend to conceptualise a new software system.

In relation to the operational level, we have already discussed how HCI notation derives from usual cognitive patterns like image-schemas, blends and conceptual integration.

The following table shows a possible representation for the underlying assumptions of the vision of a new system (the Trouble Ticketing System). We use the Unified Process terminology (Inception Phase, Elaboration Phase, Construction Phase, etc). (Jacobson et al., 1999)

Inception (First Phase)	**System Vision (Rationale)**	**Underlying Assumptions (Where are requirements coming from?)**
To build a new system for Trouble Ticketing	The system will replace another one, which is based on workers' conversations. In these conversations, they compared notes about what was going on at each end of the circuit of printers.	**Metaphor 1:** "Employees are children" abandoning their responsibilities and engaging others in conversation without any usefulness for the job **Metaphor 2:** "The Mind is a container", so knowledge is an autonomous internal process that do not need continuous feedback from external co-workers (conversations)

8.3 HCI language is fundamentally figurative

Cognitive semantics –or Experientialism- has provided enough evidence that much of the empirical research has led to a rejection of dichotomies that have in practice separated the study of language from the study of literature, namely, the supposed autonomy of syntax from meaning, the supposed autonomy of the language system from other cognitive systems and the supposed separation between literal language and figurative language (Turner, 1991, p. 20-21). Turner proposes to speak of concepts and their connections as having degrees of generative entrenchment, where degree of generative entrenchment is the extent to which the rest of the conceptual system depends upon or is invested in that connection.

Turner points out that –as an example of the dynamism in the intermediate range of generative entrenchment, some researchers of artificial intelligence like Allen Newell and Herbert Simon, worked so continuously under the guidance of the analogical connection between the brain and a contemporary serial digital computer and the brain, that they ultimately came to see them as in truth literally the same thing, belonging to the same category (op. cit., p. 141). The consequence is that –in their research paradigm- the analogical (figurative) connections acquired a high degree of generative entrenchment. In a similar way, Agre (1997) speaks of the tendency of cognitive science –meant here as an equivalent to cognitivism- to understand a wide variety of concepts in abstract terms. "In each case, I will argue, the result has been the steady entrenchment in computational practice of a mentalist view of both human beings and computation" (op. cit., p. 77).

What Experientialism has shown is that there is no basic cognitive difference between both types of cognitive processes. We have shown a list of common examples in HCI, from basic concepts like instruction, procedure or sentence to more elaborated ones (layered architecture, sandbox or broker). As computing is a quite abstract discipline, we can say that most –if not all- of our language is figurative. The implication of this statement is that we have not to worry about the use of metaphors but quite the opposite trying to encourage the use of this trope. It is not a question whether or not to use metaphors, the question is to analyse whether a metaphor or a group of them fit the requirements of a given software system.

A very important job in HCI is the search for new conceptual integration. We recognise that real, complex new conceptual integration appears from time to time (such as a spreadsheet or desktop interface) and that usual applications are built on already made developments of user interfaces and known systems architecture. Nevertheless, we can always use conceptual integration to create new, usable and efficient software. The frequently reaction against the use of metaphor in designing user interfaces or the statement about the dangers of anthropomorphizing the computing concepts is the effect of an objectivist philosophy: the use of a literal, formalised language allows us to treat and discuss software systems in a more suitable, precise way while the use of figurative language –metaphors- may be the source of wrong designs.

The problem with such a conception is that it does not take into account the fact that HCI is more and more a discipline of human needs about computers and computer-based tools. Therefore the technical issue of architectures, languages, communications and so on is important and usually complex, but the human related part is substantially more complex.

Under the *generative entrenchment* view of conceptual structure, what matters for a conceptual system is its fitness, its capacity to empower us to operate within forms of life (Turner, 1991).

There are a great number of examples of technically correct systems but which are not usable because of ergonomic considerations or procedural problems. The human –or organisational- aspect implies the use of some suitable guidelines for designing new software systems, and metaphor and conceptual integration offers a non-bettered approach to this aim. The reason is not that we have found a new method or approach to use, but that metaphor and conceptual integration are two ongoing processes we apply in our everyday activities.

Then, the problem is how to apply conceptual integration in an ordered way, and how to derive some interesting guidelines for the use of both metaphors and conceptual integration, something we have also proposed as a pattern to follow.

The following table shows how –in a similar way to that applied to underlying assumptions- we can represent the successive transformations between the original figurative language and some of the features conceived for the new system:

Figurative Language	Meaning	Features
The Java Sandbox	The sandbox allows children to be at play without risk for the neighbour	**Feature 1:** "The Java system is a container" for programs to be executed in such environement **Feature 2:** "The Java system is safe", so there are no risks for the external programs and resources
Layered Architecture	Different systems are different layers in an structure	**Feature 1:** The relationship between two adjacent layers is a dependency one **Feature 2:** The relationship between two adjacent layers means that elements of the API are to be executed through the Java Virtual Machine
Agents are insects	Agents may add some pheronome to resources (documents)	**Feature 1:** The agent add information in order for other agents to detect it. **Feature 2:** This information attracts other users

8.4 Conceptual Integration may be framed as a pattern

The goal was to treat a cognitive process we deal with unconsciously in everyday activities in a way that could be useful for design purposes. The discovery by Christopher Alexander about the creation of a pattern language that approximately reproduces image-schemas when applied to specific cultural practices like architecture, gives us the guidelines for frame conceptual integration in the form of a pattern.

The underlying intention is to define a process as a pattern in order for us to be able to use conceptual integration in an ordered way. The usual idea is that there is a metaphor and a design problem to determine all specific characteristics of individual elements of the new domain, in this case the desktop graphical user interface. The pattern aims at giving a detailed definition of each of the elements of the conceptual integration with an indication of the emergent structure and analysis of each of the input mental spaces involved in the integration.

The pattern tries to offer a guideline for defining computer-based signs, a complex blend between all the input mental spaces. Another important aspect of using a pattern is to assure that all interesting features from input mental spaces are integrated in the conceptual integration, and to rationalise why others are given up and help to maintain the optimality principle.

The conceptual integration pattern provides part of the whole model that some authors call the Conceptual Model (Winograd, 1996). Sometimes, this model is called the user's conceptual model, or based on the user's mental model. It is evident that a completely new conceptual integration might not be based on such user's mental model: instead, our experientialist design approach is based on the concepts of image-schema and metaphor.

Conceptual integration is the raw material for new ideas It is the generative mould for the construction of computer-based signs, but this implies an iterative process. This method assures the usability of computer-based signs and the integration of such signs in higher-level systems. Conceptual integration on its own does not guarantee either the efficiency or the suitability of computer-based signs, but it is a rich source of inspiration. The designer's job is, when seen as a combination of conceptual integration and usability testing, a very heavy workload.

8.5 Scripting and Conceptual Integration are two essential cognitive tools

Our criticism of the top-down approach led us to a new way of thinking about software requirements and design. The top-down approach –as recursive applications of bubbles-containers- is not a useful method for deriving the model for a new system. The problem is that usually there is nothing in each of the containers we continue to open. The container image-schema is useful for representing some elements in a modelling notation since we are focusing on entities or classes.

The current use of Use Cases and its successful application in many different domains is in line with our statement that scripting is cognitively representable, as it is one of the idealised cognitive models we employ in everyday activities. Scripts are abstract stories that model our interaction in some specific situations like going to the restaurant, going shopping,

withdrawing money from a automatic teller machine and so on. This way we can use Use Case notation more efficiently than DFDs because there is a cognitive model that is used to describe a specific situation, and in the case of requirements capture, a specific interaction between a user and the system.

This is why we can say that Use Cases are cognitively correct (Imaz, 1999a). There are other proposals for using concrete user's stories in place of the abstract stories of Use Cases. We argue that both approaches are not incompatible, that user's stories may be the raw material we use (including the same characteristics of traceability we proposed for conceptual integration) for creating Use Cases. Use cases are more adequate for further realising –refining- them in terms of collaborating objects. The granularity of Use Cases better fits the need for specifying a system than user's stories would. We have already discussed (Imaz and Benyon, 1999) how a user story may be a partial description of a Use Case, a request to modify an already existing Use Case as in the example of a Payroll System.

A user story may offer additional information for our developer purposes. But a user story is usually composed of many different requests, points of view and intentions that we have to sort out in order for it to be useful.

Finally, we can add that –as already stated for conceptual integration- the second cognitive tool to guide our design process is a blend using multiple metaphors. The use of multiple metaphors for defining what a given system will do is an excellent guideline. As our objectivist (de)formation led us to consider the multiplicity of metaphor as something wrong, an irregular situation where we would have to decide which of them fits better the design problem, we usually think of one of them as the correct –or better- choice.

The pressure to select only one metaphor is also determined by the referential –literalness- conception of language. As language has to refer to something in the real world, it would not be possible for multiple metaphors to be applied correctly to the same subject. But the example of requirements and the various simultaneous points of view derived from them –as well as other examples- convinced us that this is more the case than the exception. We can perfectly apply multiple metaphors to our designs, which would imply a richness of features encapsulated in conceptual integration and that would be ready for use when needed. What we would have done is to open the informal specification of systems so as to convert them into large sources of ideas that would be provided as new features. This was Goguen's aim of maintaining both types of information, informal and formal, where informal –or wet- information could be useful in an immediate way for transforming it into a formal –dry- one.

8.6 Further work

This thesis is a small step along a long path that attempts to construct theories of cognitive foundations of HCI methods in particular, and of scientific knowledge in general. We have already explained why we have chosen Experientialism (as a criticism of Western rationalistic knowledge whose aim is to attain a higher degree of rationality, the idea of knowledge as processes that ground on bodily activity and so on). It was not our goal to try to show a broad range of approaches that might have been used with a similar purpose nor to compare them with Experientialism, as this task would have implied a thesis on its own.

The same way as we use axioms in mathematics whose usefulness may be derived from the consequences deduced from using them, we may evaluate the usefulness of Experientialism

from the consequences of employing its concepts. As Heisenberg points out, it is the fertility of the concepts –and not their theoretical *precision*- that have to be evaluated.

The fertility of a given approach is not only measured by the direct results but also by the whole questions and new problems it raises. Among all the possible open-ended questions Experientialism raises, we may point out the following:

8.6.1 Computer-based signs (blends)

We have shown how some computer-based signs are derived as blends from a useful metaphor, which has to be further refined. In particular, most of the computer-based signs derived from such metaphors –and particularly the desktop metaphor- may be viewed as *boundary objects*, in terms of the UML notation. The concept of computer-based sign suggests a broader view of a blend, a view that may have a set of representations: as a boundary object, a control object and an entity object. The concept of computer-based sign gives a unity to a set of –apparently- different objects. We have only one sign –the *folder*, for example- with various representations (as a *boundary object* when visible on the screen and as an *entity object* when stored as a file or table in the file system).

The concept of blend –computer-based sign- allows us to give unity to a set of objects. It would be interesting to study the relationships between signs and objects and to see whether the sign offers a higher level of abstraction to the concept of object. At the same time, as signs may be considered as coarse grained objects, it would be useful to ask about the relationships between signs and Use Cases. In such cases, signs and Use Cases might be considered as equivalents, in which case Use Cases are built as a group of –intermediate level- signs.

8.6.2 Conceptualising by means of figurative language

After having shown that figurative language is a powerful tool for conceptualising new software systems, there is a broad range of questions that could be stated. It would be particularly useful to see whether there are some type of figurative language –or special uses of figurative language- more suitable for a given purpose.

Must there be some special relationship between both domains –the conceptual integration and the source domain- when conceptualising one of them in terms of the other? When using a network of domains to conceptualise another domain, is there some type of relationship between all of them? Or does the network of domains only facilitate a further and more precise categorisation?

8.6.3 Stories to describe requirements

We have seen that stories are the raw material, the essence, of scenarios. Scenarios are the source of inspiration for deducing the needed user cases in order to design the new software system. But when describing scenarios, there are some stories apparently not related to the main flow of events that determine the use cases needed for the design, These stories are intermixed with others that could be considered as *essential* to the design.

The question might be how to merge *essential* and *unessential* stories to build scenarios. The description of scenarios have to recreate –in a high degree- the context of the use of artifacts, but how do we evaluate the precision, the degree of suitability of scenarios in

function of stories used to describe them? Maybe the answer is that there is no way of evaluating in advance the degree of suitability of scenarios, that such suitability reveals during an iterative process with higher degrees of precision. The conclusion will mean giving up such an attempt and will represent a progress in the field of HCI.

We have pointed out that it is not easy to foresee whether some of the unessential stories will not be used later for some issues of the design. Moreover, is it possible to differentiate between *essential* and *unessential* stories at an early stage as scenarios?

Is it a good criterion to include a lot of stories in the scenarios in order to have a much richer source of inspiration? Or is it better to limit the amount of stories in function of their suitability for the use cases we are looking for?

Appendix A

THE FLEX PROJECT

(Source: http://www.dcs.napier.ac.uk/~dbenyon/Flexoverview.html)

Flexible knowledge-based information access and navigation using multimodal input/output

Project Objectives

The project will provide the technology for bringing the information age into the home for everybody to use. The target product is a new concept for a device called the Home Information Center (HIC) which describes a device for multimodal access to information and services relevant for the everyday life in a family. The FLEX project will provide the FLEX Technology software for supporting intuitive navigation in and visualisation of information and flexible queries for information, as needed for the realisation of a HIC.

The markets for information access to home shopping, home banking, video-on-demand, web-access, traffic information, etc. are expected to explode within the next two years. The HIC addresses the niche of trend setting families who are open to changes and who already have PCs and audio-visual equipment. As the FLEX Technology of intuitive navigation and flexible search meets the users' strong demand for enhancements of information access systems, the business prospects also include the integration of the FLEX Technology both into the partners' own products as well as OEM products.

The Home Information Center

Today we see two user situations in the home: The TV in the livingroom as a lean-back situation, and the PC in the office as a lean-forward situation. In the lean back situation the user is being entertained and relaxes, and in the lean-forward situation the user is actively and focused producing. Already a variety of devices brings internet with thousands of programs to the living room through set top boxes, and also services such as home shopping, home banking, video on demand, health care, energy management, traffic information etc. are provided to the home. It is not sure that the lean-forward situation of a PC or the lean-back situation of a TV in a living room will be the right contexts or the right devices for the new services. Instead we try to define a new device HIC which is neither a lean back device as the traditional TV nor a lean forward product like a PC, but a kind of moving around device where users can get and provide information while they are occupied with other house doings. We call this a move-around situation for infotainment. See the illustration in Figure 1.1.

Generically, the Home Information Center is a multimodal input/output device that can be placed in any fixed position in the home and that supports flexible interaction and an intelligent underlying information structure. A vast set of services is relevant, and the design should allow for extensions with no need to redesign the system. The project will focus on services supporting information and messages in the daily life of a family. Examples of such services are:

Memorisation of handwriting and of written text, for notes and messages.
A calendar.
A cultural information base for planning family entertainment activities.

TV information.
Traffic information.
Internet access.

The FLEX Technology

The utilisation of the phenomenally growing quantity of information and services offered online is hindered by relatively primitive access mechanisms, requiring users to point and click their way through myriads of hyper-links and web sites [Lau et al. 1997]. The HIC requires an improvement to the current praxis. FLEX addresses this problem by producing and integrating two key technologies that together will make up the FLEX Technology, cf. Figure 1.2:

- An intuitive navigation support system. The user must be provided a comprehensible mental model of the current information space and the system must be able to interpret concrete references to the system output [Benyon and Höök 1997]. The user understanding is supported by intuitive representations which may be '4D' such as signifying informational distance by blurring and marking referents with different colours.

- A flexible query system. FLEX flexible, powerful query mechanisms [Christiansen et al. 1997] supported by an abstract information space adaptable to the user profile and current dialogue.

In addition, a number of supplementary technologies will be adapted and integrated:

1) An *information space*. This is an abstraction of the contents of the information bases underlying the tasks in the Home Information Center. The abstraction is extracted and maintained semi-automatically via datamining.

23) A *dialogue interaction*. The FLEX Technology will support multi-turn interaction as well as the use of context from the current and from earlier interactions, both individually and across family members. For context across different user interactions this is associated with the concept of awareness.

5) A *multimodal interaction*. The FLEX Technology will support speech, pen, touch, and keyboard as input, and sound, images, text, and animation as output.

2) A *speech interaction*. For the hands-free situations FLEX will provide support for speech input at the level of speaker independence, up to 1000 words in global vocabulary, 50 words perplexity, and commands.

Objectives

With reference to the objectives of the Esprit Theme: Information Access and Interfaces, and to the above descriptions of the Home Information Center and FLEX Technology, it can be stated that:

> **The main objective of the FLEX project is to establish the technological platform for, and determine the value of, the synergy of intuitive interfaces, intelligent**

navigation and flexible search in the multimodal environment of a Home Information Center.

FLEX addresses the need for providing easy access to information for ordinary people in their homes. The overall product addresses the modern family's need for a higher and more coordinated information level. The intuitive interfaces, the intelligent navigation, and the flexible search are each exploitable separately, while they also support and complement each other nicely.

In relation to the approach of this project the objectives may be detailed according to users, technology, and business. The user driven objectives for this project are:

- **To conduct laboratory and field studies of the HIC.**
  The studies will provide measures including both quantitative numbers such as speed and transaction success rate, and qualitative evaluations such as user satisfaction. Laboratory studies will be madefor each of the two first prototypes, field studies for the third prototype. See also the other user and technology objectives.

- **To find out where in the home a *HIC should be placed.***
  Likely places are the entry, the kitchen, and the living room. The place should enable casual use while people are moving around performing other home tasks.

- **To provide an appropriate minimal set of HIC applications.**
  The applications or services provided with HIC should fit with and support a modern family's at times fragmented everyday life. The applications included should be naturally used during user tests, and users should express satisfaction with the given set. The application set must include memorisation and calendar, and should include three other services, currently information about cultural events and places, TV programmes, and traffic information are proposed, but this will be decided on within the first 4 months of the project. During the later experiments users may express a wish for moreapplications to be included which is ok as long as the application types can be pointed out to be of a type catered for in the FLEX/HIC architecture and not increasing the cost of the product itself. The cost for extra services should be paid by the content provider who will have the option to provide the service for free or to require a subscription or per use rate.

- **To produce an intuitive interface with easy access to applications.**
  An important aspect of a HIC is that ordinary family members can and will use its applications. User tests should show that each of the example applications has a transaction success rate above 90%, and for selected homes the people should use the HIC and its applications daily.

The *technology driven objectives* concern the software necessary for the Home Assistant, including:

- **To establish an open, modular architecture for search and navigation via intuitive multimodal interfaces.**
  The architecture must enable content providers to join in with a low starting overhead.

- **To design, develop, and validate the components for flexible search and intuitive navigation in semi-automatically structured information spaces.**
  The two components should meet standard production level software standards and have APIs that make them reusable in other contexts as is.

- **To construct a fuzzy information space for representing information from a diversity of sources.**
  For information from free text fields in databases and from hypertext pages the information space should be constructed automatically. For a stochastic selection of terms the constructed associations should correspond 60% with a human test panel, cf. e.g. work by Philip Resnik.

- **To construct a speaker independent recognition with 1000 words suitable for navigation and daily message activities, robust in the noisy home environment, 50 words "perplexity"/"local vocabulary".**
  The word recognition rate should be at least 95% for a person in front of the HIC in a normally quiet room in the home, at least 80% if other people are speaking or there is music (at normal speech level) in the other end of the room. [Percentages to be confirmed].

- **To construct an interaction control which supports use of mixed-modality input, multiple turns and inter-person awareness.**
  This means 1) that users should be able to navigate and provide information and queries incrementally using the available modalities interchangeably at their own choice, and 2) that the actions of earlier users should be reflected in the interface when appropriate.

- **To construct an intuitive metaphor or visualisation for navigation in information.**
  The visualisation should support different modalities. In addition to the direct visualisation the underlying FLEX Technology should include a model of the output enabling the user to refer to the active fields and buttons in the HIC output.

The *business objectives* concern the cost and marketability of the HIC and the FLEX Technology, and include:

- **To demonstrate that there is a market for HIC.**
  The consortium is confident that there is a need, but the HIC is a new concept and a number of questions need to be validated, cf. the other objectives.

- **To associate committed content providers with the project.**
  In addition to the price of the physical product and the software technology an important point is if content providers will use HIC for new or value-added products.
  Content providers associated with the FLEX project will supply the project with current information for the HIC and with feedback on the business opportunities and potential success of the HIC and the FLEX Technology. A project success indicator is the ability of the project to involve at least one committed content provider after year one.

- **To produce the FLEX Technology (navigation and search) for other markets.**
  The components for navigation and search are expected to be exploitable separately on an OEM basis. As a minimum, each of them should be exploited in other, commercial systems of one of the project partners.

The FLEX Technology and the HIC

The overall area of FLEX is Navigation and search in information spaces for non-IT specialists, materialised in the concrete shape of the HIC. The main product of FLEX are software technologies materialised in a HIC demonstrator to validate the navigation and search paradigms. The key software technologies for this to succeed that are produced within the FLEX project are (cf. also Section 1.2):

- *Intelligent knowledge systems*: Inference engines that on the basis of the interaction history can: a) Interpret the current user input as a query, and perform that query to the information space. b) Make thematically relevant extracts of web pages and database records in the information bases. c) Create, structure, and continuously evolve the information space from the underlying information bases using techniques such as datamining, search engines, and web agents.

- *Interaction control*: The tracking and control of the evolving user-system interaction. This includes the maintenance of the interaction history and the top level control of the acts of the user and the system respectively, e.g. deciding what domain and meta-information to present to the user, and keeping track of whether the user refers to the information space or to the navigation.

- *Navigation support*: The user should be guided through the information space and the interaction via suitable navigational clues and content representations. Moreover, users are likely to refer to spatial and logical elements of the HIC output and the interaction history. The navigation module, making use of an abstract model of the screen output, the user model, and the current interaction history supports both of these.

- *Linguistic analysis*: The inclusion of spoken commands and written natural language input requires semantic parsing, using tailored sub-language grammars, interaction history, and current domain task focus.

In addition, the construction of the HIC requires the inclusion of off-the-shelf speech recognition and html rendering modules, and the assembly of the necessary hardware setup.

Usage and scenarios

The scenarios provided in this section serve the purpose to concretise what the system may be expected to do. In the beginning of the project it will be analysed which kinds of applications and combination thereof are most likely to boost user acceptance and commercial success. Together with the architectural description in Sections 2.1 and 2.2 this forms the preliminary specification of the system to be built. For the prototype and validation purposes a smaller selection will be made.

The typical users could be a family with half-grown children, each family member having his/her busy and not too coordinated activities. A larger scenario is provided at the end of this section. Other possible scenarios include:

- **Navigation:** A single person is using the system to find the best route to his new buddy.
  Displays: A map and choice lists of names. The system may be operated via touch or via speech. The system is different from an ordinary web in the following respects: Navigation clues are emphasised, search requests may be built up during a multiturn interactive dialogue, and the navigation is based on a fuzzy information space.

- **Task1:** The man wants to produce a list of what to buy for dinner. He first locates appropriate recipes in the cooking database, and then invokes the list assistant.
  Displays: A purchase list. Feedback incrementally shown on the screen as the user touches ingredients for the dinner. Vocabulary troubles, perhaps solved via written input.

- **Task2:** The teenager is about to make dinner. She first locates appropriate recipes in the cooking database, and then invokes the task assistant.
  Displays: An ingredient list. A step-by-step guide including appropriate photos or animations of critical steps. Feedback incrementally shown on the screen as the user steps through the recipe. Tasks such as cooking takes a longer period during which they may be suspended by other tasks (and other persons). Kitchen machine noise.

- **Beginner:** The untrained user on her own using the system for the first time.
  Displays: The intuitive interface will enable users to learn the system by using it. Instructions on the HIC Monitor.

- **Family:** A family of 3 is looking at football (soccer) in the TV Saturday afternoon, and now in the pause they want to find some information on other, related sports results.
  Displays: Group of persons. Who-is-in-charge of the interaction. Noise problems. Relation between what is on TV and the HIC.

By the end of the project the contents information will be represented by at least four different contents packages. The packages will represent different aspects of expected HIC uses:

- Practical everyday support such as traffic information or cooking recipes.
- Small factual information such as weather forecasts.
- Larger, descriptive information such as cultural life and cultural possibilities.
- Business information such as stock exchange, banking, or goods catalogues.
- Directly TV related such as film guides for choosing what to see. TV related information which supplements what is on TV. An example is sports results which may supplement sports broadcasts.
- Instructions such as how to operate TV equipment (including HIC itself.)

The project will concentrate on the information aspect, but of course commercial trade is possible and expected for several of the above issues, e.g. film guides coupled with video-on-demand or business information coupled with ordering. However, these and other interactive uses such as games are not considered in the project.

The project currently has formal agreement with a cultural provider (CCIS, Edinburgh) and a TV and news provider (Danmarks Radio, Denmark), and informally with a sports results provider (Denmark).

A scenario for a couple in the morning:

Arnold is busy in the kitchen, washing up the dishes from the party he gave yesterday evening. He listens to the morning news in his loudspeakers, connected to his HIC in the *kitchen. He is going to be at work within an hour, and the speaker was saying something about a roadblock on the road between his home and his work. He turns his attention to the HIC saying, "Turn on". He could have used touch or the keyboard as well, but his is occupied with his hands right now.*

The screen now gives Arnold some possibilities. Arnold has earlier on defined his personal profile in the HIC, very dependent on the traffic situation between his home and his *work, so one of the options given is service number 5 "traffic". He could say 5, but he chooses to say "traffic situation". He is now given some* more possibilities, and says "*local" number 2. The screen now shows the block on the main road, and he realises that he has to leave 10 minutes earlier, because he has to* take another road. He hurries *up, just leaving anything in the kitchen, taking the pen from the pen holder attached to the HIC, and quickly writes " Good morning dear. Have* to leave early, will you finish *cleaning up, and walking the dog. Love Arnold". This text is memorised in the HIC's message heap.*

He turns the HIC off (really switching standby mode, ready for new interaction) and leaves.

Jane is woken by the sound of his car, and when she comes down to the kitchen, she sees a flashing note indicator on the HIC. She says, "turn on" and she sees the message *from Arnold attached with the local traffic map, and understands why the kitchen is in such* a *mess. She presses the CD button on the remote* and begins cleaning up while she *listens to the new record with Placido Domingo.*

Finished cleaning up and walking the dog, she goes to the HIC, rolls out the keyboard and writes: "When will the road be cleared for traffic?" The HIC, intelligent and flexible as *it is, automatically will know that she is referring to the roadblock on the local road, due to Arnold's request early on in the morning, displays and* says "13:30". Jane then writes "*mail Arnold", and the HIC then automatically turns on the Internet connection, turns on the mail application, and has filled out the "to:" address* ready for keyboard input. Jane *now writes "road is clear, take the easy way home? Love Jane" She says "send" and "Turn off", and is already on her way to her job.*

- *Displays: The HIC has a multimodal interface letting the user choose according to his/her immediate situation and purpose, and the HIC both can be used for the external information and the more family internal doings.*

Appendix A

Bibliography

Agha, Gul. (1990). Actors. The MIT Press.

Agre, P. (1997). Computation and Human Experience. Cambridge University Press.

Andersen, P. B. (1997). A Theory of computer Semiotics . Cambridge University Press.

Alexander, Christopher et al. (1977). A Pattern Language. Oxford University Press.

Beck, K (1999). See Web site: <http://c2.com/ cgi/wiki?UserStory

Bellamy, R.K.E. (1996). Designing Educational Technology. In Nardi (Ed) *Context and Consciousness. Activity Theory and Human-Computer Interaction.* The MIT Press

Benjamins, V. R., Fensel, D., and Gómez Pérez, A. (1998). Knowledge Management through Ontologies. In Reimer, U. (Ed.) *Proc. of the 2nd Int. Conf. on Practical Aspects of Knowledge Management (PAKM98)* Basel, Switzerland, 29-30 Oct. 1998.

Benyon, D. R. and Imaz, M (1999). Metaphors and Models: Conceptual Foundations of Representations in Interactive Systems Development. In *HCI. Special Issue about Representations in Interactive Systems Development.* Volume 14, Number 1 & 2. Lawrence Erlbaum Associates, Publishers.

Bødker, S. (1991). Through the Interface: A Human Activity Approach to User Interface Design. Lawrence Erlbaum Publishers.

Bødker, S. (1996). Applying Activity Theory to Video Analysis: How to Make Sense of Video Data in Human-Computer Interaction. In Nardi (Ed) *Context and Consciousness. Activity Theory and Human-Computer Interaction.* The MIT Press.

Booch, Grady. (1991). Object-Oriented Design with applications. Benjamin/ Cummings.

Boyd, Richard. (1993). Metaphor and theory change, in Ortony, A. (ed). *Metaphor and thought* . Cambridge University Press.

Buschmann, F et al. (1996). A System of Patterns: Pattern-Oriented Software Architecture. John Wiley & Sons.

Caputo, J. (1987). Radical Hermeneutics: Repetition, Deconstruction, and the Hermeneutical Project. Bloomington: Indiana University Press.

Carroll, J. M. (Ed.) (1987). Interfacing Thought: Cognitive Aspects of Human-Computer Interaction. Cambridge, MA: MIT Press.

Carroll, J. M. (Ed.) (1991). Designing Interaction: Psychology at the Human-Computer Interface. Cambridge, Cambridge University Press.

Carroll, J. M. (1994). Making Use: A Design Representation. In *Communications of the ACM.* Vol. 37. No. 12. December 1994.

Carroll, J. M. (1995) Scenario-based Design. John Wiley

Carroll, J. M. and Rosson, M. B. (1992). Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario. In *ACM Transactions on Information Systems*. Vol. 10. No. 2. April 1992.

Carroll, J and others. (1991). The Task-Artifact Cycle. In Carroll, J (Ed.) *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge University Press.

Chorafas, Dimitris and Steinmann, Heinrich. (1993). Object-Oriented Databases. Prentice Hall.

Clancey, W. (1989). The Knowledge Level Reinterpreted: Modeling How Systems Interact. In *Machine Learning* , 4, 285-291. Kluger Academic Publishers.

Clancey, W. (1994). The Conceptual Nature of Knowledge, Situations, and Activity. Institute for Research on Learning. 2550 Hanover Street. Palo Alto, CA 94304

Coad, Peter and Yourdon, Edward. (1990). Object-Oriented Analysis. Prentice Hall.

Coad, Peter et al. (1995). Object Models. Strategies, Patterns, & Applications. Yourdon Press.

Cockburn, A (1997) Structuring Use Cases with Goals. Journal of Object-Oriented Programming. Sep-Oct and Nov-Dec issues. See also Web site: <http://members.aol.com/acockburn>.

Cockton, G., Clarke, S., Gray, P. and Johnson, C. (1996) Literate Development: Weaving Human Context into Design Specifications. In Benyon, D. R. and Palanque, P. *Critical Issues in User Interface Systems Engineering*. Springer-Verlag

Coleman, Derek *et al.* (1994). Object-Oriented Development. The Fusion Method. Prentice Hall.

Collins, Dave. (1995). Designing Object-Oriented User Interfaces. Benjamin/Cummings.

Davis, A. M. (1993) Software Requirements; Objects, Functions and States . Prentice Hall.

Dijkstra, E. (1972). "Notes on Structured Programming", in *Structured Programming*. NY: Academis Press.

Dijkstra, Edsger. (1989). On the cruelty of Really Teaching Computing Science. In *Communications of the ACM*. Volume 32, No. 12.

Dreyfus, H. (1972, 1979, 1992) What Computers Still Can't Do . The MIT Press. Fifth printing 1997. The title of the first edition was "What Computers Can't Do".

Dreyfus, H. and Dreyfus, S. (1988) Mind over Machine. The Power of Human Intuition and Expertice in the Era of the Computer . The Free Press. A Division of Macmillan, Inc. NY.

Eco, U., Santambrogio, M. & Violi, P (Eds.). (1988). Meaning and Mental Representations. Indiana University Press.

Edwards, P. (1996). The Closed World: Computers and the Politics of Discourse in Cold War America , Cambridge, MA: MIT Press.

Engeström, Y. (1999). Activity theory and individual and social transformation. In Engeström et al (Eds.) *Perspectives on Activity Theory*. Cambridge University Press.

Engeström, Y. and Middleton, D. (1996). Introduction: Studying work as mindful practice. In Engeström, Y. and Middleton, D. (Eds.) *Cognition and Communication at Work*. Cambridge University Press.

Erickson, T. (1995). Notes on Design Practices: Stories and Prototypes as Catalysts for Communication. In Carroll, J (Ed.). *Scenario Based Design* . John Wiley & Sons, Inc.

Ericksson, Y. (1990). Working with Interface Metaphors. In Laurel, B (Ed.). *The Art of Human-Computer Interface Design* . S. Joy Mountford. Addison-Wesley

Fauconnier, G. (1988). Quantification, Roles, and Domains. In Eco, U., Santambrogio, M. & Violi, P. (eds.). *Meaning and Mental Representations*. Indiana University Press.

Fauconnier, G (1994). Mental Spaces. Aspects of Meaning Construction in Natural Language. Cambridge University Press.

Fauconnier, G. (1997). Mappings in thought and language. Cambridge: Cambridge University Press.

Fauconnier, G. and Sweetser, E. (Eds) (1996). Spaces, Worlds and Grammar. The University of Chicago Press.

Fauconnier, G. and Turner, M. (1994). Conceptual Projection and Middle Spaces. UCSD. Cognitive Sciences Technical Report.

Fauconnier, G and Turner, M (1998a). Principles of Conceptual Integration. In Koenig, J-P. (Ed.): *Discourse and Cognition: Bridging the Gap*, pp. 269-283. CSLI Publications. Standford. California.

Fauconnier, G and Turner, M (1998b). Conceptual Integration Networks. In *Cognitive Science*, 22(2) pp. 133-187.

Feigenbaum, E. and McCorduck, P. (1983) The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World . Addison-Wesley. Quoted by (Dreyfus and Dreyfus, 1988)

Fowler, M. (1997). Analysis Patterns: Reusable Object Models. Addison-Wesley.

Fowler, M and Scott, K. (1999). UML Distilled, Second Edition: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Object Technology Series.

Franklin, S. (1995). Artificial Minds . The MIT Press. A Bradford Book.

Fritzinger, J.S and Mueller, M. (1996). Java Security. In *White Papers*, Sun Microsystems, Inc.

Gamma, E. et al. (1995). Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley.

Gärdenfors, P. (2000). Conceptual Spaces: The Geometry of Thought. A Bradfrod Book. The MIT Press.

Gartner, K. (1998). Cognitive Patterns. Problem-Solving Frameworks for Object Technology. Cambridge University Press. SIGS Books.

Gentner, D and Gentner, D. (1983). Flowing waters or teeming crowds: mental models of electricity. In Gentner, D and Stevens, A. (Eds.) *Mental Models* . Hillsdale, NJ. Lawrence Erlbaum, Associates.

Gentner, D. and Nielsen, J. (1996). The Anti-Mac Interface . In *Communications of the ACM.* 39, 8. pp. 70-82.

Gibbs, R. Jr. (1994). The Poetics of Mind: Figurative thought, language, and understanding. Cambridge University Press.

Gibbs, R. Jr. (1998). The Fight Over Metaphor in Thought and Language. In Katz, A., Cacciari, C., Gibbs, R. Jr. and Turner, M. *Figurative Language and Thought*. Oxford University Press.

Gibson, J. (1986). The Ecological Approach to Visual Perception. Lawrence Erlbaum Associates, Publishers. London.

Goguen, Joseph. (1992). The Dry and the Wet. Programming Research Group. Technical Monograph PRG-100. Oxford

Goguen, J. (1993a). Towards a Social Theory of Information. Programming Research Group. Technical Monograph. Oxford

Goguen, J. (1993b). On Notation, In Magnusson, B., Meyer, B., and Perrot, J-F. (Eds) *TOOLS 10: Technology of Object-Oriented Languages and Systems*. Prentice-Hall, pp. 5-10.

Goguen, J. (1994). Requirements engineering as the reconciliation of social and technical issues. In Jirotka, K and Goguen, J. (Eds.) *Requirements Engineering. Social and Technical Issues* . Academic Press.

Goguen, J. (1996) Formality and Informality in Requirements Engineering. *Proceedings of the Fourth International Conference on Requirements Engineering*, IEEE Computer Society, April 1996, pages 102-108.

Goguen, J. (1999). An Introduction to Algebraic Semiotics, with Applications to User Interface Design. In *Computation for Metaphor, Analogy and Agents*. Nehaniv, C. (Ed.), Lecture Notes in Artificial Intelligence, Springer.

Goldberg, Adele and Rubin, Kenneth. (1995). Succeeding with Objects. Addison-Wesley

Goldstein, N. and Alger, J. (1992). Developing Object-Oriented Software for the Macintosh. Analysis, Design and Programming. Addison-Wesley.

Graham, I. (1995). Migrating to Object Technology. Addison-Wesley.

Greenbaum, J. and Kyng, M. (eds.) (1991) Design at Work: Cooperative Design of Computer Systems. Lawrence Erlbaum Associates: Hillsdale, NJ.

Gupta, R. and Horowitz, E. (Eds). (1991). Object-Oriented Databases with applications to Case, Networks and VLSI CAD. Prentice Hall.

Halasz, F., and Moran, T. P. (1982). "Analogy considered harmful". In *Proceedings of the ACM Conference on Human Factors in Computer Systems* (Gaithersburg, Md., March 15-17) pp. 383-386. Reference in (Gentner and Nielsen, 1996).

Haiduk, P. (1989). Object-Oriented Turbo Pascal. McGraw-Hill.

Humphrey, W. (1995). A Discipline for Software Engineering. Addison Wesley.

Hutchins, E. (1999). Cognitive Artifacts. In *The MIT Encyclopedia of the Cognitive Sciences*. Wilson, R and Keil, F. (Eds). A Bradford Book.The MIT Press.

Imaz, M. (1995). Object-Oriented Methods: An Epistemological Approach. MSc dissertation. Open University (unpublished).

Imaz, M. (1999a) Why are Use Cases Cognitively Correct. In *Proceedings of the Fourth CAiSE/ IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design* (EMMSAD'99). Heidelberg, Germany. June 14-15.

Imaz, M. (1999b) Designing Interactions through Meaning. In *Proceedings of the 8th International Conference on Human-Computer Interaction*. August 22-27, Munich, Germany.

Imaz, M . and Benyon, D. (1996) Cognition in the Workplace: Integrating Experientalism into Activity Theory. *In Proceedings of the Eighth European Conference on Cognitive Ergonomic:*. Green, T., Cañas, J. and Warren., C. Eds. Granada. Spain. 10-13 September 1996.

Imaz, M., and Benyon, D. (1999). How Stories Capture Interactions. In *Proceedings of the Seventh IFIP Conference on Human-Computer Interaction*, INTERACT'99. Edinburgh, Scotland. August 30th – September 2nd .

Ince, Darrel. (1991). Object-Oriented Software Engineering with C++. McGraw Hill.

Jacobson, Ivar et al. (1992). Object-Oriented Software Engineering. A Use Case Driven Approach. Addison-Wesley.

Jacobson, I. (1995) The Use Case construct in Object-Oriented Software Engineering. In Carroll, J. M. (1995) Scenario-based Design John Wiley

Jacobson, I, et al. (1995) The Object Advantage: Business Process Reengineering With Object Technology, Addison-Wesley, Reading, Massachusetts, 1995.

Jacobson, I, et al. (1999) The Unified Software Development Process. Addison-Wesley.

Jeffries, R (1999) See Web site: < http://www. armaties.com/StoryCards.htm>

Jirotka, M. and Goguen, J. (1994). Requirements Engineering. Social and Technical Issues. Academic Press.

Johnson, G. (1994). Of Metaphor and Difficulty of Computer Discourse. In *Communications of the ACM.* Vol. 37, No. 12.

Johnson. Mark. (1987). The Body in The Mind. The Bodily Basis of Reason and Imagination. University of Chicago Press.

Kaptelinin, V. (1996a). Computer-Mediated Activity: Functional Organs in Social and Developmental Contexts. In Nardi (Ed) *Context and Consciousness. Activity Theory and Human-Computer Interaction.* The MIT Press.

Kaptelinin, V. (1996b) Activity Theory: Implications for Human-Computer Interaction. In Nardi, B. (Ed) *Context and Consciousness. Activity Theory and Human-Computer Interaction.* The MIT Press.

Kaptelinin, V. et al (1999). The Activity Checklist: A Tool for Representing the 'Space' of Context. In *Interactions.* Vol. 6, No. 4, July 1999. ACM Digital Library.

Keepence, B. and Mannion, M. (1999). Using Patterns to Model Variability in Product Families. In IEEE Software. July/August 1999, pp. 102-108.

Kenny, A. (Ed). (1994). The Wittgenstein Reader. Blackwell

Knudsen, Lindskov et al. (1994). Object-Oriented Environments. The Mjølner Approach. Prentice Hall.

Kristen, G. (1995). Object-Oriented Fairy Tales, in *Report on Object Analysis and Design* . Vol. 1. No. 6. March-April 1995. SIGS Publications

Kupfer, A. (1994). Software 'Agents' will make life easy. Fortune. Vol.129. No. 2. January 24.

Kuutti, K. (1996). A Framework for HCI Research. In Nardi, B. (Ed) *Context and Consciousness. Activity Theory and Human-Computer Interaction.* The MIT Press

Kyng, M. (1995) Making Representations Work. in *Communications of the ACM* Vol.38. No. 9.

Laird, J. and Rosenbloom, P. (1987). SOAR: An Architecture for General Intelligence. In *Artificial Intelligence* , 33: 1-64. Quoted by Franklin (1995).

Lakoff, G. (1987). Women, Fire and Dangerous Things: What Categories Reveal About the Mind . Chicago University Press.

Lakoff, G. (1988). Cognitive Semantics. In Eco, U., Santambrogio, M. & Violi, P (Eds.). *Meaning and Mental Representations* . Indiana University Press.

Lakoff, G. (1993). Contemporary theory of metaphor. In Ortony, A. (Ed). *Metaphor and thought* . Cambridge University Press.

Lakoff, G, and Johnson, M. (1980). Metaphors We Live By. University of Chicago Press.

Lakoff, G, and Johnson, M. (1999). Philosophy in the Flesh: The Embodied Mind and Its Challenge to Western Thought. Basic Books. Perseus Group Books.NY.

Langacker, Ronald. (1987). Foundations of Cognitive Grammar. Vol. 1. Stanford University Press.

Langacker, Ronald. (1991). Foundations of Cognitive Grammar. Vol. 2. Stanford University Press.

Lano, K. and Haughton, H. (1994). Object-Oriented Specification. Case Studies. Prentice Hall.

Laurel, B. (1993). Computers as Theatre . Addison-Wesley.

Lave, J. & Wenger, E. (1991). Situated Learning: Legitimate Peripheral Participation. Cambridge, England: Cambridge University Press.

Leont'ev, A. N. (1978) Activity, Consciousness and Personality. Prentice Hall; Englewood Cliffs, NJ.

Love, T. (1993). Object Lessons. SIGS Books.

Luff, P. et al. (Eds). 1990. Computers and Conversation. Academic Press.

Lund, A. and Waterworth, J. (1999). Experiential Design: Reflecting embodiment at the human-computer interface. http://www.informatik.umu.se/~jwworth/MetaDesign.html

MacBreen, P (1999). See Web site: <http://c2.com/ cgi/wiki?UserStory

Mack, R. (1995). Discussion: Scenarios as Engines of Design. In Carroll, J (Ed.). *Scenario-Based Design* . John Wiley & Sons, Inc.

Maes, P. (1994). Agents that Reduce Work and Information Overload. *In Communications of the ACM.* Vol. 37. No. 7.

Maes, P. (1997). Pattie Maes on Software Agents: Humanizing the Global Computer. In *IEEE Internet Computing.* 1. 4. pp. 10-19. July-August.

Malcolm, G and Goguen, J. (1998) Signs and Representations: Semiotics for User Interface Design. In *Visual Representations and Interpretations*, Paton, R and Nielson, I (Eds), Springer Workshops in Computing. pp 163-172.

Mandrioli, D. and Meyer, B. (1992). Advances in Object-Oriented Software Engineering. Prentice Hall.

Margolis, N. (1989). IBM repository fog burning off. Computerworld. April 11.

Martin, J.A. (1989). HP boosts minis across the board. Computerworld. April 11.

Martin, J. and MacClure, C. (1988). Structured Techniques. The basis for CASE. Prentice-Hall.

Martin, J. and Odell, J. (1992). Object-Oriented Analysis and Design. Prentice Hall.

Mayr, E. (1984). Species Concepts and their Applications". In Sober, E. (Ed.), *Conceptual Issues in Evolutionary Biology.* Cambridge. Mass.:MIT Press.

Meyer, B. (1988). Object-Oriented Software Construction. Prentice Hall.

Minsky, M. (1994). A Conversation with Marvin Minsky About Agents. (Conversation with Doug Riecken). In *Communications of the ACM.* Vol. 37. 7.

Monin, N., and Monin J.(1994). "Personification of the Computer: A Pathological Metaphor in IS". In *SIGCPR 94* . pp. 283-288

Nakakoji, K. (1994). Crossing the Cultural Boundary. Byte. June. pp. 107-109

Nakakoji, K. (1996). Beyond Language Translation: Crossing the Cultural Divide. *In IEEE Software.* November 1996. pp. 42-45.

Narayanan, S. (1997). Knowledge-based Action Representations for Metaphor and Aspect (KARMA). PhD Thesis (Computer Science), University of California at Berkeley.

Nardi, B. (!995) Some reflections on Scenarios. In Carroll, J. M. (1995) Scenario-based Design John Wiley

Nardi, B. (1996a). Activity Theory and Human-Computer Interaction. In Nardi, B. (Ed) *Context and Consciousness. Activity Theory and Human-Computer Interaction.* The MIT Press.

Nardi, B. (1996b). Studying Context. In Nardi, B. (Ed) *Context and Consciousness. Activity Theory and Human-Computer Interaction.* The MIT Press.

Nardi, B. (Ed.). (1996c). Context and Consciousness. Activity Theory and Human-Computer Interaction. The MIT Press.

Nardi, B. and Zarmer, C. (1993). Beyond Models and Metaphors: Visual Formalisms. In User Interface Design. In *Journal of Visual Languages and Computing* . 4, pp. 5-33.

Nielsen, J. (1993). Usability Engineering: AP Professional.

Norman, D. (1983). Some Observations on Mental Models. In Gentner, D and Stevens, A. (Eds.). *Mental Models* . Hillsdale, NJ. Lawrence Erlbaum, Associates.

Norman, D. (1995). A Conversation with Don Norman, by John Rheinfrank. In *Interactions*. Vol. II. 2. Publication of the ACM. April 1995.

OMG (1992). Object Management Group: The Common Object Request Broker Architecture and Specification. OMG Document Number 91.12.1 Revision 1.1.

OMG (1999). Object Management Group: Unified Modeling Language Specification (draft). Version 1.3.

Ortony, A. (Ed.) (1993). Metaphor and Thought. 2nd. Edition. Cambridge University Press.

Parsons, J. And Wand, Y. (1997) Choosing Classes in conceptual modeling. In *Communications of the* ACM 40 (6) 63 - 69, June

Pree, W. (1995). Design Patterns for Object-Oriented Software Development. Addison-Wesley.

Preece, J. et al (1994). Human-Computer Interaction. Addison-Wesley.

Presmeg, N. (1997). Reasoning With Metaphors and Metonymies in Mathematical Learning. In L. English (Ed.), *Mathematical Reasoning: Analogies, Metaphors and Images*. Lawrence Erlbaum Associates.

Putnam, H. (1975) Mind, Language and Reality. Philosophical Papers, vol. 2. Cambridge University Press.

Putnam, H. (1981) Reason, Truth and History. Cambridge University Press.

Ramesh, B. (1998) Factors Influencing Requirements Traceability Practice. In CACM, 41, 12, pp: 37-41.

Reddy, M. (1993). The conduit metaphor: A case of frame conflict in our language about language. In *Metaphor and Thought* . Ortony, A. (Ed.). 2nd. edition. Cambridge University Press.

Rohrer, T. (1995). Feelings Stuck in a GUI web: Metaphors, image-schemata, and designing the human computer interface.

Rumbaugh, J. et al. (1991). Object-Oriented Modeling and Design. Prentice Hall.

Sachs, P. (1995). Transforming Work: collaboration, Learning and Design. In *Communications of the ACM*. Vol.38. No. 9.

Schank, R. (1990) Tell Me a Story. Narrative and Intelligence. Northwestern University Press. Evanston, Illinois.

Schneider, G. and Winters, J. (1998). Applying Use Cases. A Practical Guide. Addison-Wesley.

Schön, D. and Bennett, J. (1996). Reflective Conversation with Material. An Interview with Donald Schön and John Bernnett. In Bennet, J., De Young, L and Hartfield, B. In *Bringing Design to Software*. Winograd, T. (Ed.). Addison-Wesley.

Sfard, A. (1997) Commentary: On Metaphorical Roots of Conceptual Growth. In L. English (Ed.), *Mathematical Reasoning: Analogies, Metaphors and Images*. Lawrence Erlbaum Associates.

Shlaer, S. and Mellor, S. (1992). An Object-Oriented Approach to Domain Analysis. In Mandrioli and Meyer (Eds.) *Advances in Object-Oriented Software Engineering*. Prentice Hall.

Shneiderman, B. (1992). *Designing the User Interface. Strategies for Effective Human-Computer Interaction* . 2nd edition. Addison-Wesley.

Siegel, D. (1996). Creating Killer Web Sites: The Art of Third-Generation Site Design. Hayden Books.

Siegel, D. (1997). Secrets of Successful Web Sites: Project Management on the World Wide Web. Hayden Books.

Simon, H. and Newell, A. (1958). Heuristic Problem Solving: The next Advance in Operations Research. In *Operations Research* , 6: 1-10. Quoted by Franklin (1995).

Sommerville, I. (1992). Software Engineering. 4th Edition. Addisson Wesley.

Sontag, S. (1988). Illness as Metaphor . Farrar Straus & Giroux.

Steels, Luc. (1995). Building Agents out of Autonomous Behavior Systems. In Steels, L. and Brooks, R. (eds.). *The Artificial Life Route to Artificial Intelligence. Building Embodied, Situated Agents* . Lawrence Erlbaum Associates.

Suchman, L. (1987). Plans and Situated Actions. Cambridge University Press.

Sweetser, E. and Fauconnier, G. (1996). Cognitive Links and Domains: Basic Aspects of Mental Space Theory. In Fauconnier, G. and Sweetser, E. (Eds), *Spaces, Worlds and Grammar*. The University of Chicago Press.

Taylor, D. (1992). Object-Oriented Information Systems. Wiley.

Tufte, E. (1983). The Visual Display of Quantitative Information. Graphics Press. Cheshire, Connecticut.

Tufte, E. (1997). Visual Explanation: Images and Quantities, Evidence and Narrative. Graphics Press. Cheshire, Connecticut.

Turner, M. (1991). Reading Minds: The Study of English in the Age of Cognitive Science. Princeton University Press. Princeton, New Jersey. (First paperback printing, 1994).

Turner, M (1994). Design for a Theory of Meaning. In Overton, W and Palermo, D. (Eds) *The Nature and Ontogenesis of Meaning*, Lawrence Erlbaum Associates. pp 91-107.

Turner, M. (1996). The Literary Mind . Oxford University Press.

Turner, M (1998). Figure. In Katz, A., Cacciari, C., Gibbs, R. Jr. and Turner, M. *Figurative Language and Thought*. Oxford University Press.

Turner, M. and Fauconnier, G. (1995) Conceptual Integration and Formal Expression. In *Metaphor And Symbolic Activity*. 10(3), 183-204. Lawrence Erlbaum Associates, Inc.

Varela, F. (1995). The Re-Enchantment of the Concrete. In Steels, L. and Brooks, R. (eds.). *The Artificial Life Route to Artificial Intelligence. Building Embodied, Situated Agents* . Lawrence Erlbaum Associates.

Varela, F., et al. (1991). The Embodied Mind. The MIT Press. (4th printing. 1995).

Viller, S. and Sommerville, I. (1999). Coherence: An Approach to Representing Ethnographic Analyses in System Design. *In HCI. Special Issue about Representations in Interactive Systems Development*. Volume 14, Number 1 & 2. Lawrence Erlbaum Associates, Publishers.

Wayner, P. (1995). Agents Unleashed: A Public Domain Look at Agent Technology. Academic Press Professional.

Williams, M et al. (1983). Human Reasoning About a Simple Physical System. In Gentner, D and Stevens, A. (Eds.) *Mental Models* . Hillsdale, NJ. Lawrence Erlbaum, Associates.

Winograd, T. (1995). From Programming Environments to Environments for Designing. In *Communications of the ACM*. Vol. 38, No. 6.

Winograd, T. (1996). Design of the Conceptual Model. An Interview with David Liddle. In Bennet, J., De Young, L and Hartfield, B. *Bringing Design to Software*. Winograd, T. (Ed.). Addison-Wesley.

Winograd, T. and Flores, F. (1987). Understanding Computers and Cognition: A New Foundation for Design. Addison-Wesley

Wirfs-Brock, R. et al. (1990). Designing Object-Oriented Software. Prentice Hall.

Yourdon, E. (1989). Modern Structured Analysis. Prentice Hall.

Yourdon, E. (1994). Object-Oriented Systems Design: An Integrated Approach. Prentice Hall.