Zooming In and Out in Requirements Engineering

Manuel Imaz

BlendMind Madrid, Spain
imaz@mac.com

Abstract. In this paper we present some cognitive guidelines to move from higher degrees of abstraction to lower, more concrete degrees of abstraction, and reciprocally. For a time, in Requirements Engineering this approach was called the *top-down method* and it was intended to be used as a general method in the first steps of specifying a system: the whole application was decomposed using this method. In everyday analysis and design, it continues to be a fundamental way of separation of concerns, applying the motto of *divide and conquer*. In order to divide, we must visualize the whole system in terms of lower level components: processes, use cases or user stories. But the important question is that zooming in can not start –in Requirements Engineering– at every level but at a given level of abstraction, usually at the level where some event will trigger the execution of some processes: such is the case of user stories and use cases. Another important issue we want to stress is the essential role that stories play in our cognition: this is why user stories –as well as use cases– have been and continue to be intensively used in software development.

Keywords: zooming in, zooming out, decomposition, abstraction level, top-down.

1 Introduction

The traditional structured analysis and design methods which appeared in the last '70 promoted –for requirements engineering– the top-down method, which allowed to move from higher levels of abstraction to intermediate ones and finally to lower levels of abstraction. This idea, inherited from the structured programming field, had the underlying conception that moving between abstraction layers was something that we could arbitrary do without any limitation.

Unfortunately for the top-down method, some authors [22] detected it was, in fact, a *myth*. In fact, the *method* can be applied by analysts when they have already gained experience with previous similar systems they have already analyzed and designed (at least in the early steps), so they are, in fact, applying their previous knowledge and are not discovering the internal structure of new systems by going top down. Analysts had, this way, the delusion of applying a real method instead of observing that they were working using their skills

X. Franch and P. Soffer (Eds.): CAiSE 2013 Workshops, LNBIP 148, pp. 174–185, 2013.

and retrospectively adjudicating the *success* to the supposedly well established top-down method.

In Imaz [10] we stated that:

> *Starting at a high level, we conceive a whole large system implying composition by other subsystems. We are a long way from basic-level categories. It is at this level where there is a risk of loosing one's way, as Ed Yourdon ([22], p. 360) has pointed out in relation to the top-down problem: "analysis paralysis," the "six analyst" phenomenon, or the "arbitrary physical partitioning". Yourdon's proposal is to use a middle-out method, in which the middle is closer to basic-level categories, and from where to go down and up looking for concepts that fit the analysis purpose better (see fig. 1). Moreover, somebody who successfully applies the top-down method is a person who has sound skills in the specific domain that one is working on.* [10] p. 86

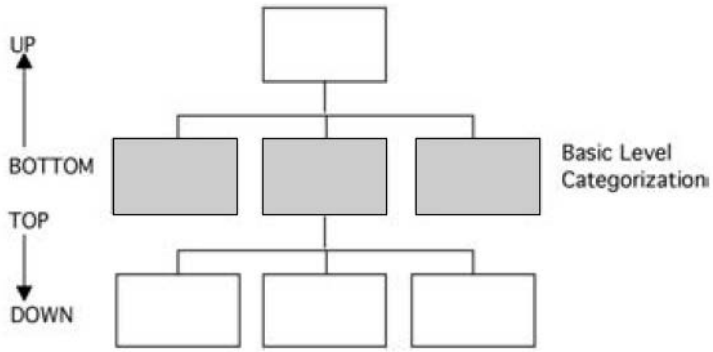

Fig. 1. Two different movements: bottom-up and top-down

What Yourdon pointed out about top-down method was in line with the categorization theory that Eleanor Rosch established in the mid-1970s, for which she tried to offer an empirical foundation. One important result of Rosch's [12] [16] work is the concept of basic-level categories. Basic-level categories are at a middle level of abstraction. They refer to subcategories or category members that have a special cognitive status: that of being a best example.

Basic-level categories are basic in a number of respects. From a perceptual point of view, basic-level categories map to an overall shape, to single images with fast identification. From a functional point of view, basic-level categories are associated with a general motor program –the way in which our brains control our movements– for using, for example, chairs and tables. In terms of mental images, we can form a general mental image of basic-level categories like chairs or dogs. But we cannot form a mental image for superordinate categories like furniture or animals.

Matching the points raised by Yourdon (what we can categorize are processes at an intermediate level, which are launched by some concrete events), in the middle '90 was coined the concept of *use case* and with the Agile movement appeared a similar construct: that of *user stories*. Both constructs appear as solid basic-level categories –at an intermediate level– and their success until now provide the empirical evidence that they are solid requirement engineering constructs due to this intermediate level conceptualization. In the following sections we will give some additional cognitive arguments and foundations.

2 Cognitive Semantics

In this section we will show how the concept of use case is based on some cognitive semantics' ideas of perspective, viewpoint, focusing and so on.

Leonard Talmy states that Cognitive Semantics is the study of the way conceptual content is organized in language. In Talmy's view, a sentence (or other portion of discourse) does not objectively represent its referent scene –the represented scene is not something out there in the world–, but it evokes in the listener a cognitive representation, defined as an emergent, compounded by various cognitive processes out of the referential meanings of the sentence elements, understanding of the present situation, general knowledge, and so on [19] p. 93, note 2.

Historically, science has tried to show its objectivity eliminating the subject from the scientific discourse. The same effort has been assumed by the software engineering community when using a disembodied discourse, but this intention is unmasked when analyzing in detail some conceptual structures in which the subject surreptitiously reappears, as –for example– the concept of *perspective* implies an object and a subject and the concept of *focusing* implies that the subject is using his visual capacity (as Langacker defines it: "what we choose to look at" [13]).

Another aspect of cognitive semantics is that the conceptual structure is *embodied*, that is, the nature of the human mind is largely determined by the form of the human body. But the *form* of the human body must be understood in a broad sense, meaning the human being in an environment, in a given situation –cultural, social, and so on– as some concepts of CS imply.

In terms of literary analysis it is said that a narration has a content but, equally important, it has also a style. Analyzing the question from the point of view of Cognitive Semantics (in particular, Langacker's Cognitive Grammar), it can be stated that:

> *In viewing a scene, what we actually see depends on how closely we examine it, what we choose to look at, which elements we pay most attention to, and where we view it from. The corresponding labels I will use, for broad classes of construal phenomena, are* ***specificity***, ***focusing***, ***prominence***, *and* ***perspective***. *They apply to conceptions in any domain.* [13] p. 66 (bolds in the original)

Associated with the concept of scene, for example, we have the concepts of focusing, perspective and so on. It is evident that a perspective implies a subject observing a scene from a given point of view, that is, a subject in a given situation.

The concept of *perspective* allows us to make a difference between observing a computer system in the organization or observing the enterprise itself, and conceptualizing the internals of the computer system or the business processes running into the enterprise. The concept of perspective is represented in (Fig. 2):

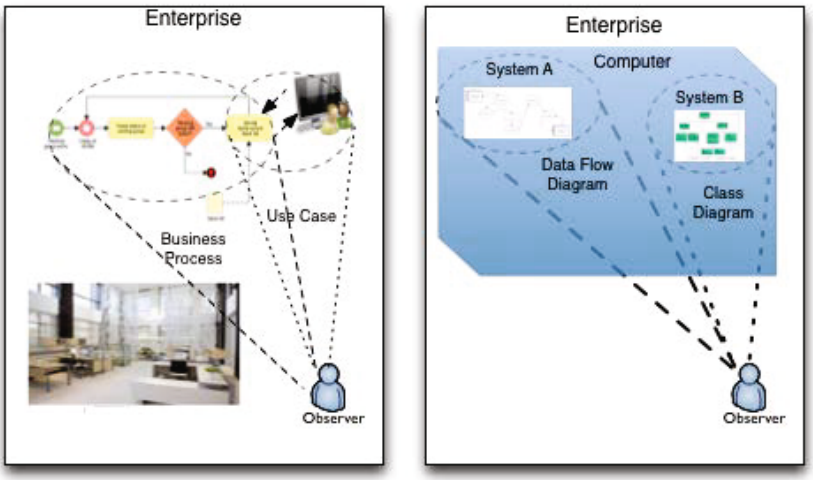

Fig. 2. Two different scenes: the enterprise and two systems into a computer

When focusing on the computer system we need additional concepts in order to use different categories applied to the same system. Besides what we choose to look at –focusing– we need to take into consideration which elements we pay most attention to or *prominence*, in particular one sort of prominence: *profiling*. Langacker states that:

> *As the basis for its meaning, an expression selects a certain body of conceptual content. Let us call this its conceptual* ***base****. Construed broadly, an expression's conceptual base is identified as its maximal scope in all domains of its matrix (or all domains accessed on a given occasion). Construed more narrowly, its base is identified as the immediate scope in active domains –that is, the portion put 'onstage" and foregrounded as the general locus of viewing attention. Within this onstage region, attention is directed to a particular substructure, called the* ***profile****.* [13] p. 66 (bolds in the original)

Langacker illustrates the concept of profile using the week example. The concept of week is a seven-day cycle, which is the *base* and may be *profiled* in different segments called Monday, Tuesday, Wednesday, etc.

In our example, one conceptual base is the computer system and the profile may be a process or a data flow –a particular substructure– or an object in another profile. That is, the same conceptual base may be considered in terms of different profiles: data flows and processes or objects, for example.

Both ways of categorizing the computer system are different types of conceptual integrations or blends of mental spaces. 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'*. [6] A conceptual integration or blend [7] [10] is an operation that could be applied to a couple of input spaces, which gives as a result a blended space or blend. The blend receives a partial structure from both input spaces but has an emergent structure of its own.

On the other hand, a conceptual base such as a business process, may be profiled in terms of tasks, decision points, etc., or may be also profiled as use cases, that is, subsets of the business process in which some actors –users– interact with software components in order to achieve a goal.

3 The Essential Role of Stories

The same way that use cases are based on some cognitive processes, use stories are also based on an important cognitive concept: that of stories. There is an extensive literature about the significant role of stories in cognitive sciences. Even business reviews like Harvard Business Review have published many articles about stories [4] [17] in the last years. The reason for not having paid enough attention to stories reflects prejudices about literature and its non scientific nature.

In the mid-nineties, Turner [20] and Fauconnier have developed some observant theories about stories and mental spaces. In particular, Turner has illustrated how an aspect of our mind –the literary side– provides us a knowledge that sometimes lacks a more scientific discipline.

> *The literary mind is not a separate kind of mind. It is our mind. The literary mind is the fundamental mind. Although cognitive science is associated with mechanical technologies like robots and computer instruments that seem unliterary, the central issues for cognitive science are in fact the issues of the literary mind.*
> ***Story** is a basic principle of mind. Most of our experience, our knowledge, and our thinking is organized as stories. The mental scope of story is magnified by **projection** –one story helps us make sense of another. The projection of one story onto another is **parable**, a basic cognitive principle that shows up everywhere, from simple actions like telling time to complex literary creations like Proust's **A la recherche du temps perdu.*** [20] p. V (bolds are italics in the original)

Turner continues telling us that his book is an attempt to show how wrong the common view is and to replace it with a view of the mind that is more

scientific, more accurate, more inclusive, and more interesting, a view that no longer misrepresents everyday thought and action as divorced from the literary mind.

We constantly witness at stories that involve actors engaged in bodily actions, that is, small stories of events in space: the wind blows clouds through the sky, a child throws a ball, a girl pours water from a glass.

When we watch someone sitting down into a chair, we see what physics cannot recognize: an animate agent performing an intentional act involving basic human-scale categories of events like *sitting* and objects like *chair*. So what we, as human, additionally see is the *intention* of sitting in the animate agent.

In Requirements Engineering, the same idea of stories has been resumed in the agile movement, to the point that Leffingwell ([14] p. 56) points out that "the teams backlog consists of all the work items the team has identified. In the meta-model, we generically call these work items stories (some call them backlogs or backlog items)". The author, before using the concept of user stories defines generically *stories* as a work item.

These basic ideas of stories are resumed in the concept of user stories. A user story –used for requirements elicitation in agile and lean methods– is a particular way of describing functionality that will be valuable to either a user, developer or purchaser of a system or software. User stories are composed, in agile methods, of three aspects [2] p. 4:

- a **written description** of the story used for planning and as a reminder
- **conversations** about the story that serve to flesh out the details of the story
- **tests** that convey and document details and that can be used to determine when a story is complete

These three aspects are also called Card, Conversation, and Confirmation. The Card may be the most visible manifestation of a user story, but it is not the most important. It represents customer requirements rather than document them. This is the suitable way to think about user stories: while the card may contain the text of the story, the details are worked out in the Conversation and recorded in the Confirmation.

It is important to point out that conversations are not a detailed specification of the user story, but are merely general points that act as reminders in order to later resume the details of the story.

Conversation represents a discussion between the development team, customer, product owner, and other stakeholders, which is necessary to determine the more detailed behavior required to implement the intent. In other words, the card also represents a "promise for a conversation" about the intent. [15], p. 5.

According to a proposed template to create user stories, each story can be written in the following format:

I as a (*role*) want (*function*) so that (*business value*)

In the previous section we have seen that an expression selects a certain body of conceptual content: the conceptual base. But attention may be directed to a particular substructure, the profile.

4 Basic Level Candidate Elements

In this section we intend to show why use cases and user stories are at a basic level of categorization. We may consider three types of scenes when viewing a system: the system itself, viewed from an *external perspective*; the classical *internal perspective*, used in the old top-down method (on which ovals correspond to subprocesses) or in modern views based on UML views and the *scenario perspective*, used on use cases and user stories perspectives.

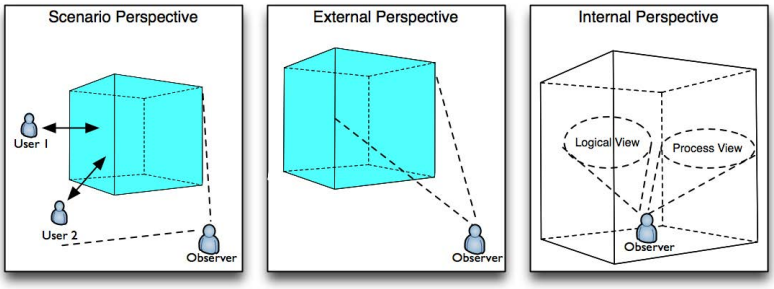

Fig. 3. Two different scenes: the enterprise and two views of the system

In classical approaches –as the top-down or previous– the external perspective represents the system such as called by its name. In Langacker's terms, we are ***focusing***– what we choose to look at– on the system from an external ***perspective*** –where we view it from.

But while in classical methods we started doing a zooming in –going directly to an internal perspective– to see the components of the system, in modern approaches –use cases and user stories– we do a zooming out to cover the users involved. This makes both pieces of elicitation essential elements in requirements engineering, as they are the equivalent of basic-level categories in cognitive semantics.

The question would be: why use cases and user stories are the basic-level categories in requirements engineering? Because they are connected to bodily motor schema: those of interacting with machines, devices and particularly, digital devices. We can easily imagine interactions with pieces of software but not with abstract subprocesses that constitute the whole system we are going to implement. In use cases and user stories somebody –the actor or the user– is the agent that triggers processes in order to obtain some valuable results.

They essentially decrease the complexity in terms of abstraction, as when associated to bodily experience, there is a more direct visualization of requirements in terms of previous similar digital devices. Another argument in favor of considering use cases and user stories as such basic-level categories is the generalized use of them in most classical, agile and lean developments.

Another important reason is the direct communication obtained with final users and the ease with which they understand the goal of such stories.

Even if both use cases and user stories may be defined as a scenario perspective, there are some differences. The main is the profile, the particular substructure to which is directed the attention: while in user stories we pay attention to the *user* and his intention (goal or business value) such as can be seen in Fig. 4, in use cases the attention is directed to the *technical structure specification*.

Sometimes we say that user stories are *business* directed but use cases are *system* directed. In fact, some authors define two types of use cases: Fowler, for example, recognizes that a *system use case* is an interaction with the Software, whereas a *business use case* discusses how a business responds to a customer or an event [8], p. 103.

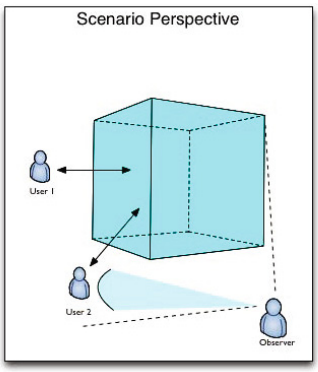

Fig. 4. In user stories the attention is directed to the user

5 Use Cases vs. User Stories

It would be interesting to consider aspects of both constructs. Leaving aside the different profiles between user stories and use cases (paying attention to the user or to the technical aspects) the other major difference is the degree of specification. When zooming in a user story, we may find some points that are not stated in a procedural way. In [5] we have some stories and conversations. One of them is the story:

> *As a Creator, I want to upload a video from my local machine so that any users can view it.*

Conversation is an open dialogue between all the team members and the client. Anyone can raise questions, ask for things to be clarified, and the answers can be recorded down as bullet points for later reference.

- *The "Upload" button will be a persistent item on every page of the site.*
- *Videos must not be larger than 100MB, or more than 10 minutes long.*

- *File formats can include .flv, .mov, .mp4, .avi, and .mpg.*
- *Upload progress will be shown in real time.*

As we can observe, these are points to be remembered, but are not a specification in natural language. They are part of a collective memory: team members and user stories.

On the other side, use cases have an associated structured template where to include a specification. In the template there are sections as: use case title, primary actor, level, precondition, postcondition, main success scenario, extensions, etc.

In this differentiation between user stories' and use cases' internals, the intentions of the team members are critical , as we can see in Fig. 5. A collaborative organization –such as in agile or lean teams– allows some decisions and specifications to be deferred until the moment they will be implemented. It is the case of a just-in-time implementation, considering each of the points of the conversation.

Indeed, in use cases the team group has decided to postpone the implementation to a later time, for different team members, as it usually occurs with more classical development processes. And in such a case the right decision is to specify the use cases in order to get the right implementation. When zooming in a user story we find some comments in the form of points to consider, and when zooming in use cases we find structured specifications in a procedural mode.

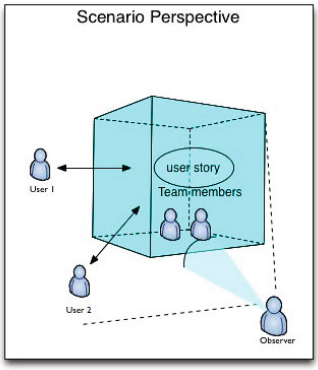

Fig. 5. The intentions of the team members are critical

Even if we limit –in this paper– the scope of use cases to agile and lean methods, we must remember that use cases were incorporated in the Rational Unified Process, a classical development method. We must also remember that use cases are also applied to large and complex systems, as exposed by Rotem-Gal-Oz [18]. But, unlike the purpose of Rotem-Gal-Oz (*[my] paper is not about explaining why use cases should be used for enterprise requirements analysis*), our aim is clearly to explain why use cases should be used.

6 Conclusions

In the early '70 Dahl, Dijkstra and Hoare published Structured Programming [3], where the authors already stated that the only efficient way to deal with complicated systems is in a hierarchical fashion. *The dynamic system is constructed and understood in terms of high level concepts, which are in turn constructed and understood in terms of lower level concepts, and so forth* [3], p. 176. The previous year, Wirth [21] had published a paper where he proposed that *the program is gradually developed in a sequence of refinement steps.* This is the essence of what was later named the top-down method.

In the following years, this approach was rapidly generalized to design and analysis, in what was called *structured design and analysis.* The problem with the approach is that it was not scalable to the analysis and, in particular, to zooming in from the higher level of abstraction –the system or software– to an intermediate level, as stablished by Yourdon when he points out the "analysis paralysis," the "six analyst" phenomenon, or the "arbitrary physical partitioning".[22].

In the early '90 was formulated the concept of use case and in these same years appears the agile movement, which coined the concept of user stories. The long and strong history of these constructs –twenty years for use cases– is an empirical evidence of their success and that they are at the right level of abstraction: basic level.

A cognitive point of view about two of the most frequently used constructs of Requirements Engineering shows that this popularity is due to certain cognitive foundations. One important aspect of user stories and use cases is that they both are packages of requirements determining a useful purpose for the user. The other important point we have seen here is that both are at an intermediate level of abstraction, corresponding to what has been named basic-level categories in cognitive sciences.

The important fact is that basic-level categories are closely linked to motor capacity, which makes them immediately understandable: the bodily motor experiences are those of interacting with machines and devices, so there is a direct projection from the bodily –concrete– experience to the requirement artifacts we are defining. If we zoom in –in terms of a classical top-down method– to more details, we get complex sets of isolated requirements that need to be packaged into meaningful units that help to give them some meaning. But zooming in –in terms of user stories– we get conversations and the confirmation (test case) or, in terms of use cases, a semi-formal specification, depending on the applied profile.

This is the reason of their popularity and the long and solid life they have maintained through many years in agile and lean methods. Above these constructs may be some abstract components, derived from the aggregation of user stories (or use cases) that can accommodate to more general categories. The top-down approach called these abstract components subsystems or processes in general. Such components may be rebuild from use stories or use cases using a zooming out movement (Yourdon proposed to speak of middle-out method, from which we move bottom-up) and categorizing them as bigger groups, but they can not be *discovered* as the top-down method sometimes intended to do.

As future work, we plan to evaluate the possibility of extending the concept of use case –as based on human body schemas– to more formal constructs, such as used in complex systems like operating systems, mission critical systems, etc. Operating systems, for example, are based on concepts like tasks, jobs or commands. Considered in detail, these concepts are metaphorical extensions of human activities and we may investigate the concrete activities which constitute the source mental space of such metaphors. The immediate observation we can make is that tasks are most likely abstract views of use cases, in which actors correspond to events that trigger the tasks, the task itself correspond to a use case and the result of the execution is common to both constructs.

As Turner says: *in our small stories, we distinguish objects from events, objects from other objects, and events from other events. We categorize some objects as belonging to the category* ***person*** *and other objects as belonging to the category* ***chair***. [20] pp. 14-15 (bolds are italics in the original). That is, what is involved in stories, generically, are objects and events.

Acknowledgements. The author is grateful to Mauricio Milchberg for his revision of the manuscript and valuable comments.

References

1. Allwood, J., Gärdenfors, P. (eds.): Cognitive Semantics: Meaning and Cognition. John Benjamins Publishing Company, Amsterdam (1999)
2. Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley Professional (2004)
3. Dahl, O.-J., Dijkstra, E.W., Hoare, C.A.R.: Structured Programming. especially. Academic Press (1972)
4. Duarte, N.: Structure Your Presentation Like a Story. HBR (2012), `http://blogs.hbr.org/cs/2012/10/structure_your_presentation_li.html`
5. Dennis, S.: How to write meaningful User Stories (2010), `http://www.subcide.com/articles/how-to-write-meaningful-user-stories/`
6. Fauconnier, G.: Mappings in thought and language. Cambridge University Press, Cambridge (1997)
7. Fauconnier, G., Turner, M.: The Way We Think: Conceptual Blending and the Minds Hidden Complexities. Basic Books (2002)
8. Fowler, M.: UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd edn. (2003)
9. Imaz, M.: Abstracción y Conceptualización: Un Enfoque Cognitivo. 40 JAIIO, Jornadas Argentinas de Informática, 29 de Agosto al 2 de Septiembre de 2011, Córdoba, Argentina. Anales 40 JAIIO /CD40JAIIO/T2011/ASSE/603.pdf (2011)
10. Imaz, M., Benyon, D.: Designing with Blends. MIT Press (2007)
11. Kramer, J.: Is Abstraction the Key to Computing? Communications of the ACM 50(4), 37–42 (2007)
12. Lakoff, G.: Women, Fire, and Dangerous Things: What Categories Reveal about the Mind. University of Chicago Press, Chicago (1987)
13. Langacker, R.: Cognitive Grammar: A Basic Introduction. Oxford University Press, Inc., New York (2008)

14. Leffingwell, D.: Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional (2011)
15. Leffingwell, D., Behrens, P.: A User Story Primer. Whitepaper (2003), http://scalingsoftwareagilityblog.com/wp-content/uploads/2010/01/user-story-primer_1.pdf
16. Margolis, E., Laurence, S. (eds.): Concepts: Core Readings. MIT, Cambridge (1999)
17. McKee, R., Fryer, B.: Storytelling That Moves People HBR (2013), http://hbr.org/web/special-collections/insight/communication/storytelling-that-moves-people
18. Eotem-Gal-Oz, A.: Methodology for developing Use Cases for large systems? (2004), http://www.rgoarchitects.com/blog/content/binary/uc.pdf Methodology for developing Use Cases for large systems
19. Talmy, L.: Toward a Cognitive Semantics. Concept structuring systems, vol. 1. MIT Press, Cambridge (2000)
20. Turner, M.: The Literary Mind. Oxford University Press, Oxford (1996)
21. Wirth, N.: Program Development with Stepwise Refinement. Communications of the ACM 14(4), 221–227 (1971)
22. Yourdon, E.: Modern Structured Analysis. Prentice Hall, Englewood Cliffs (1989)