IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITB12684

This paper appears in the book, *Emerging Trends and Challenges in Information Technology Management, Volume 1 and Volume 2* edited by Mehdi Khosrow-Pour © 2006, Idea Group Inc.

Users as Subjects in and of Co-Design

Peter Rittgen, University College of Borås, 501 90 Borås, Sweden, T +46-33-435-5930, F +46-33-435-4007, peter.rittgen@hb.se

ABSTRACT

Co-design implies the active participation of all stakeholders in the system development process. Users have a double role in this being subjects both **in** and **of** co-design – they take an active part **in** the design and the system is designed **for** them. This requires a design language that allows users to make meaningful contributions. Common system design languages, such as UML, do not fulfil this requirement. Here we investigate to what extent metaphors can provide an appropriate communicative medium to increase user involvement and how we can translate them into more detailed designs.

INTRODUCTION

The term co-design refers to a concept that was elaborated in (Forsgren 1991 & 2005). It has its roots in the "systems thinking" approach as established by (Churchman 1968). His principal idea was that we can design an unlimited number of views on reality. They may differ in their granularity (level of detail), their perspective, their level of abstraction, and so on. But from Churchman's point of view this is not sufficient. We must also "calibrate" the viewing instrument (or measurement scale) to arrive at (or agree on) a view that is supposed to be implemented. This collective process of designing views and choosing the best one is called co-design. It has shaped the way we look at social systems in general and information systems in particular (Ackoff 1981, Checkland 1988, Mitroff and Mason 1981). Two dimensions can be identified determining four different roles in the design process: the subjects or objects **in** or **of** co-design. The development of an information system can be seen as a project where the participants co-design the system. The participants include system designers and users. System designers are "subjects **in** co-design". Users are subjects both **in** and **of** co-design because they participate **in** the design and they are also the ones the system is designed for (subjects **of** design). The information system is the result of this project and therefore the "object **of** design". The design languages (and other tools) used in this project are "objects **in** design". We focus on the latter by using metaphors as expressions in a general, albeit imprecise, design language and combining this with a more sophisticated design language (UML). This allows users to actively contribute to the design of "their" system.

A simple example of a metaphor is that of a CD player. Most people have experience in operating a hardware player so that its design can be used as a template for that of the software player. The user interface of the latter will thus include images of the knobs, slides, dials, buttons, lamps, displays and all other devices found on a hardware player and the user will be able to operate it without studying a manual. For more sophisticated information systems it can be hard to find a suitable metaphor and to translate it into system design. Design patterns can facilitate this by providing a general, reusable solution to a reccuring problem but they are often abstract and technology-oriented and do not appeal to intuition or common sense. That makes them unsuitable as a platform for communication between IS stakeholders. But design patterns make a valuable contribution to well-structured, reusable and less error-prone software. It is therefore worthwhile to investigate the translation of metaphors into design patterns to lower the communicative barriers between stakeholders and to allow them to join in the co-design of "their" system. To this end we first address the fundamental concepts of metaphors and design patterns and the respective languages. We proceed by specifying a translation based on structure-mapping theory and providing an example that employs the metaphor of the labour market to develop the architecture of an eService system.

METAPHORS

A metaphor is a figure of speech that involves a transfer of meaning between seemingly unrelated terms. When we say "The customer is king" we do not mean it literally but in the sense that we attribute to a customer qualities we associate with a king: influence, importance, power and so on. A metaphor can transfer complex meaning such as in that of a nation as "a ship of state". According to cognitive linguistics (Lakoff and Johnson 1980) a metaphor maps one conceptual space (source) onto another (target). As an example we might use the labour market with employers, job seekers, job announcements etc. to explain the architecture of a yet-to-be-built eService market with eService providers, eService clients, eService offers and so on.

The Metaphor Language (ML) is used to express both source and target. It must therefore be a simple language closely resembling our "intuitive" way of conceptualizing the world around us. A common approach is to view the world as a number of things (classes, objects, entities) that are related in some way. They can be on the instance or type level, i.e. they refer to individual things or sets of things, respectively. The same holds for relations between things (links, associations). A metaphor model in ML consists of *entity types* and *transfer types*. Entity types are sets of concrete things sharing common properties. Transfer types are ternary relation types where an entity of the second type is transferred from one of the first to one of the third. The first and third entity types must be *active*, i.e. their entities must be able to perform activities. An example of a transfer is an employer who makes a job offer to a job seeker.

Figure 1 shows the source of the labour market metaphor. Job seekers register at the placement service by giving information on themselves and their qualifications. Employers hand in a description of each vacancy (a list of required qualifications). The placement service compares required and provided qualifications and sends the employers information about matching candidates. Based on that employers can decide to make a job offer which the job seeker in turn can accept or decline.

DESIGN PATTERNS

The design patterns used in the development of object-oriented software were introduced by Gamma et al. (1995). They have been influenced

Figure 1. The labour market metaphor (source)

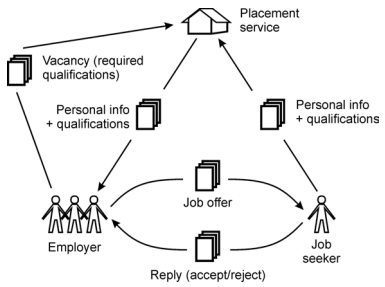


Table 1. Translation of some of the labour market terms

Source term	Target term
Employer	eService client
Job seeker	eService provider
Placement service	eService broker
Vacancy	eService request

Table 2. Mapping ML constructs onto UML

ML construct	UML construct
ML-entity type	UML-class
ML-active entity type	UML-active class
ML-transfer type	UML-association (ternary) + UML-operation (message)

strongly by Alexander's architectural design patterns where each pattern is defined as a three-part rule, which expresses a relation between a certain context, a problem, and a solution (Alexander 1979). The context describes the conditions under which the pattern can be applied. The problem is a "system of forces" that the solution is supposed to balance. According to Trætteberg (2000) a design pattern involves:

1. a description of a problem and a possible solution,
2. examples of usages of the solution in actual designs,
3. reasons for why the solution actually solves the problem,
4. criteria for when to use and when to avoid this particular solution and
5. relations to other design patterns.

In object-oriented design the solution is often described as a class diagram that can be accompanied by some interaction diagram. Sometimes the term design pattern is used to refer to the solution alone. We use the Unified Modeling Language (UML, OMG 2003) to express the solution part of design patterns, in particular the class and sequence diagrams of UML.

TRANSLATING METAPHORS INTO DESIGN PATTERNS

The translation of metaphors into design patterns is based on structure-mapping theory. It has originally been developed to describe the nature of analogies (Gentner 1983) but has later been adapted to metaphors (Gentner et al. 1988). It assumes that the likeness between source and target is not so much in the properties of the objects themselves but rather in their relations. In our example of the labour and eService markets an employer and an eService client have little in common but the relation between an employer and a job seeker on the one hand, and an eService client and an eService provider on the other hand is close: in both cases the second offers a service to the first. In structure-mapping we define interpretation rules that map knowledge about a source domain onto a target domain by relying on the syntactic properties of the knowledge domain alone and not on the specific content of the domains.

Figure 3. The class diagram of the eService pattern

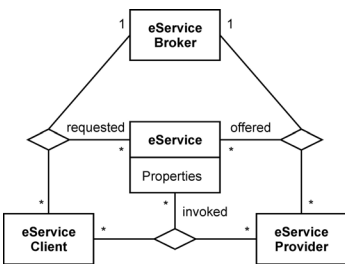
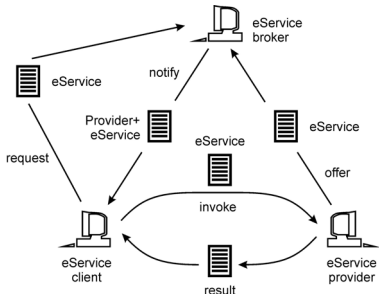


Figure 2. Metaphor target model of the eService market



The translation proceeds in two steps: metaphorical mapping and design mapping. The first maps terms from source to target largely conserving the structure of the source. The resulting model is still in ML but refers to the target (metaphor target model). Metaphorical mapping is supported by a translation table showing related source and target terms (see Table 1). The metaphor target model is shown in Figure 2.

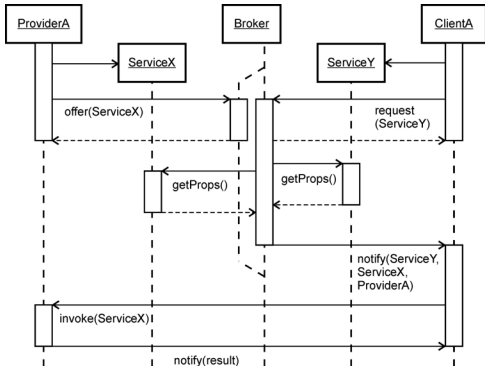
The second step, design mapping, translates the metaphor target model into the design pattern. This mapping is between different languages (ML and UML) and hence more complex. An analysis of the languages reveals that the constructs of the ML should be translated into UML according to Table 2.

The mapping from ML to UML does not cover all UML constructs. This means that we have to supply additional information that is not present in the metaphor. In particular we have to provide the order of messages, class attributes, multiplicities and objects for the sequence diagram. Figures 3 and 4 show the result of this step, the class and sequence diagrams,

CONCLUSION

Developing an information system is a complex task requiring the active collaboration of a number of stakeholders if it is to be successful. The stakeholders have to speak the same language to agree on some common design for the system. This process is called co-design (Liu et al. 2002, Forsgren 1991). To facilitate mutual understanding the common language we use must reflect the hetorogenity of the stakeholders' cultures, i.e. it must not be derived from any particular culture such as that of systems engineering. Hence UML is not suitable. But metaphors provide the required features because they resort to knowledge that is rooted in common sense and therefore shared by everybody. We have introduced a simple language for expressing metaphors and suggested a way to translate them into patterns that can be used for system design. Finding a suitable metaphor is still a creative act (or maybe a stroke of genius) that can hardly be supported in a systematic way. But once it has been found we can provide some help in performing the ensuing steps.

Figure 4. The sequence diagram of the eService pattern



REFERENCES

Ackoff, RL (1981) Creating the corporate future. New York: Wiley.

Alexander C (1979) The Timeless Way of Building, Oxford University Press, New York

Checkland, PB (1988) Soft systems methodology: An overview. J. of Applied Systems Analysis, 15, 27-30.

Churchman, CW (1968) The Systems Approach. New York: Dell Publishing.

Forsgren O (1991) Co-constructive computer applications: Core ideas and some complementary strategies in the development of a humanistic computer science. In: Bazewicz M (ed) Information systems architecture and technologies - ISAT'91. Politechnika Wroclawska, Wroclaw, pp 45-53

Forsgren O (2005) C West Churchman and The New World of Co-Design. In: McIntyre-Mills, J (ed) Rescuing the Enlightenment from Itself - Critical and Systemic Implications for Democracy. Springer, Berlin, pp 37-44

Gamma E, Helm R, Johnson R and Vlissides J (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA

Gentner D (1983) Structure mapping: A theoretical framework for analogy. Cognitive Science 7(2), 155-170

Gentner D, Falkenhainer B and Skorstad J (1988) Viewing metaphor as analogy: The Good, The Bad and The Ugly. In: Helma DH (ed) Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science and Philosophy. Kluwer, Dordrecht, The Netherlands, pp 171-177

Lakoff G and Johnson M (1980) Metaphors we live by. The University of Chicago Press, Chicago

Liu K, Sun L and Bennett K (2002) Co-Design of Business and IT Systems. Information Systems Frontiers 4(3), 251-256

Mitroff, II and Mason, RO (1981) Creating a dialectical social science. Dordrecht: Reidel.

OMG (2003) Unified Modeling Language Specification: Version 1.5. Object Management Group, Needham, MA, USA. Online version available at http://www.omg.org/docs/formal/03-03-01.pdf

Trætteberg H (2000) Model based design patterns. Position paper, Workshop on User Interface Design Patterns, CHI'2000, The Netherlands.