Modelling requirements with UML: a rigorous approach
(Doing requirements well with UML)

Half-day tutorial proposal

Proposers

Prof. Luigi Lavazza[1,2] and Dr. Vieri del Bianco[1], PhD

(1) Università degli Studi dell'Insubria
Dipartimento di Informatica e Comunicazione
Via Mazzini, 5 – 21100 Varese (Italy)

(2) CEFRIEL
Via Fucini, 2 – 20133 Milano (Italy)

E-mail: luigi.lavazza@uninsubria.it, delbian1970@yahoo.it

Tel. +39-0332-218930, +39-0332-218938

Abstract

The tutorial addresses the problems connected with requirements modelling in a UML-based development process. UML supports requirements modelling by means of use cases. This practice suffers from several limitations, especially as use cases are quite informal descriptions –thus allowing ambiguities and misinterpretation of requirements– and they are not object-oriented –thus making traceability of requirements in object-oriented models problematic.

Rigorous approaches –not based on UML– were proposed, but they did not achieve a great popularity, mainly because they are considered not promptly applicable in UML-based development processes, and consequently hardly productive in the short term.

The proposed tutorial promotes the application of the Reference Model for Requirements and Specifications [2] and the usage of Michael Jackson's Problem Frames [1].

Problem frames drive developers to understand and describe the problem to be solved, which is crucial for a successful software development process. Moreover, Problem Frames provide a framework to arrange the elements of the problem and solution domains; the final purpose of such detailed framework is to let developers write requirements and specifications in an ordered, clear and rigorous way.

The goal of this tutorial is to show how the mentioned rigorous approaches to requirements engineering can be applied in the context of a software development process based on UML. In particular, it is shown how familiar UML constructs (like class and state diagrams) can be used in a disciplined way in the construction of models that represent properly the problem domain, the user requirements, and the specifications of the software system.

The final objective is to increase the quality of requirements models, while making them suitable to drive UML-based software development.

Tutorial aims, objectives and scope

Motivations: problems addressed

The tutorial addresses the problems connected with requirements modelling in a UML-based development process. UML supports requirements modelling by means of "use case" diagrams. Unfortunately, use cases suffer from several limitations, amply described in the literature [1][11][12].

The main problem with use cases is that they are completely informal. Often the information provided by use cases is complemented by explanatory text, which makes it possible to introduce ambiguities in the requirements model. Such ambiguities may lead to misinterpretation of requirements, thus causing very expensive errors.

Another problem is that use cases are not an object-oriented notation. This makes it necessary to map the concepts appearing in the use case diagrams into elements of an object-oriented model. This activity requires an ad-hoc methodology (like the robustness analysis [10]). As a consequence the resulting object-oriented model contains elements that generally are not easily traceable back to requirements. Difficult traceability makes the management of the development process more difficult and increases the probability of errors.

Finally, use cases are, by definition, inclined to describe dynamic requisites; that is, requisites that manifest themselves in an interaction. When a requisite can be seen as a constraint, it's not so easily captured by a use case. More generally, use cases are known to be hardy suitable to describe non functional requirements.

The tutorial illustrates a technique that leads analysts to build requirements models that are both precise and object-oriented, so that they can be used directly as a basis for the subsequent design phases, also guaranteeing traceability. In practice, the conceptual and methodological base of problem frames is applied in the context of UML, thus leading to high quality requirements, suitable to effectively support the design, implementation and testing of software systems.

Rigorous approaches to requirements modelling

Rigorous approaches –not based on UML– were proposed, but they did not achieve a great popularity, mainly because they are considered not promptly applicable in UML-based development processes, and consequently hardly productive in the short term.

The proposed tutorial promotes the application of the Reference Model for Requirements and Specifications proposed in [2] and the usage of Michael Jackson's Problem Frames [1].

The Reference Model for Requirements and Specifications is employed to clearly identify and characterize the five artefact types that represent the Environment and the System (see Figure 1).

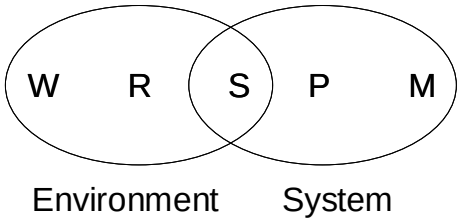

Figure 1. Five Software artefacts.

Five types of artefacts are involved in the software development process. They are:

- The *domain knowledge* (W) that provides presumed environment facts.
- The (user) *requirements* (R) that indicate what the customer needs from the system, described in terms of its effect on the environment.
- The *specifications* (S) that provide enough information for a programmer to build a system that satisfies the requirements. As graphically illustrated in Figure 1, Specifications are defined in terms of the phenomena that are shared between the Environment and the System.
- A *program* (P) that implements the specification using the programming platform.
- A *programming platform* (M) that provides the basis for programming a system that satisfies the requirements and specifications.

More precisely, within the Environment and the System it is possible to identify elements (described through "designated terms") that are controlled and others that are only visible. Figure 2 illustrates this situation. Letters e and s represent phenomena of the Environment and System, respectively. Subscripts h and v indicate phenomena that are hidden from external elements or visible by outside elements, respectively. For instance, elements e_v are controlled by the Environment and are visible by the System (and possibly affect the behaviour of the System).

User requirements are expressed from the point of view of the user, who knows well the problem domain and is interested in the effects of the system on the whole environment. Requirements are thus expressed in terms of e_h e_v s_v.

On the contrary, Specifications represent the objectives of the development. They are addressed to the designer; therefore they do not include the details of the environment, namely the parts of the environment that are not visible by the system. In conclusion, Specifications involve only e_v and s_v, i.e., the interface between the system and the environment.

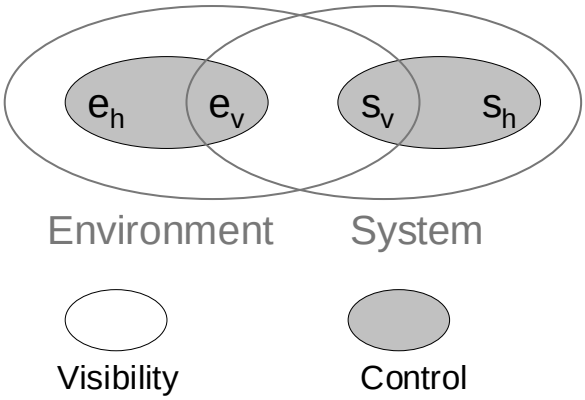

Figure 2. Visibility and control for designated terms.

The Problem Frames approach is to be used when gathering requirements and creating specifications for computer software. The core concepts are simple:

- The best way to approach requirements analysis is through a process of parallel decomposition of user requirements.
- User requirements are about relationships in the real world, not about functions that the software system must perform.

When analyzing a problem, the Problem Frames approach considers the software application to be a kind of software Machine and define the goal of a software development project in changing the real world by creating a software machine, adding it to the real world, where it will bring about the desired effects.

For instance, Figure 3 illustrates the "commanded behaviour" frame applied to the problem of controlling a gate at a railroad crossing. Such frame is employed to express those problems where the behaviour of a domain (in this case, the Gate) has to be determined according to the signals originated by another domain (the tracks in the crossing region). However, the signals cannot be treated as commands and sent directly to the commanded behaviour, rather they have to be elaborated by a machine, which implements the "intelligence" needed to guarantee that the behaviour of the controlled domain conforms to the user needs (e.g., the safe operation of the gate).

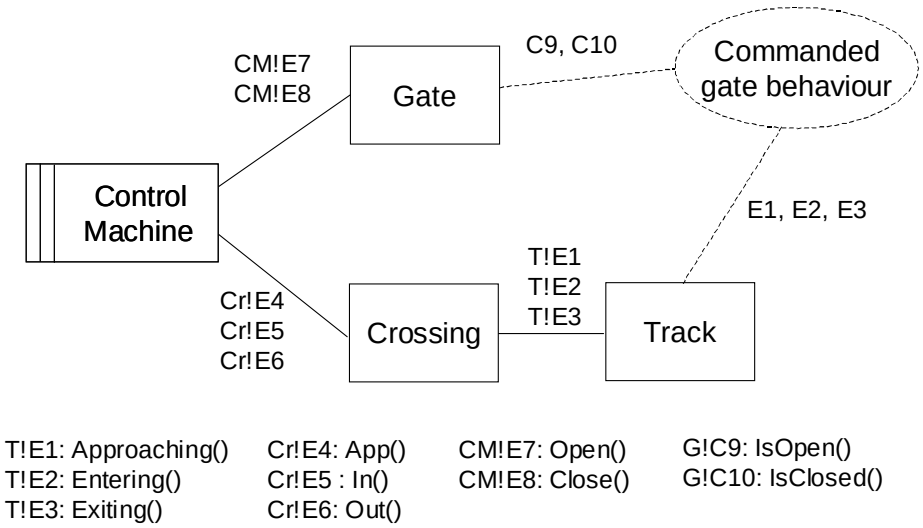

Figure 3. A commanded behaviour Problem Frame.

Every Problem Frame proposes a "concern" that identifies the descriptions that need to be made and how they fit into a correctness argument. Addressing the frame concern means assuring that domain descriptions, requirements and specifications fit together in an appropriate frame.

> The specification S describes the properties that we want the machine to have at its interface with the world. Jackson gives a general rule for mastering the problem and satisfying the requirements: "If the machine achieves the behaviour S at its interface with the world given the domain properties W the requirement R would be satisfied", i.e. S, W |- R.

In conclusion, problem frames help not only in describing requirements, but also in their fulfilment.

Objectives of the tutorial

The tutorial provides the participant with the means to apply rigorous requirements methods –thus achieving a more reliable process, and better quality products– without having to pay a high fee in terms of steep learning curves, as the familiar UML notation and concepts can be employed. In fact, the goal of this tutorial is to show how the Reference Model for Requirements and Specifications by Gunter et al. and Michael Jackson's Problem Frames can be applied *in the context of a UML process*. In particular, it is shown how familiar UML constructs can be used in a disciplined way in the construction of models that represent properly the problem domain, the user requirements and the specifications of the software system.

The tutorial presents a technique for the application of the reference model by means of UML where domains are modelled by means of UML classes and components; relations among domains are modelled by means of UML associations, aggregations, etc; shared phenomena are modelled by means of interfaces and public methods and attributes [3].

Similarly, the behavioural characteristics of problem domains and of the user requirements are described by means of state diagrams whenever an operational

specification is applicable and by means of OCL when a more declarative style is preferable [4].

The application of the proposed techniques is eases the transition from the requirements specification phase to the design phase, as no changes in the employed notation (UML) are required and –for the same reason– to improve the understanding of requirements by designers. Traceability is also improved, since elements of the requirements models are the same that appear in design models.

The final objective is to increase the quality of requirements models, while making them suitable to drive UML-based software development.

Expected audience and required background

The intended audience for this tutorial is composed of practitioners and researchers who are familiar with UML and the traditional way of developing software in a UML-based process, and are interested in:

- Learning the difference among UML models representing problem domains, requirements, specifications and design.
- Learning how to characterize the aforementioned types of models according to the Reference Model for Requirements and Specifications proposed in [2].
- Learning how to write requirements specifications according to Michael Jackson's Problem Frames [1] but using UML as the notation.
- Learning how to address non functional requirements (namely time-related properties).

All the concepts will be illustrated by means of an example derived from the industrial practice.

In conclusion, the audience will get acquainted with a most advanced technique for modelling requirements in a UML-based development process.

The adoption of the proposed approach has also some benefits on the overall development process. The tutorial can thus be interesting also for managers interested in improving UML-based processes.

Required background: the knowledge of UML is required. Some knowledge of OCL and of the Problem Frames is beneficial, although not necessary.

Context

The tutorial is based on the idea of exploiting the valuable concepts underlying the Problem Frames approach in a UML-based development process. This idea was originally proposed by prof. Lavazza at the International Conference on Software Engineering and Knowledge Engineering (SEKE) in 2003 [3].

The integration of the Problem Frames concepts into UML was then refined by addressing increasingly complex issues and employing the latest release of UML [4][5]. Some initial work addressing the integration of scenario-based requirements description

in the context of Problem Frames was performed [6]. An extension of OCL to deal with time was also defined [13].

The tutorial concerns techniques that are currently subject of a growing interest. Actually, there is a large movement that is addressing the usage of UML (beyond use cases) in the representation of requirements. Other authors propose to merge the usage of UML and problem frames [7], or exploring the application of (suitable extensions of) OCL to the specification of real-time requirements [8][9].

In conclusion, the proposers believe that the amount of work done to define the integration of Problem Frames concepts in the UML development process is complete and mature enough to support a methodology that can be employed in industrial settings.

Table of contents and time allocation

The tutorial is three hours long (not including breaks). It is organized into the following sections.

1. The problem of requirements modelling (40')

This section addresses the problem of establishing the precise meaning of requirements modelling. The concepts of Environment and Problem Domain are introduced. The relationship between the Environment and the machine (which is seen as a black box in this phase) is described in terms of "phenomena" that are shared between the Problem Domain and the Machine. Shared phenomena constitute the "interface" between the machine and the environment. The problem domain is defined as the portion of the Environment that is visible by the machine.

The requirements are defined as properties expressed over the Environment and the Machine, while specifications are defined as properties expressed over the interface only.

An example application is used to illustrate the concepts. This example will be used throughout the tutorial in order to illustrate the presented concepts.

2. Limits of the use cases (20')

The requirements for the example application are expressed by means of use cases. The limits of the use cases are observed.

3. Problem frames (30')

Problem frames are briefly introduced. An example for each type of frame is given. The frame concern is illustrated. Why 'system models' trying to combine the specification, domain properties and requirements are unsuitable to support complex developments, and can actually lead to bad results.

4. Problem frames with UML (1 h)

This section shows how to apply UML to model the Environment, problem domain, and shared phenomena. Requirements (i.e., the desirable properties of the whole system) are expressed by means of state diagrams and OCL.

Correctness arguments are also addressed.

In order to illustrate the technique, a set of problems frames are presented and modeled by means of UML. The whole example application is then modelled.

5. Dealing with time: extending OCL (15')

UML is limited with respect to the possibility of specifying temporal aspects. By means of OCL it is not possible to reference different time instants in a single OCL formula. Only invariant properties can be formalized, which at most include references to attribute values before or after method execution.

An extension of OCL is presented to overcome these limitations. Examples are given.

6. Conclusions and possible evolutions of the proposed method (15')

The final discussion will involve a brief evaluation of pros and cons, applicability conditions, tool support and evolution, taking into consideration new modelling language proposals (e.g., SysML).

Teaching methods

The tutorial consists mainly of presentations, which will address both the description of the methodology and its application to a real problem.

The subproblems that will emerge during the process of requirements modelling will be discussed with the participants, and their opinion and contribution will be solicited.

Requested Audio/Video equipment

The presenter will use his own portable PC. A projector is required.

References

[1] M. Jackson, *Problem Frames - analysing and structuring software development problems*, Addison-Wesley ACM Press, 2001.

[2] C.A. Gunter and E.L. Gunter and M. Jackson and P. Zave. A Reference Model for Requirements and Specifications. *Software*, 17(3), May-June 2000.

[3] L. Lavazza, "Rigorous Description of Software Requirements with UML", *15th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE2003)*, San Francisco, July 2003.

[4] L. Lavazza and V. Del Bianco "A UML-based Approach for Representing Problem Frames", *1st International Workshop on Advances and Applications of Problem Frames (IWAAPF)*, an ICSE'04 Workshop, Edinburgh, 24 May 2004.

[5] L. Lavazza and V. Del Bianco, "Combining problem frames and UML in the description of software requirements". *Fundamental Approaches to Software Engineering (FASE06)*, part of the Joint European Conferences on Theory And Practice of Software ETAPS 2006, 25 March-2 April 2006, Vienna, Austria.

[6] V. del Bianco and L. Lavazza, "Enhancing Problem Frames with Scenarios and Histories: a Preliminary Study", *2nd International Workshop on Advances and Applications of Problem Frames (IWAAPF)*, an ICSE'06 Workshop, 23 Maggio 2006, Shanghai.

[7] C. Choppy and G. Reggio "A UML-based Method for the Commanded Behaviour Frame, *1st International Workshop on Advances and Applications of Problem Frames (IWAAPF)*, an ICSE'04 Workshop, Edinburgh, 24 May 2004.

[8] S. Flake, W. Müller "An OCL Extension for Real-Time Constraints". In T. Clark and J Warmer (Eds.), *Advances in Object Modelling with OCL*, Springer Verlag, October 2001

[9] P. Ziemann, and M. Gogolla, "An Extension of OCL with Temporal Logic". Critical Systems Development with UML - *Proceedings of the UML'02 workshop*, pages 53-62. TUM, Institut fur Informatik, Septeber 2002, TUM-I0208.

[10] Doug Rosenberg, Kendall Scott, “Use Case Driven Object Modeling with UML” Addison-Wesley Object Technology Series, 1999.
[11] M. Glinz, S. Berner, S. Joos, J. Reyser, N. Schett, R. Schmid, Y. Xia, “The ADORA Approach to Object-Oriented Modelling of Software”, Proc. of CAiSE'01, Interlaken, June 2001.
[12] R.R. Hurlbut, “A Survey of Approaches For Describing and Formalizing Use Cases”, http:\\www.iit.edu\~rhurlbut\xpt-tr-97-03.html
[13] L. Lavazza, S. Morasca, A. Morzenti “A Dual Language Approach to the Development of Time-Critical Systems with UML” TACoS (International Workshop on Test and Analysis of Component Based Systems) in conjunction with ETAPS 2004, Barcelona, March 27 - 28, 2004.

Biographies of speakers

Luigi Lavazza is associate professor at the Department of Computer Science and Communications at the University of Insubria at Varese, Italy. Formerly he was assistant professor at the Department of Electronics and Informatics of Politecnico di Milano, Italy. Since 1990 he is also a member of the Software Engineering Unit at CEFRIEL. Here he led several research projects as well as several consultancy activities and technology transfer initiatives.

His research interests include: Model based development, especially concerning real-time and embedded software, Software process modelling, measurement and improvement, Requirements engineering and Configuration management.

He has been member of the Program Committee of the International Conference on Engineering Complex Computer Systems (ICECCS) in 1997 and 1998, of the Software Engineering and Applications Conference (SEA) in 2005, and of the International Workshop on Advances and Applications of Problem Frames in 2006.

He is co-author of over 70 scientific articles, published on international journals, or on the proceedings of international conferences or on books. He was involved in several international research projects. He also served as reviewer of ESPRIT and IST projects, and of international journals and conferences such as IEEE Transactions on Software Engineering, Empirical Software Engineering Journal (Springer), Software and System Modeling (Springer), ICSE (International Conference on Software Engineering), ESEC (European Software Engineering Conference), FSE (Foundations of Software Engineering).

Luigi Lavazza has taught object oriented modelling techniques since 1993 in continuous education programs of Politecnico di Milano, CEFRIEL and private organizations.

He is coauthor –with L. Baresi and M. Pianciamore– of the book “Dall’idea al codice con UML 2.0 – Guida all’utilizzo di UML attraverso esempi”, (*From the idea to code with UML 2.0 – A guide to using UML by means of examples*), Pearson 2006 (in Italian).

Vieri del Bianco received the Laurea in Electronics Engineering at the University of Firenze in 1999. The final dissertation concerned the development of a software process suited for research and pilot projects in an academic setting; the process was applied for the refactoring of an existing system, the system objective was the model checking of TCTL properties on systems modelled with TPN (Time Petri Nets). In 2000 he received the Master degree in Information Technology, with specialization in Software Engineering, from CEFRIEL. During the master course he worked in a project concerning the evaluation of application servers through software metrics and usage scenarios. Since 2000 he works as a researcher at the Software Engineering unit at CEFRIEL. In 2005 he obtained his Ph.D. degree from Politecnico di Milano; the main thesis concerned the development of a UML based formal notation, UML+, to support the specification of Real-Time systems and to enable the transformation of UML+ models to other formal languages (TRIO and Timed Automata). Currently he is a

researcher at University of Insubria, where he is working at the Qualipso project, an Integrated Project funded by the EU and devoted to define and implement technologies, procedures and policies to leverage the Open Source Software development current practices to sound and well recognised and established industrial operations.

He was involved in a few research projects, including the ITEA project DESS (Development process for real-time Embedded Software Systems), and the MIUR project DICE (Distributed Infrastructure for Cultural hEritage).

In DESS he worked at the integration of UML and formal methods for the rigorous and verifiable specification of real-time systems. In DICE he worked at the definition of a peer-to-peer architecture of a platform supporting information sharing among stakeholders of cultural heritage information.

His current research interests are embedded and real-time systems, formal systems and notations, software processes, agile methodologies, aspect-oriented programming, configuration management, software test, software metrics and software reuse.

Relevant publications of the proposers

Publications concerning the usage of UML in the representation of requirements

"Rigorous Description of Software Requirements with UML", 15th International Conference on Software Engineering and Knowledge Engineering (SEKE2003), San Francisco, July 2003 (Luigi Lavazza)

"A UML-based Approach for Representing Problem Frames", 1st International Workshop on Advances and Applications of Problem Frames (IWAAPF), an ICSE'04 Workshop, 24 May 2004, Edinburgh (Luigi Lavazza, Vieri Del Bianco)

"Combining problem frames and UML in the description of software requirements". Fundamental Approaches to Software Engineering (FASE06), part of the Joint European Conferences on Theory And Practice of Software ETAPS 2006, 25 March-2 April 2006, Vienna, Austria. (Luigi Lavazza, Vieri Del Bianco)

Publications concerning object-orientation and UML

"Updating the Schema of an Object-Oriented Data Base", IEEE Data Engineering Bullettin1, vol.14 n.2, June 1991. (A. Coen-Porisini, L. Lavazza, R. Zicari)

"A Comment on Considering Class Harmful", Communications of the ACM, vol.36 n.1, technical correspondence. (L. Lavazza)

"Assuring Type-Safety of Object Oriented Languages", Journal of Object-Oriented Programming, February 1994 (A. Coen-Porisini, L. Lavazza, R. Zicari)

"Experiences in the Implementation of a Process-centered Software Engineering Environment using Object-Oriented Technology", Theory and Practice of Object Systems, vol. 1, n. 2, 1995 (S. Bandinelli, L. Baresi, A. Fuggetta, L. Lavazza)

"Combining UML and formal notations for modelling real-time systems", Joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE), Vienna, 10-14, September 2001. (Luigi Lavazza, Gabriele Quaroni, Matteo Venturelli)

"Deriving executable process descriptions from UML", ICSE 2002, International Conference on Software Engineering, Orlando, Florida, 19-25 May 2002. (E. Di Nitto, L. Lavazza, M. Schiavoni, E. Tracanella, M. Trombetta)

"A Formalization of UML Statecharts for Real-Time Software Modeling", The Sixth Biennial World Conference on Integrated Design Process Technology (IDPT 2002), Pasadena, California, 23-28 June 2002. (Vieri Del Bianco, Luigi Lavazza, Marco Mauri)

"Model Checking UML Specifications of Real-Time Software", The Eighth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2002), Greenbelt, Maryland, 2 - 4 December, 2002. (Vieri Del Bianco, Luigi Lavazza, Marco Mauri)

"Towards UML-based formal specifications of component-based real-time software", Fundamental Approaches to Software Engineering (FASE03), ETAPS European Joint Conferences on Theory And Practice of Software, 5-13 April, 2003, Warsaw, Poland. (Vieri Del Bianco, Luigi Lavazza, Marco Mauri)

"A Dual Language Approach Extension to UML for the Development of Time-Critical Component-Based Systems", TACoS (International Workshop on Test and Analysis of Component Based Systems) Warsaw, 13 April 2003. (Luigi Lavazza, Sandro Morasca, Angelo Morzenti)

"Towards a component-based, model-driven process supporting variability of real-time software" Workshop on Software Variability Management, a satellite event of the International Conference on Software Engineering 2003, May 3, 2003, Portland, Oregon, USA. (Vieri Del Bianco, Luigi Lavazza)

"Simulation-based Verification of UML models", 15th International Conference on Software Engineering and Knowledge Engineering (SEKE2003), San Francisco, July 2003 (Luigi Lavazza e Giuseppe Occorso)

"A Dual Language Approach to the Development of Time-Critical Systems with UML" TACoS (International Workshop on Test and Analysis of Component Based Systems) in conjunction with ETAPS 2004, Barcelona, march 27 - 28, 2004 (Luigi Lavazza, Sandro Morasca, Angelo Morzenti)

Publications concerning Requirements engineering

"A Conceptual Basis for Feature Engineering", Journal of Systems and Software, 49 (1) (15 December 1999) pp. 3-15 (C. Reid Turner, Alfonso Fuggetta, Luigi Lavazza, Alexander L. Wolf)

"Requirements-based Estimation of Change Costs", Empirical Software Engineering - An International Journal, vol. 5, n. 3, Novembre 2000. (Luigi Lavazza e Giuseppe Valetto)

"Enhancing Requirements and Change Management through Process Modelling and Measurement", ICRE2000 Fourth IEEE International Conference On Requirements Engineering, June 19-23, 2000, Schaumburg, Illinois (Luigi Lavazza e Giuseppe Valetto)

"Enhancing Problem Frames with Scenarios and Histories: a Preliminary Study", 2nd International Workshop on Advances and Applications of Problem Frames (IWAAPF), an ICSE'06 Workshop, 23 May 2006, Shanghai. (Vieri Del Bianco, Luigi Lavazza)

"Enhancing Problem Frames with Scenarios and Histories in UML-based software development" to appear on Expert Systems – The Journal of Knowledge Engineering, Blackwell Publishing. (Vieri Del Bianco, Luigi Lavazza)