Information Systems and Web Information Systems: A Methodological Perspective

Colette Rolland

INFORMATION SYSTEMS AND WEB INFORMATION SYSTEMS : A METHODOLOGICAL PERSPECTIVE

Colette Rolland
Université Paris1 Panthéon Sorbonne
CRI, 90 rue de Tolbiac
75013 Paris, France
rolland@univ-paris1.fr

ABSTRACT

The Development of Information Systems (ISD) is a complex activity that requires methodological support. Current ISD methods are predominantly following an object driven paradigm that allows us to capture the static as well as the dynamic dimensions of real world objects into information objects. However, Web Information Systems have facets such as information presentation, user profile, navigation structure etc. which pose new challenges to ISD methods. The keynote talk will provide an overview of current ISD methods and discuss the new challenges raised by the development of Web Information Systems. This paper is a brief overview of the key points developed in the talk.

INFORMATION SYSTEM DEVELOPMENT

In this section we briefly introduce the two-phase organisation of the process for developing information systems namely, Requirements Engineering as a prerequisite for System Engineering. We then, use the [Olle88] framework for classifying IS conceptual models that traditionally support the requirements engineering phase and comment their characteristics. We discuss the limitations of conceptual models as supports for ensuring the match of system functions to organisational requirements and then, introduce more recent requirements engineering models. Finally we comment the evolution of process models to support the modelling process.

ISD organisation

Traditionally Information System Development (ISD) has made the assumption that an information system captures some excerpt of world history and hence has concentrated on modelling information about the Universe of Discourse [Olle88]. Thus the IS development can be organised in two phases as shown in Figure 1. The first phase where conceptual modelling is carried out aims at abstracting the specification of the required information system i.e. the conceptual schema, from an analysis of the relevant aspects of the Universe of Discourse about which the user community needs information [Dubois89]. The succeeding phase, that of system engineering, uses the conceptual schema to design and implement a working system which is verified against the conceptual schema.

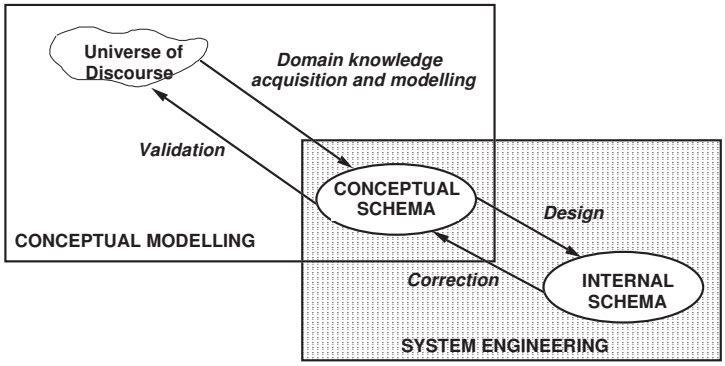

Figure 1. Two-phase organisation of system life-cycle

Classification framework of conceptual models

The information systems community has developed a large number of conceptual models for representing conceptual schemata. This variety has arisen because of the need to capture as many aspects of real world semantics as possible. Given this plethora of models, it has been found necessary to develop frameworks for classifying and understanding these. One framework which classifies models based on the perspective adopted to view the Universe of Discourse was developed by [Olle88]. It organises models into the classes of *process-oriented*, *data-oriented*, and *behaviour-oriented* models. In Figure 2, this framework has been shown as defining a three-dimensional space within which conceptual models can be positioned.

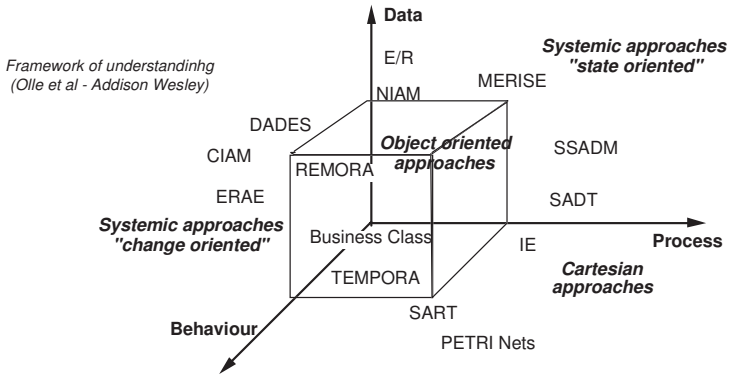

Figure 2 : The three dimensional framework for classifying conceptual models

The three dimensional framework highlights the fact that information systems can be looked upon in three different ways. When seen as *process-oriented*, an information system is a function in an organisation which returns some information. When seen as *data-oriented*, information systems are viewed as mirroring the information contents of organisations and it is expected that the information system would be a supplier of this information. Finally, in the *behavioural perspective*, an information system is an artefact which handles interesting events that occur in the organisation by performing one or more functions. These functions modify the information contents of the information system which are again available for manipulation through events.

These different views naturally lend themselves to specific kinds of treatment. Thus, when the information system is viewed as a function in the Universe of Discourse, then during analysis, the components of this function are discovered. This is because the function may be very complex and needs to be broken down into its functional elements to understand it better. If any of the functional components are themselves complex then, they are

decomposed recursively till simple, well understood functions are reached. Clearly, this results in a hierarchy of functions rooted in the original function. Whereas this hierarchy identifies the functional components of the information system function it does not establish an inter-relationship between these components, i.e., which function receives data from which function and sends data to which one is not articulated. This is done by using conceptual models for building data flow diagrams.

It can be seen that the process-oriented perspective views information systems as processors of information. In contrast, the data oriented approach looks at an information system as mirroring the information contents of the real world, as a storehouse of information. Since information is to be kept about real life things, an identification of all these relevant 'things' coupled with their abstraction as information carrying entities is carried out. The abstracted entities and their inter-relationships are then represented as a conceptual schema. As the mirrored world changes, so the information system must reflect these changes. Therefore the information system is seen as a data manager, maintaining and delivering information at all times.

Finally, in the behavioural perspective, the attempt is to identify the interesting events that occur in the real world, the information affected by their occurrence and the functions that cause this effect to be felt. For this, three things are done (a) Real events are abstracted into information bearing events, (b) Real world things are abstracted (as in the data perspective) into information bearing entities and relationships, and (c) Functions to be invoked to carry out the effect of the event are identified and associated with it. It can be seen that the behavioural view promotes a transaction management view of an information system.

Over the years, the usefulness of having three completely different perspectives with little integration in them has come to be questioned. Two distinct trends towards integration emerged. The first was the development of object-oriented conceptual models, the majority of which integrated together the process and data-oriented perspectives, though some conceptual models that also integrated the behavioural one were developed [Brunet90], [Desfray94], [Martin92]. The second was a trend towards 'loosely connected' conceptual models which consisted of a set of conceptual models, each according to a different perspective. Therefore, the Universe of Discourse was conceptualised as individual but connected conceptual schemata. This inter-connection was seen in the Yourdon approach [Yourdon89] in the mid-eighties which loosely connected the data flow, ER modelling and state transition diagram techniques. It was also seen later in OMT [Rumbaugh91] which integrated an object-oriented model with data flow diagrams and event modelling. This movement led to the unified Modelling language, UML as a denotational standard for expressing conceptual schemata as well as design and implementation schemata.

Requirements Engineering models

In the traditional view of engineering information systems through conceptual modelling, the focus is on producing a formal specification of the system to be developed. This specification concentrate on what the system should do, that is on its functionality. Such a specification acts as a prescription for system construction. It seems that this leads to a number of failures in system development. Indeed, a number of studies show [Lubars93; McGraw97; Standish95] that systems fail due to an inadequate or insufficient understanding of the requirements they seek to address. Further, the amount of effort needed to fix these systems has been found to be very high [Johnson95]. To correct this situation, it is necessary to address the issue of requirements elicitation, expression and validation in a more focussed manner. The expectation is that as a result of this, more acceptable systems will be developed in the future.

In tackling the above problem, the area of requirements engineering tries to go beyond the functionality based view of conceptual modelling. Requirements engineering extends the ' what is done by the system' approach with the ' why is the system like this' view. This why

question is answered in terms of organisational objectives and their impact on information systems supporting the organisation. In other words, information systems are seen as fulfilling a certain purpose in an organisation and requirements engineering helps in the conceptualisation of these purposeful systems. This has two implications (a) elicitation and validation of the requirements of a system is done with respect to their purpose in organisations and (b) only organisationally purposeful systems are conceptualised.

To deliver the foregoing, requirements engineering models have being developed that can be classified into :

- *goal-driven approaches* and,
- *scenario-based approaches.*

Goal driven approaches model organisational objectives so as to relate them to the functions of the system. In this sense, they aim at the conceptualisation of purposeful systems only. They contribute to the interpretation of requirements before they are understood and before they are transformed into system function specifications. Thus they support conceptualising purposeful systems. The broader view of a requirements specification that they advocate is going beyond the classical conceptual schema describing the system functionality. It includes enterprise modelling which represents the *Why* part of system requirements.

Enterprise modelling as developed in the F3 project [Bubenko94] is an example of such a goal-driven approach to requirements engineering. Enterprise modelling was further refined in the EKD method to support change management [Loucopoulos98; Rolland97; Kardasis98; Rolland98]. In the KAOS approach [Dardenne93], the emphasis is on supporting formal refinement of high level goals into system constraints. Although generic models are advocated, goal modelling and refinement have supplied simple guidance via heuristics[van Lamsweerde95]. The I* approach [Yu94a, b, c] creates models of the environment of the system that emphasise agents and their relationships. Their strategic dependency and rationale models allow tracing of dependencies between agents, goals and tasks and support reasoning to identify trade-offs between functional requirements and non functional requirements [Mylopoulos92].

Independently of goal modelling, an alternative approach to RE, the scenario-based approach [Jacobson95] has been developed. *Scenario based approaches*, by focussing on the users' view points, help in modelling purposeful system usage from which useful system functions can be derived. Scenarios provide dynamic meaning to goals whereas goals provide the intentional setting within which scenarios find meaning. By capturing examples, scenes, narrative descriptions of contexts, use cases and illustrations of agent behaviours, scenarios have proved useful in requirements elicitation in a number of ways : to elicit requirements in envisioned situations [Potts97], to help in the discovery of exceptional cases, to derive conceptual object-oriented models, to understand needs through scenario prototyping and animation, to reason about design decisions, to create context for design [Kyng95] and so on. The underlying reason for the popularity of scenario-based approaches seems to be that people react to descriptions of real happenings and real things. This reaction helps in clarifying requirements expected of systems. Thus, the scenario school argues, that typical scenarios are easier to get in the first place than goals. Goals can be made explicit only after deeper understanding of the system has been gained.

Scenarios have been developed for different purposes with different contents, expressed in different levels of abstraction and with different notations. In [Rolland98b] the reader will find a framework to classifying and comparing scenario-based approaches and an analysis of 12 significant scenario-based approaches.

It might also be noticed that in order to overcome some of the deficiencies and limitations of goal-driven and scenario-based approaches used in isolation, some proposals

have been made recently to couple goals and scenarios together. Goals have been considered as contextual properties of use cases and as a means to structure use cases. The goal scenario combination has been used to operationalise goals, to check whether or not the current system usage captured through multimedia scenarios fulfils its expected goals, to infer goals specifications from operational scenarios and to discover new goals through scenario analysis.

As an example of an approach which combines goal modelling and scenario authoring consider the CREWS-L' Ecritoire approach [Rolland97; Rolland98a] developed within the CREWS ESPRIT project. CREWS-L' Ecritoire uses a bidirectional coupling allowing movement from goals to scenarios and vice versa. The complete solution is in two parts : when a goal is discovered, a scenario can be authored for it and once a scenario has been authored, it is analysed to yield goals. By exploiting the goal-scenario relationship in the reverse direction, i.e. from scenario to goals, the approach proactively guides the requirements elicitation process. In this process, goal discovery and scenario authoring are complementary steps and goals are incrementally discovered by repeating the goal-discovery, scenario-authoring cycle.

ISD process models

This brief overview of ISD methods would not be completed without considering the second dimension of any method : *the process dimension*. This is dealt with in the following.

The conceptual modelling community emphasised the product aspects of systems at the expense of the process employed to deliver the product. Thus, the structure of the conceptual schema, its completeness, and consistency etc. was more important than how it was developed. Early process models were activity based. They looked upon the process as consisting of a set of activities which could be decomposed into simpler ones and which were linearly ordered. Every successive activity was to be performed after the completion of the previous one. Such process models are known to be restrictive [Wynekoop93] because they assume

(a) that it is possible to pre-define the development path that can be taken through the activities of a process model. Thus, they restrict the creativity of the developer in choosing a path specific to a given situation.
(b) that each conceptual schema is built afresh and therefore there is no need to keep track of the processes that built them.
(c) the 'upon completion' rule which prohibits movement to an activity later in the order or backtracking to one earlier in the order.
(d) that the relationship between an activity and the product built by it was not interesting.

Later, a number of other more flexible process models were built. Yet, by and large, conceptual modelling continued to follow the activity based approach to process models i.e. the Waterfall model [Royce70].

In contrast to conceptual modelling that largely ignored the development process, requirements engineering has explicitly considered the issue of the process support to be provided. Two important issues arise :

1. How can attention be channelled to deal with the real productive tasks of requirements engineering? In other words, it is necessary to guide the requirements engineering process to concentrate on discovering goals, scenarios etc.
2. How can one learn from past practice? That is, if some decisions were taken in a given situation in the past then how can one benefit from experience with that? Thus it is necessary to keep a trace of past decisions.

These two aspects of the requirements engineering process, namely *guidance* and *tracing* led to guidance and trace models embedded in computer assisted tools to support the RE process in a semi-automated manner.

Guidance

Some experience in guidance exists in software engineering where guidance was classified as *active* or *passive* [Dowson94]. The former was focussed on ensuring that the development process employed was an instance of the process model and consequently, guidance was directed towards process model enforcement. The latter was concerned with an identification of what could be done next in the development process. In [Feiler93] passive guidance has been defined as the generation and subsequent presentation of the set of legal steps that were available at any moment in the development process. One out of these could then be selected as the task to be done next.

The software engineering view is that *active* guidance should be provided. Thus, guidance cannot be provided without an adequate *process model*. Existing process models do not seem adequate to requirements engineering as they prescribe a predefined plan of actions. *Activity-oriented process models* [Royce70] come from an analogy with problem-solving and provide a frame for manual management of projects. This linear view is inadequate for methods which support backtracking, reuse of previous designs, and parallel engineering. *Product-oriented process models* [Humphrey89; Finkelstein90; Franckson91] represent the development process through the evolution of the product. They permit design tracing in terms of the transformations performed and the resulting products. Finally, *decision-oriented models* integrate more deeply the semantics attached to evolutionary aspects. The notion of design decision facilitates understanding of the designer' s intention, and better reuse of results [Potts89] but the flexibility and situatedness of requirements engineering processes is not adequately handled in existing decision-oriented models.

The importance of *situatedness* in process modelling is also acknowledged by the software engineering community where it was found that departures from the process model occurred in actual practice. A concerted effort was put in to allow process models to respond to these departures. One approach was to assume prescriptive models and then, modify them to accommodate real processes. This modification could be achieved in two ways. First the extent of deviations from the prescription that could be allowed was modelled as constraints. Any actual deviation that satisfied the constraints was therefore manageable and the process enactment mechanism could handle it. This way of handling deviations took the prescriptive approach to its logical conclusion: it prescribed the deviations allowed in a prescription. The second way of handling deviations was to allow changes to be made in the prescription as and when they are needed. Thus, a level of dynamicity is superimposed on the basic prescription.

In contrast to this, the requirements engineering community recognised that the core of their task was the generation and exploration of alternatives from which the right one is selected for the situation at hand. This can be seen in the IBIS process model [Potts89] where a number of alternatives for resolving an issue were generated. This process model is at a very high level of abstraction and had to be buttoned down to real methods and tools. The contextual model [Rolland91; Rolland95; Pohl96] attempted to do this. A context was defined here as the application of an intention to a given requirements engineering situation. It organised requirements engineering methods as a set of contexts of three kinds, *executable*, *plan*, and *choice* contexts respectively. A Choice context groups together all possible alternative ways of meeting its intention. These alternatives were themselves contexts thus leading to a hierarchy of alternatives. A plan context is a collection of simpler contexts such that their execution, in the various possible orders prescribed in the plan context meets its intention. Finally, an executable context is one which can be directly executed to meet its intention (and is atomic in this sense). It can be seen that the contextual model attempted to reconcile process prescription with alternative generation, the former through plan contexts and the latter through choice contexts. Another attempt to root the notion of alternatives in methods was made in the decisional approach [Prakash97].

Tracing

In the requirements engineering community there is no longer the question whether traceability is a useful thing or not. Capturing and maintaining traces is seen as an essential activity to be performed during requirements engineering and standards such as [DoD-2167A; IEE-830] mandate that requirements traceability be practiced. A comprehensive overview of possible usage of trace information and the expected benefits can be found in [Gotel94], [Ramesh93a] and [Pohl96]. These reports indicate that requirement traceability is a vital component in implementing a quality system, essential for consistent change integration, leads to less errors during system development, plays an important role in contract situations, and improves system acceptance.

Process traceability can be divided into three parts [Pohl96] :

* Process execution traceability, i.e. the recording of data that enables the reassembly of the sequence of steps of a process.
* Product evolution traceability, i.e. the recording of data that enables you to see how the product has evolved during the process.
* Traceability of the relationships between process execution and product evolution.

The pivotal goal of process traceability is to enable tracing of the requirements produced during the RE process. On one hand, traceability from the requirements specification through design to implementation and vice-versa is needed to understand the rationale of the implemented system. On the other hand, the process leading to the requirements specification must be traceable to understand the rationale for the requirements themselves. The former is referred to as post- traceability whereas the latter is called pre-traceability [IEEE-830].

Product traceability is available in some methods like Class/Relation, OOSE and rAdar. Post-traceability is supported by some commercial tools like RT from Teledyne Brown Engineering, RMT from Marconi Systems Technology , and RDD100 from Ascent Logic. Pre-traceability has been investigated more recently [Gotel94; Pohl96; Ramesh93b; Ramesh95].

An interesting framework for requirements pre-traceability was provided by Pohl [Pohl94] who described the requirements engineering process in a three dimensional space assuming that there are three major facets of the RE process, namely modelling the requirements in a more complete manner, modelling with more formality, and more consensus among stakeholders. The trace of the requirements engineering process is modelled as a path within the three dimensional space starting from an initial incomplete, informal specification representative of individual viewpoints and ending with the desired output which is a complete, fully agreed and formally described specification of the intended system. Capturing the RE process trace and thereby establishing requirements pre-traceability means recording information along each of the three dimensions, on the relationships between the three kinds of information and relating those to actual process performance.

WEB INFORMATION SYSTEM DEVELOPMENT

Because of the rapid development of the Web technology and of the increasing interest of users and developers, the notion of a Web site is moving from a set of HTML pages *to Web-based Information System* (WIS). A *WIS* is an Information System providing facilities to its to access complex data and interactive services through the Web. E-business applications, Intranet systems, CRM and supply chain applications are examples of WIS.

Despite this rapid evolution (or because of this), WIS development is essentially '*ad hoc'*. Developers often consider Web development as a media manipulation and presentation creation rather than traditional IS development. Thus, WIS development does not follow the well established engineering principles and consequently, it is difficult to ensure the quality of the resulting product.

Another important factor is that the Web users' community is very heterogeneous. Many of the so-called 'Internauts' navigating on a site might have different goals and different backgrounds and knowledge. This leads to usability problems such as "loss in Hyperspace" and "cognitive overload" which have been reported in the literature.

WIS development is more complex than IS development and raises many new issues such as presentation issues, user profiling, dynamic adaptation of the format and informational content presented to a user, navigation support etc. Over all, there is clearly a need for developing WIS development method that can provide a disciplined way-of-working to ensure quality WIS development. There is some concerns in the literature about the problems that can occur if WIS development remains 'ad-hoc' [Zelnick98], [Gibbs94].

As an example of such effort to define a WIS development method, let us briefly introduce the AWIS-Method (AWIS-M) [Gnaho99]. AWIS-M support the analysis and design of Adaptive Web-based Information Systems (AWIS). The key aim of the AWIS method is to adapt the information provided and the services offered by a WIS to the individual needs and preferences of its users.

As shown in Figure 3, the AWIS-M considers the creation of an Adaptive Web-based Information System according to four perspectives :

- the management of the informational contents,
- the definition of the navigational structure,
- the definition of the user interface and,
- the identification and description of potential users and of their goals.

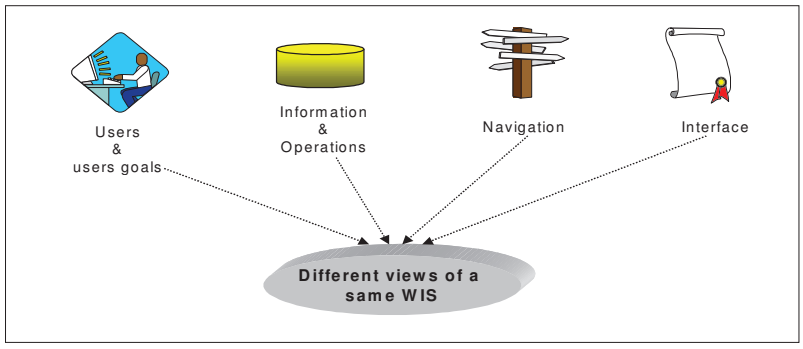

Figure 3. The four perspectives of Web-based systems development

Each of these views is supported by appropriate *models* which allows the WIS engineer to design the system considering each view in isolation from the others.

- Models that support the *Users & users' Goals* view capture (a) information about the potential user such as their background, knowledge, preferences etc. in order to define user categories and (b) usage goals of these potential users.
- Models related to the *Information view* are used to model the WIS information contents, i.e. the domain knowledge that the WIS will store and return to its users.
- Models that support the *Navigation* view deals with the presentation structure of the WIS informational contents. They help structuring the hyperspace in a net of nodes and links among them.

- Finally, models associated to the *Interface* view deal with the formatting of pages associated to the navigation structure.

Obviously, traditional IS development is mainly concerned with the second view and to a certain extend, with the fourth one. Besides, whereas the three last views are mentioned in a number of WIS development research projects [Fernandez98], the first one is an original contribution of the AWIS-Method. By capturing the user profiles, the method is able to define user categories and to tune the presentation of the WIS contents according to the specificity of the user profile. Besides, capturing the user goals, the method is able to define guidelines for navigating in the hyperspace to optimise the satisfaction of the user goal.

It shall be noticed that identifying the four perspectives help mastering the complexity of a WIS development. There is some independence between the view that allows the WIS engineer to model a view in isolation from the others. In addition, the WIS engineer can define several navigational structures for a single informational contents, or present the same navigational structure in different ways. Therefore, the goal is not to create a single Web Information System, but a set of views of the same system, each of them being aligned to the individual needs and interests of a WIS user.

CONCLUSION

The paper is an overview of the keynote talk that will be delivered during the Conference. It focuses on a methodological perspective to both IS development and Web IS development. The key part of this overview is a state of the art on ISD methods, considering both their product aspect and their process aspects. The former deals with conceptual and requirements engineering models whereas the latter is concerned with process models to guide and trace the system development process. The paper introduces the new challenges posed by Web ISD and discusses some of the key issues.

REFERENCES

Brunet J., Cauvet C., Lasoudris L(1990), Why using Events in a High-level Specification, in Proceedings of the International Conference on Entity-Relationship Modelling, ER90, Lausanne, Switzerland.

Bubenko J., Rolland C., Loucopoulos P., De Antonnellis V. (1994) Facilitating Fuzzy to Formal Requirements Modelling, Proc. Int. Conf. on Requirements Engineering (ICRE), Colorado Springs, US.

Dardenne, A., A. v. Laamsweerde and . Fickas S. (1993) Goal Directed Requirements Acquisition, Science of Computer Programming, 20 (1-2), pp3-50.

Desfray, P. (1994) Object Engineering, the Fourth Dimension, Addison-Wesley/Masson,.

DoD-2667A Military Standard (1988) Defence System Software Development, Department of Defence.

Dowson M., Fernstrom C. (1994) Towards requirements for Enactment Mechanisms, Proc. European Workshop on Software Process Technology.

Dubois E., Hagelstein J., Rifaut A. (1989) Formal Requirements Engineering with ERAE, Philips Journal of Research, Vol 43, No 4.

Feiler P.H. and Humphrey W.S. (1993) Software Process Development and Enactment: Concepts and Definitions, Proceedings of the. Second Intl. Conf. On Software Process.

Fernandez M.F., Florescu D., Levy A.Y., and Suciu D. (1998) Catching the boat with Strudel: Experiences with a web site management system, in Proc. Sigmod'98, pages 414-425, Seattle.

Finkelstein A., Kramer J., Goedicke M. (1990) ViewPoint Oriented Software Development, Proc. Conf Le Génie Logiciel et ses Applications, Toulouse, p 337-351.

Franckson M., Peugeot C. (1991) Specification of the Object and Process Modeling Language ESF Report n° D122-OPML-1. 0.

Gibbs W. (1994) Software's chronic crisis, Scientific American.

Gnaho C. , Larcher F. (1999) A Goal Driven Approach to Web Navigation Modeling, Proc. International Workshop on Internet Data Management, Florence, Italy.

Gotel O., Finkelstein A. (1994) Modelling the Contribution Structure Underlying Requirements, in Proc. First Int. Workshop on Requirements Engineering : Foundation of Software Quality, Utrech, Netherlands.

Humphrey W.S., Kellner M.I.(1989) Software Process Modeling: Principles of Entity Process Models, Proc. 11th Int. Conf. on Software Engineering.

IEEE-830 (1984) Guide to Software Requirements Specification, ANSI/IEEE Std 830.

Jacobson I. (1995) The use case construct in object-oriented software Engineering. In Scenario-based design: envisioning work and technology in system development, John M. Carroll (ed.), John Wiley, 309-336.

Johnson J. (1995) Chaos : the Dollar Drain of IT project Failures. Application Development Trends, pp.41-47.

Kardasis P., Loucopoulos P. (1998) Aligning Legacy Information Systems to Business Processes. Proceedings of the 10th Conference on Advanced Information Systems Engineering, CAiSE98. Pisa, Italy.

Loucopoulos, P., Kavakli, V., Prekas, N., Dimitromanolaki, I. Yilmazturk,C., Rolland, C., Grosz, G., Nurcan, S., Beis, D., and Vgontzas, G. (1998) The ELEKTRA project : Enterprise Knowledge Modelling for change in the distribution unit of Public Power Corporation, 2nd IMACS International, Conference on Circuits, Systems and Computers (IMACS-CSC98), Athens, Greece, pp. 352-357.

Lubars M., Potts C., Richer C. (1993) A review of the state of the practice in requirements modeling,. Proc. IEEE Symp. Requirements Engineering, San Diego.

Martin J., Odell J. (1992) Object Oriented Analysis &Design, Prentice Hall, Englewoods Cliffs, NJ07632.

McGraw K., Harbison K. (1997)User Centered Requirements, The Scenario-Based Engineering Process. Lawrence Erlbaum Associates Publishers.

Mylopoulos J., Chung, L., Nixon, B. (1992) Representing and using non functional requirements: a process-oriented approach, IEEE Trans. Software Eng. Vol 18, N 6.

Olle, T.W., Hagelstein, J., MacDonald, I.G., Rolland, C., Sol, H.G., Van Assche, F.J.M., Verrijn-Stuart, A.A. (1988) Information Systems Methodologies: A Framework for Understanding, Addison-Wesley.

Pohl, K. (1994) The Three Dimensions of Requirements Engineering: a framework and its application, Information Systems Vol 19, N 3, pp 243-258.

Pohl K. (1996) Process Centered Requirements Engineering, J. Wiley and Sons Ltd.

Potts C. (1989) A Generic Model for Representing Design Methods, Proc. 11th Int. Conf. on Software Engineering.

Potts C. (1997) Fitness for use : the system quality that matters most. Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ' 97 , Barcelona, pp. 15-28.

Prakash N. (1997) Towards A Formal Definition of Methods, The Requirements Engineering Journal, 23 -50.

Ramesh B.(1993a) A Model of Requirements Traceability for Systems Development, Technical Report, Naval Postgraduate School, Monterey, CA.

Ramesh, B., Edwards, M. (1993b) Issues in the Development of a Requirements Traceability model, Proc. IEEE Symp. on Requirements Engineering, IEEE Computer Society Press, San Diego, Ca.

Ramesh, B., Powers T., Stubbs C. and Edwards, M. (1995) Implementing Requirements Traceability : A Case Study, in Proceedings of the 2nd Symposium on Requirements Engineering (RE' 95), pp89-95, UK.

Rolland C., Cauvet C. (1991) ALECSI : An Expert System for Requirements Engineering, Proc. 3th Int. Conf. on Advanced Information Systems Engineering, Springer Verlag.

Rolland C., Souveyet C., Moreno M. (1995) An Approach for Defining Ways-of-Working , Information Systems Journal, Vol. 20, No 4, pp337-359.

Rolland C., Ben Achour C. (1997) Guiding the construction of textual use case specifications. Data & Knowledge Engineering Journal Vol. 25 N° 1, pp. 125-160.

Rolland C., Souveyet C Ben Achour., C. (1998a) Guiding Goal Modelling using Scenarios, IEEE Transactions on Software Engineering, Special Issue on Scenario Management, Vol. 24, No. 12, 1055- 1071.

Rolland C., Ben Achour C., Cauvet C., Ralyté J., Sutcliffe A., Maiden N.A.M., Jarke M., Haumer P., Pohl K., Dubois E., Heymans P. (1998b) A Proposal for a Scenario Classification Framework. Requirements Engineering Journal, Vol; 3, No. 1, pp. 23-47.

Royce W. W. (1970) Managing the Development of Large Software Systems, Proc. IEEE WESCON 08.

Rumbaugh J., Blaha M., Premerlani W., Eddy F., and Lorensen W.(1991) Object-oriented modelling and design. Prentice Hall.

The Standish Group (1995) Chaos. Standish Group Internal Report, http://www.standishgroup.com/chaos.html.

Van Lamsweerde A., Dairmont R., Massonet P. (1995) Goal Directed Elaboration of Requirements for a Meeting Scheduler : Problems and Lessons Learnt, in Proc. Of RE' 95 2nd Int. Symp. On Requirements Engineering, York, pp 194 -204.

Wynekoop J. D., Russo N. L. (1993) System Development methodologies: unanswered questions and the research-practice gap,, Proc. of 14th ICIS (eds. J. I. DeGross, R. P. Bostrom, D. Robey) pp. 181-190.

Yourdon E.E (1989) Modern structured analysis, Prentice Hall, 1989.

Yu E. S. K, Mylopoulos J. (1994) Understanding Why in Software Process, Modelling, Analysis, and Design, Proc. of the 16th International Conference on Software Engineering Sorrento (Italy).

Yu E. S. K, Mylopoulos J. (1994) From ER to AR_ modelling strategic Actor Relationships for Business Process Reengineering. Proc. of the 13th International Conference on the Entity-Relationship Approach, Manchester, UK.

Yu E. S. K, Mylopoulos J. (1994) Towards Modelling Strategic Actor Relationships for Information Systems Development- with Examples from Business Process Reengineering . Proc. of the 4th Workshop on Information Technologies and Systems, Vancouver, Canada.

Zelnick N. (1998) Nifty Technology and Nonconformance : The Web in Crisis, Computer, PP115-119