CREWS Report Series 99 - 11

FROM CONCEPTUAL MODELLING TO REQUIREMENTS ENGINEERING

Colette Rolland*, Naveen Prakash+

* Université de Paris 1 Sorbonne
Centre de Recherche en Informatique (CRI)
90, rue de Tolbiac. 75013 Paris – France

\+ Dehli Institute of Technology
Kashmere Gate
110006 Dehli, India

* rolland@univ-paris1.fr
\+ np@dit.ernet.in

Submitted to : Annals of Software Engineering, Special Volume on Comparative Studies of Engineering Approaches for Software Engineering, 1999

From Conceptual Modelling to Requirements Engineering

Colette Rolland
Université Paris 1-Sorbonne
17, rue de la Sorbonne
75231 Paris Cedex 5
rolland@univ-paris1.fr

Naveen Prakash
Delhi Institute of Technology
Kashmere Gate
110006 Delhi, India
np@dit.ernet.in

Abstract

Conceptual modelling is situated in the broader view of information systems requirements engineering. Requirements Engineering (RE) explores the objectives of different stakeholders and the activities carried out by them to meet these objectives in order to derive purposeful system requirements and therefore lead to better quality systems i.e. systems that meet the requirements of their users. Thus RE product models use concepts for modelling these instead of concepts like data, process, events etc. used in conceptual models. Since the former are more stable than the latter, requirements engineering manages change better. The paper gives the rationale for extending traditional conceptual models and introduce some RE product models. Furthermore, in contrast to conceptual modelling, requirements engineering lays great stress on the engineering process employed. The paper introduces some RE process models and considers their effect on tool support.

1. Introduction

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 clearly necessary to address the issue of requirements elicitation, validation, and representation in a relatively more focussed manner. The expectation is that as a result of this, more acceptable systems will be developed in the future. The field of requirements engineering has emerged to meet this expectation.

The traditional way of engineering information systems is through conceptual modelling which produces a specification of the system to be developed. This specification concentrates on what the system should do, that is, on its functionality. Such a specification acts as a prescription for system construction.

Of the assumptions on which conceptual modelling is based, we find three very important ones :

- System requirements are highly stable, i.e., they do not change with time. As a consequence the conceptualised system is itself stable.
- System requirements are given. Users have just to be questioned about their requirements. Thus, the interesting problem is that of specifying the system to meet these requirements ; and system analysts are the right persons to do it.

- Validation of system requirements can be done with reference to system functionality. In other words, the conceptual schema is the appropriate support for communicating, negotiating and reaching an agreement with users and system stakeholders.

Today, it is becoming clear that these assumptions do not hold any longer. Due to economic pressure and emergence of new technologies, organisations change much faster than before. As a consequence, expectations from information systems also change much faster which, in turn, implies that requirements are no longer stable [Harker93]. The issue of understanding the effect of business changes and recording decisions made about their impact on requirements is noted by Lubars et al [Lubars93] as an issue which has not been solved yet. It is also known that requirements even change during development and cause considerable problems during development as reported by Curtis et al [Curtis88]. Since requirements change, it is no longer possible to treat them as given. Rather, it is necessary to determine new requirements for legacy systems and to carry requirements models through the entire systems life cycle. Further the central role of system analysts shall be replaced by the participation of a large variety of stakeholders bringing their various view points on what the system should do [Finkelstein90]. Finally, requirements validation must now be rooted in organisational change rather than in system functionality : if requirements models are to be validated then, this validation must be with reference to organisational needs rather than system functionality. It is only then that computer based systems will be able to adapt to changing organisational needs.

In tackling these problems, the area of requirements engineering tries to go beyond the functionality based view of conceptual modelling. We highlight here two dimensions along which this attempt is made :

- 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 an 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 are done with respect to their purpose in organisations and (b) only organisationally purposeful systems are conceptualised.
- requirements engineering does not treat the functionality of a system, what is done by the system, as abstracted by system analysts from a set of given requirements. Rather, it considers that the potential users of the system are most suitable to provide useful and realistic view points on the system to be developed. Therefore, a detailed exploration of the various ways the system might be used and the activities it shall carry out is performed. This can be done, for example, by looking at typical interactions that are expected to occur with the system (use cases). This exploration leads to the identification of 'normal' and 'exceptional' activities, whose integration models the full system behaviour. In this sense, the determination of what the system must do is itself an interesting question in requirements engineering.

To deliver the foregoing, an appropriate way of doing requirements engineering must be found and supported by computer based environments. First, changing requirements imply that the assumptions made, the decisions taken, and the alternatives explored must all be recorded and be made available for future use. Second, requirements engineering is known to be a complex task and advice/guidance on which activities are appropriate in given situations as well as on how these activities are to be performed must be provided. Finally, considerable freedom in deciding which activity is to be done next must be made available to the requirements engineer.

The foregoing indicates that there are three interesting aspects of requirements engineering, namely, *conceptualisation of purposeful systems, modelling of system usage,* and *the process support* needed for doing requirements engineering. We will highlight these in the rest of this paper. The attempt will be to show that these three aspects represent a basic departure from conceptual modelling.

In the next section we review the area of conceptual modelling. Thereafter, we turn our attention to requirements engineering and consider separately the three issues of conceptualising purposeful systems, modelling system usage and the process support.

2. Conceptual Modelling

Traditionally information system engineering 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 conceptual modelling can be treated as the first phase of the two-phase organisation of the information system life-cycle shown in figure1. It 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.

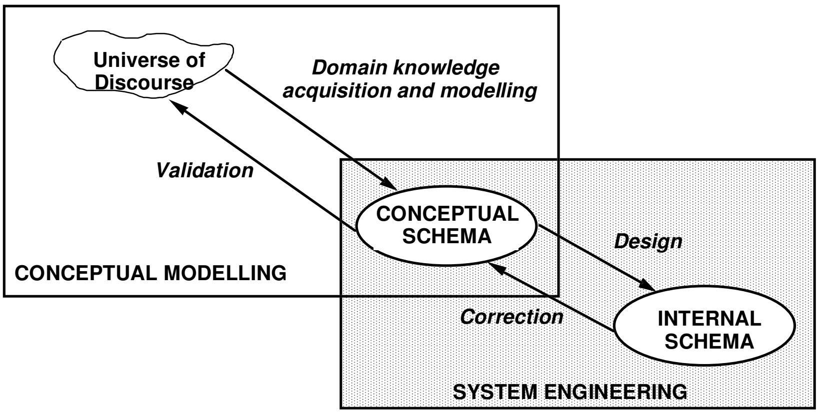

Figure 1 : Two-phase organisation of system life-cycle

2.1 Classification framework of conceptual models

The conceptual schema plays a central role in this organisation of the information system life cycle and the information systems community has developed a large number of conceptual models, systems of concepts, 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 figure2, this framework has been shown as defining a three-dimensional space within which conceptual models can be positioned.

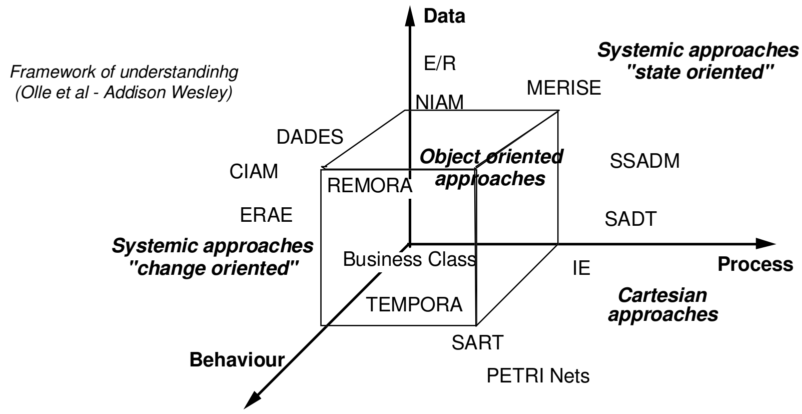

Figure 2 : The three dimensional framework for classifying conceptual models

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 ; when seen 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. Such a hierarchy identifies the functional components of the information system function but 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 yet 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 and this leads quite naturally to the data processing approach. 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. Therefore, an identification of this information is of the essence. 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 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.

2.2 Conceptual modelling process

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 previously developed conceptual schemata or 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].

To sum up

(1) It was felt that every information system served a purpose in the organisation. However, it was not always possible to articulate this purpose. Thus, for example, in the process oriented approach the purpose of the information system was to carry out a function. In small systems it was possible to precisely identify this function but as systems grew in complexity, it became more and more difficult to do so. Since the purpose was not clear, it was difficult to determine the boundaries of the information system.

(2) It can be seen that conceptual modelling specifies in the conceptual schemata, the functions expected from the system to be engineered. This specification is expressed in a formal way and is driven by the technologies available for system development. Finally, it focuses on the view of the system that is held by system analysts. In contrast, requirements belong to users [Macaulay93], [Harker93]. Requirements specifications express objectives, goals, intentions of users. This is done in an informal manner and the drive is not technology-oriented but

usage-oriented. Finally, the requirements specification reflects the views of the users and stakeholders of the system rather than of the system analysts.

As a consequence of these differences between conceptual schemata and requirements specifications, relating requirements to conceptual schemata is difficult. Thus for example, it is problematic to check whether or not the conceptual schema meets the requirements and to reflect changing requirements in conceptual schemata.

In order to deliver accurate, valid and complete specifications, requirements engineering has to address three main issues, those of :

- extending conceptual modelling in order to establish a conceptual link between requirements and system functions. As a result, the purpose of systems shall become evident. This shall ensure that *purposeful systems* shall be built at all times.
- deriving system functionality from the multiple view points of users such that systems are adapted to the *usage* that stakeholders envision for them. This shall lead to better system acceptability in organisations.
- modelling the requirements engineering *process* in order to provide the methodological guidance for requirements elicitation, requirements description, and requirements refinement activities, as well as for transformations of requirements into system specification.

These three issues affect both the product and the process aspects of requirements engineering that we consider in turn in the two following sections. The 'Requirements Engineering Models' section deals with the product aspects whereas the 'Requirements Engineering Process Support' deals with the process aspects.

3. Requirements Engineering Models

In the view of requirements engineering being proposed here, we consider that requirements come from two sources, users on the one hand, and the domain environment on the other hand. The first source provides informal statements of goals and users' intentions expressed in natural language. The second source provides requirements reflecting real world facts and constraints on the designed system implied by laws of physics independently of any user' need or wish. Hence requirements may occur into two sub-types :

- user-defined requirements which arise from people in the organisation and reflect their goals, intentions and wishes,
- domain-imposed requirements which are facts of nature and reflect domain laws.

This implies that the Universe of Discourse has to be partitioned into two, the *usage world* and the *subject world* [Jarke93]. The *usage world* describes the tasks, procedures, interactions etc. performed by agents and how systems are used to achieve work, including stakeholders who are system owners, direct and indirect users. It can be looked upon as containing the objectives that are to be met in the organisation and which are achieved by the activities carried out by agents. Therefore it describes the activity of agents and how this activity leads to useful work.

The second part of the Universe of Discourse, the *subject world*, contains knowledge of the real world domain about which the proposed system has to provide information. It contains real world objects which are to be represented in the conceptual schema.

There is a third world, the *system world* which is the world of system specifications in which the requirements arising from the two worlds must be addressed. The system world holds the modelled entities, processes, events of the subject and usage worlds with some mapping from these conceptual specifications to the design and implementation levels of the software system.

All these worlds are interrelated as shown in Figure 3. User-defined requirements (sub-type 1 above) are captured by the two relationships between the usage and the system world, namely the *intentional relationship* and the *usage fit relationship*. Domain-imposed requirements (sub-type 2 above) are captured by the *domain genericity relationship* in the Figure.

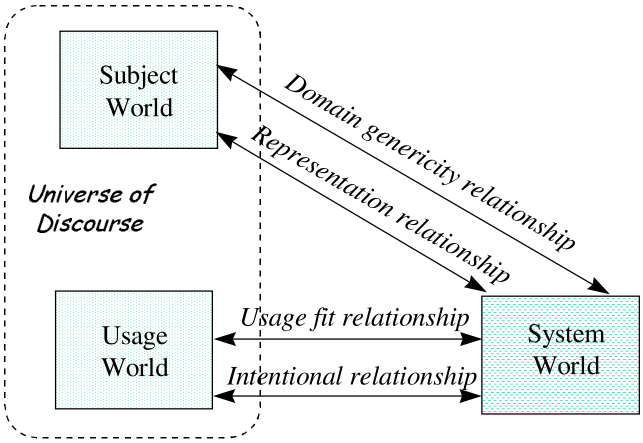

Figure 3 : The relationships between the usage, system and subject worlds

Finally, it shall be noticed that there is a *representation relationship* between the subject world and the system world which relates the domain to its representation in the system. This relationship has been the only focus of conceptual modelling whereas requirements engineering highlights the importance of the three other relationships, namely the intentional, usage fit and domain genericity relationships. All three relationships address the 'why' question and therefore provide the rationale for system development. We consider these relationships in more detail in the subsequent part of this section.

3.1 Relationships between the usage world and the system world

The usage world consists of individuals, social groups and organisational settings in which the system is intended to function [Goguen93]. The *individual*, *pragmatic* relationship with the system world is provided by the *usage fit relationship* of Figure 3 whereas the social, *semiotic relationship* is provided by the *intentional relationship*.

The usage world provides the rationale for building a system. The purpose of developing an information system is to be found outside the system itself, in the *enterprise,* or in other words, in the context in which the system will function. The social relationship between the usage and system world addresses the issue of the system purpose and relates the system to the organisation goals and objectives. This relationship explains *why* the system is developed. Modelling this establishes the conceptual link between the envisaged system and its changing environment. This suggests an augmentation of conceptual modelling to deal with the description of the context in which the system will function and will be used. In the area of requirements engineering, *goal-driven approaches* have been developed which directly model organisational objectives and relate them to system functions. These approaches address the semiotic, social link between the usage and the system world.

As brought above, the usage world is the world of the system users who will individually work with the system to meet the objectives assigned to them by the organisation. Additionally, each of them has his/her own view point and requirements regarding the system to be constructed. Taking these into account helps in the construction of relatively more acceptable systems. This suggests another augmentation of conceptual modelling, to include the role of individual thereby enabling the derivation of system functionality from the integration of users' view points. In requirements engineering, the areas of *scenario modelling* and *use case development* take this into account.

As indicated earlier, *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 further information for the interpretation of requirements before they become understood and can be transformed into system function specifications and therefore, aim to support conceptualising purposeful systems. *Scenario based approaches* by focussing on the users' view points help modelling purposeful system usage from which useful system functions can be derived. The latter provides dynamic meaning to the former whereas the former provides the intentional setting within which the usage world finds meaning.

In the foregoing, we have presented our view of requirements engineering, namely that it is concerned with the relationship between three worlds. In terms of Figure 3, conceptual modelling approaches have focused on the representation link to the exclusion of the others. This practice provides an answer of sorts to the fundamental question : what does the information handled by my system means ? However, it also tends to draw attention away from a number of equally fundamental questions concerning other type of knowledge required to engineer an information system. Looking more carefully to these helps in understanding how to engineer purposeful systems, and leads to systems fitting the constraints and laws of the domain as well. Distinguishing the subject world and the usage world therefore, helps in making explicit the relationships which were ignored by traditional conceptual modelling.

3.1.1 Goal driven approaches

The broader view of a requirements specification that we advocate here is one that goes beyond the classical conceptual schema describing system functionality. It includes *enterprise* modelling which represents the *why* part of system requirements. This part complements the *what* part provided by conceptual functionality modelling.

Enterprise modelling has been developed for example, in the F3 project [Bubenko94a], [Bubenko94b] to provide a set of models for understanding the requirements and bridging the gap between ill-defined problems and application situations on the one hand and specification of the formal, precise definition of requirements of the information systems on the other hand. The requirements specification is represented as a structured description of five interrelated sub-models (see Figure 4) which provide the context within which the requirements are elicited. Each sub-model represents a particular concern or view in requirements acquisition, and these sub-models help separating the different concerns in a workable way. The sub-models are not developed in a linear, sequential manner. Although the process usually starts with an objectives model and progresses through actor and activity models to information systems requirements this is not always the case. For instance, with an existing systems the activity and concepts models may be developed first by reverse engineering previous designs.

The *objectives* sub-model describes the *why* component of a requirements specification. It is a graph with components such as goals, problems, opportunities and weaknesses as nodes connected through relationships of the type ‘motivates’. The objectives sub-model is related to rationale models such as IBIS [Potts89] but it contains a goal decomposition hierarchy close to other proposals such as [Dardenne91], [Rolland93],[Yu94b].

The *concept* sub-model is used to define the Universe of Discourse that concerns requirements engineers. It may serve as a dictionary of user and customer defined concepts.

The *actors* sub-model is used to define the actors in the domain and their relationships with activities and objectives. Actors may be individuals, groups, roles, organisational units, systems, etc. Actors in sub model are related to goals in the objectives sub-model and therefore represent stakeholders who are responsible for achieving goals through activities described in the activities sub-model.

The *activities* sub-model describes the organisational activities, i.e. the processes and tasks of the enterprise. Components in this sub-model are created to achieve goals in the objectives sub-model, referring to components of the concepts sub-model, and resources required to carry out these activities described in the actors sub-model.

The information system requirements sub-model is meant to be derived from the other models. It includes both functional and non functional requirements. The former typically indicate needs for establishing objects, defining operations and services in the Object Oriented terminology or functions in top-down decomposition such as Structured Systems Analysis [Yourdon89]. The latter are related to the environment, performance and quality of the required system.

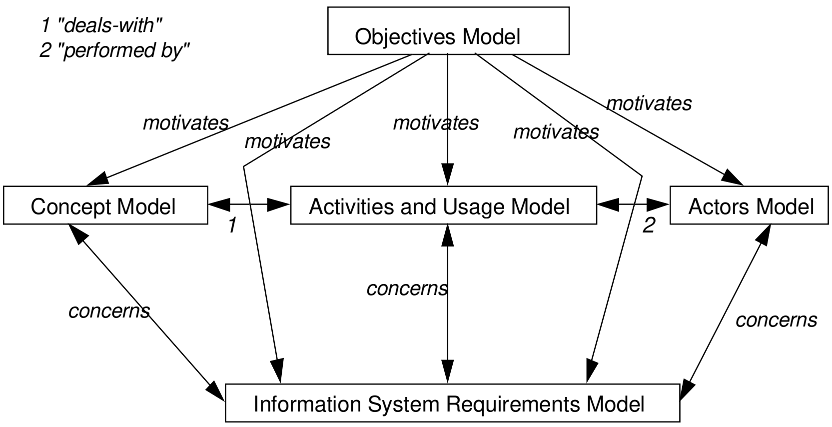

Figure 4 : The sub-models of the F3 approach

Enterprise modelling offers a set of interrelated models, each constructed with a set of predefined components types and relationships to address the ‘why’ question and understand where the ‘what’ requirements come from. The semantic links from the set of interrelated sub-models and the information system requirements model are established for reflecting the rationale, the motivation, for designing a specific information system. Enterprise modelling was one of the first attempts to understand goals, the primary ‘needs’ expression of users in context of a problem domain. Early experiences have shown [Kirikova94], [Willars92] that the use of a goal driven approach leads to improved understanding of the problem realm for decision makers, requirements holders and customers as well as for developers.

Since then there is convergence on the view that *goal modelling* is an effective way to identify requirements [Potts97], to elicit high-level goals to be achieved by the envisioned system [Dardenne93], [Anton96], [Bubenko94b], to help in the refinement of these goals [Dardenne93], [Yu94a], [Yu94c],[Rolland 98a] and their operationalisation into system requirements specifying how goals should be accomplished by the proposed system [Anton96].

Enterprise modelling was further refined in the EKD method to support change management [Loucopoulos98], [Rolland97b], [Kardasis98], [Rolland98b].The focus here is on the evaluation of alternative scenaria for change and the selection of the most appropriate one [Nurcan99a], [Nurcan99b]. In the KAOS approach [Dardenne91], [Dardenne93], the emphasis is directed towards supporting formal refinement of high level goals into system constraints meant as functional requirements. Although generic models are advocated, goal modelling and refinement have supplied simple guidance via heuristics [van Lamsweerde95]. The I* approach [Yu94a], [Yu94b], [Yu94c] create models of the environment of the system that emphasises modelling agents and their relationships. Their strategic dependency and rationale models allow tracing of dependencies between agents, the goals and tasks and support reasoning to identify trade-off between functional requirements and non functional requirements[Mylopoulos92] (referred to as 'soft goals')[Chung96]. Similarly emphasis on agents and their relationships is underlying the Albert language aiming at formally specifying requirements [Dubois94] in order to ease their validation through scenario generation. The GBRAM method [Anton 96] that is built over the Inquiry Cycle [Potts94] draws our attention on the nature of the elicitation process that is viewed as a deliberation process among the various stakeholders involved with requirements engineering.

Although goal modelling has proved to be useful for specifying purposeful systems, practical experience tends to show that there are still a number of difficulties. First, even it is often assumed that systems are constructed with some goals in mind [Davies93] in reality [Anton96], [Elektra98] goals are not given and therefore the question as to where they originate from [Anton 96] acquires importance. In addition, enterprise goals which initiate the goal discovery process do not reflect the actual situation but an idealised environmental one. Therefore, proceeding from this may lead to ineffective requirements. Eliminating uninteresting and spurious goals is necessary and difficult [Potts97]. Thus, goal discovery is rarely an easy task. Additionally, it has been shown [Anton96] that the application of goal reduction methods [Dardenne93], to discover the components goals of a goal, is not as straight-forward as literature suggests. Finally it seems to be difficult to deal with the fuzzy concept of a goal. This was apparent when we applied the goal driven approach as embodied in the EKD method to several domains air traffic control, electricity supply, human resource management, tool set development [ELEKTRA97], [Loucopoulos94]. This led to some formalisation of the goal formulation [Prat97],[Rolland97a]. Yet, domain experts need to discover the goals of real systems.

It is thus evident that help has to be provided so that goal modelling can be meaningfully performed. This help must (a) facilitate the work of the domain expert by *getting over the problem of the fuzzy nature of goals* (b) *help discover goals* and (c) *aid in the task of goal reduction*.

3.1.2 Scenario based approaches

Independently of goal modelling, an alternative approach to RE, the *scenario-based approach*, has been developed. 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 [Potts 94], to help in the discovery of exceptional cases [Pots94], [Rolland 98a], [Sutcliffe98], to derive conceptual object-oriented models [Dano97], [Rumbaugh91], [Jacobso95], [Rubin92], to understand needs through scenario prototyping [Hsia94] and animation [Lalioti95] to reason about design decisions [Caroll95], [Young 87], 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. The industrial practice survey conducted by the CREWS consortium confirms that scenarios are useful in particular when abstract modelling fails [Weidenhaupt98]. In addition, since scenarios describe concrete behaviours, they capture real requirements. However, because they deal with examples and illustrations, scenarios are inherently partial and only provide restricted requirements descriptions which need to be generalised to obtain complete requirements.

Scenarios have been developed [Rolland98c] for different *purposes* with different *contents*, expressed in different *levels of abstraction* and with different *notations*.

In so far as their *purpose* is concerned, scenarios can be *descriptive*, *explanatory* or *exploratory*. *Descriptive* scenarios [Potts94] capture requirements by enabling the analyst and users to walk through a process and understand its operations, actors, the events triggering the process etc. Thus, descriptive scenarios aid in the clarification of how a process performs, who are the involved parties and how the process is activated as well as the conditions under which it is activated. *Explanatory* scenarios [Wright92] raise issues and provide rationale for these issues. They identify why something happens in the real world, what leads to it, what are its causes, what are commonly occurring events which require handling etc. Through this the attempt of explanatory scenarios is to describe the desirable features of the system to be developed. Finally, *exploratory* scenarios [Holbrook90] are useful when different possible solutions for satisfying a given system required exist. These solutions are to be examined and evaluated to arrive at the right solution. Such scenarios establish a direct link between requirements and desired solutions.

As mentioned above, scenarios have different *contents*. This can be behavioural information identifying the actions, activities, events carried out in the usage world; a description of the objects of the subject world together with their attributes; events and event histories; organisational information like the structure of the company, the groups, departments and agents found in it etc.; or even stakeholder information including the characteristics of people, their views and aspirations [Nardi92]. However, by and large scenarios concentrate on the functional features required of a system [Firesmith94], [Glinz95], [Olle88], [Potts94], [Rubin92], [Rawsthorme96], [Somé96].

Finally, scenarios have been expressed at three different *levels of abstraction*, instance, type and mixed. In the former case [Caroll95], [Potts94], [Young87], a scenario uses specific names or events with real argument values. These scenarios describe particular instances of use which can form the basis for discussion of what happens, why and how. Type scenarios [Hsia94], [Jacobson92] do not use individual entities but entity types. Thus they do not refer to Smith but to customers. Each execution of a type scenario is an instance scenario. Finally, mixed scenarios [Rolland98] are those that have some parts at the instance level and others at the type level.

Scenarios have been expressed in different *notations* ranging from the *informal*, *semi-formal* to the *formal*. *Informal* scenarios use natural language [Rolland 97a], [Erickson95] , [Holbrook90], videos [Wood94], [Haumer98] story descriptions etc. and are valuable in those cases where the user community is unwilling/unable to deal with formal notation. *Semi-formal* scenarios use a structured notation like tables [Potts 94] and scenario scripts [Rubin92] in capturing real activities. Finally *formal* scenarios are expressed in modelling languages based on regular grammars [Glinz95]or state-charts [Harel87] . They are useful to run as simulations to present a vision of what the future system will look like and to gauge user reactions to it.

3.1.3 Coupling goals and scenarios

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. In [Dano97], [Jacobson95], [Leite97], [Pohl97] goals are considered as contextual properties of use cases whereas in [Cockburn95] they are used as a means to structure use cases. The goal scenario combination has been used to operationalise goals [Anton96], [Holbrook90], [Potts94], [Rolland98a], to check whether or not the current system usage captured through multimedia scenarios fulfils its expected goals [Haumer98], to infer goals specifications from operational scenarios [van Lamsweerde98] and to discover new goals through scenario analysis [Rolland98a].

As an example of an approach which combines goal modelling and scenario authoring let us consider the CREWS-L'Ecritoire approach [Rolland97a], [Rolland98a] developed within the CREWS ESPRIT project. CREWS-L'Ecritoire uses *a bi-directional 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 pro-actively 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. In order to give some insights of the approach, we first present some of the key concepts and terminology of the CREWS-l'Ecritoire approach and then provide a brief overview of its process.

(a) Concepts and terminology

- A ***Requirement Chunk*** (RC) is a pair <G, Sc> where G is a goal and Sc is a scenario. Since a goal is intentional and a scenario is operational in nature, a requirement chunk is a possible way of achieving the goal.

- A ***goal*** is defined as "something that some stakeholder hopes to achieve in the future" [Plihon98]. In our approach, a goal [Prat97] is expressed as a clause with a main verb and several parameters, where each parameter plays a different role with respect to the verb. A detailed description of the goal structure can be found in [Rolland98a]. An example of a goal expressed in this structure is the following :

  Provide $_{verb}$ (efficiently) $_{qual}$ (electricity) $_{tar}$ (from PPC producer) $_{so}$ (to our non eligible customer) $_{ben}$ (using the PPC network) $_{means}$ (in a normal way) $_{manner}$

- A ***scenario*** is "a possible behaviour limited to a set of purposeful interactions taking place among several agents". It is composed of one or more *actions*, an *action* being an interaction *from* one agent *to* another. The combination of actions in a scenario describes a unique path. A scenario is characterised by initial and final states. An *initial state* attached to a scenario defines a precondition for the scenario to be triggered. A *final state* defines a state reached at the end of the scenario. We distinguish between *normal* and *exceptional* scenarios. The former leads to the achievement of its associated goal whereas the latter fails in goal achievement.

- ***Requirement chunks classification*** and ***abstraction levels*** : We have introduced three levels of abstraction called *contextual*, *functional*, and *physical*. The contextual level identifies the services that a system should provide to an organisation and their rationale. The functional level focuses on the interactions between the system and its users to achieve the needed services. Finally, the physical level deals with the actual performance of the interactions. Each level corresponds to a type of requirement chunk. As a result, we organise the requirement chunks collection in a three level abstraction hierarchy.

- ***Relationships between requirement chunks***: There are three types of relationships among requirement chunks namely, the *composition*, *alternative*, and *refinement* relationships. The first two of these lead to a horizontal AND/OR structure between RCs. These are extensions of conventional AND/OR relationships between goals. AND relationships among RCs link together those chunks that require each other to define a completely functioning system. RCs related through OR relationships represent alternative ways of fulfilling the same goal. The third type of relationship relates requirement chunks at different levels of abstraction. The refinement relationship establishes a vertical link between requirement chunks.

(b) The requirements elicitation process

The CREWS-L'Ecritoire process aims at discovering/eliciting requirements through a bi-directional coupling of goals and scenarios allowing movement from goals to scenarios and vice-versa. As each goal is discovered, a scenario is authored for it. In this sense the goal-scenario coupling is exploited in the forward direction from goals to scenarios. Once a scenario has been authored, it is analysed to yield goals. This leads to goal discovery by moving along the goal-scenario relationship in the reverse direction.

The exact sequence of steps of the process is as follows :
1. Initial Goal Identification
repeat

2. *Goal Analysis*
3. *Scenario Authoring*
4. *Goal Elicitation Through Scenario Analysis*

until all goals have been elicited.

It can be seen that goal elicitation and scenario authoring are complementary steps and goals/requirements are incrementally discovered by repeating the goal-analysis, scenario-authoring, goal-elicitation-through-scenario-analysis cycle. Each of the three steps of the cycle is supported by mechanisms to guide the execution of the step.

The guidance mechanism for goal analysis is based on a linguistic analysis of goal statements. It helps in reformulating a narrative goal statement as a goal template as introduced in the previous section. The mechanism for scenario authoring combines style/content guidelines and linguistic devices. The former advise authors on how to write scenarios whereas the latter provide semi-automatic help to check, correct, conceptualise, and complete a scenario. Finally, for goal elicitation through scenario analysis, we defined enactable rules offering three different goal discovery strategies namely, refinement strategy, composition strategy, and alternative strategy. The first of these discovers goals at a lower level of abstraction than a given goal ; the second discovers goals ANDed to the original one ; the last discovers goals ORed to the original goal.

To sum up

To sum up the impact of the usage world and its relationships with the system world, it seems that there is convergence for shifting from *conceptual modelling* to *intentional modelling*. This permits to understand 'why' an information system is needed and therefore, helps avoiding to specify a system which does not meet the expectations of the organisation. Further, in order to relax the constraints posed by abstract notations traditionally used in conceptual modelling, informal *narrative scenarios* seem to be suitable to ease the communication with the users community and get their points of view on how they envision and expect the system to behave. Finally the recent attempts to combining goal modelling and scenario authoring seem to overcome the difficulties encountered in practice when using each of the approaches in isolation. However, there is no practical evidence yet, that the percentage of successful systems will be significantly increased due to this move from conceptual modelling to enterprise modelling.

3.2 The domain genericity relationship between the subject and system worlds

As mentioned before, there is a *representation* relationship between the subject world and the system world which has been pointed out a long time ago by researchers in information systems [Rolland82], [Borgida84], [Brodie82] and temporal databases [Rolland79], [Bubenko77], [Clifford83]. However the focus has been on objects, events, operations, etc., i.e. on the functional aspects of the information system.

Non-functional quality criteria such as confidentiality, performance, accuracy and timeliness of information can also be attached to this representation relationship. Non-functional requirements modelling is a desirable feature of future requirements engineering methods which is rarely integrated in current methodologies.

There is another modelling concern captured through the *domain genericity relationship* that is the role and impact of domain knowledge [Jackson94]. Many new applications share the same requirements with well known problems, so one possibility is to create *generic domain models* of such problems as templates for requirements of certain classes of applications. Approaches along this relationship between the subject and the system worlds attempt to add reuse to requirements engineering by providing sets of predefined generic requirements for developing system requirements specification. These sets can be reused to elicit the requirements of a specific application or to validate a requirements specification. The hope is to increase the efficiency of requirements engineers as well as to lead to a better quality specification

Dependencies between systems and their environment have been analysed in detail by Jackson [Jackson 94], and Jackson and Zave [Jackson93]. Jackson points out that domains impose obligations on the required system. He formalises event dependencies between the system and its environment that are inherent to the laws of physics e.g. obligations for the required system in avionics and other real-time applications, and events that may arise in human failure, and shall impose requirements on the system to design.

The separate consideration of the subject world allows the development of so-called *domain ontologies* which consider typical classes of object and activity abstractions as reusable modelling patterns which can significantly reduce the requirements engineering effort [Feather92]. A model library for the subject world has been for example, developed in the NATURE project [Maiden94a], [Maiden94b], [Sutcliffe94]. A model is a problem abstraction which defines in generic terms the structure and the behaviour of the problem space. It is a unit of abstraction that aggregates objects linked by a purpose. The concepts used to define object models are shown in figure 5.

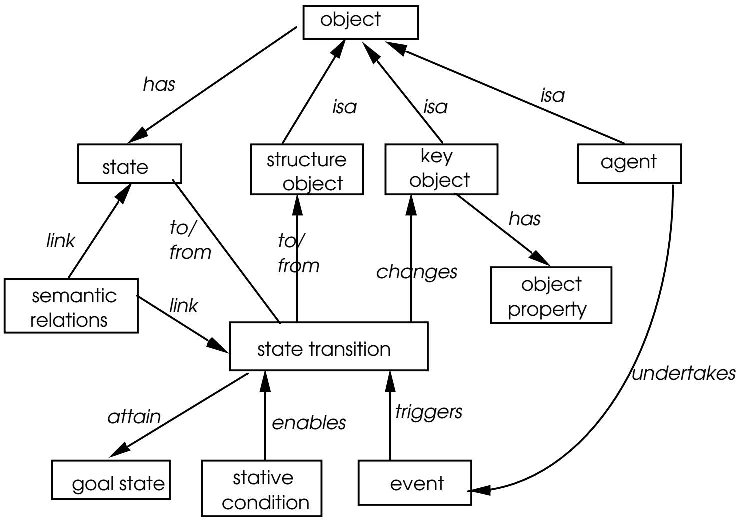

Figure 5 : Meta-schema for domain modelling

Objects : there are key objects that undergo state transitions and agents who undertake behaviour and hence cause change. Objects have properties and states.
Structure objects : model containment of objects for example a library contains books.
State transitions :model behaviour of objects and enable goal state to be achieved.
Events : model initiating 'triggers' and time points.
Stative conditions are tests on objects' states.
Goal states : describe a required state that should be satisfied.
Semantic relationships specify constraints between objects, state transitions and states.

Object models are structured in a class hierarchy. Models at different levels of abstraction are distinguished using different types of knowledge. Object structure and purpose are the most important constructs at higher levels because they discriminate effectively between different problem classes. The top level in the hierarchy is defined by state transitions, agents, states and semantic relations. Lower level object models are specialised by adding further knowledge such as goal states, events, conditions and object properties. The highest levels of the object class hierarchy are illustrated in figure 6.

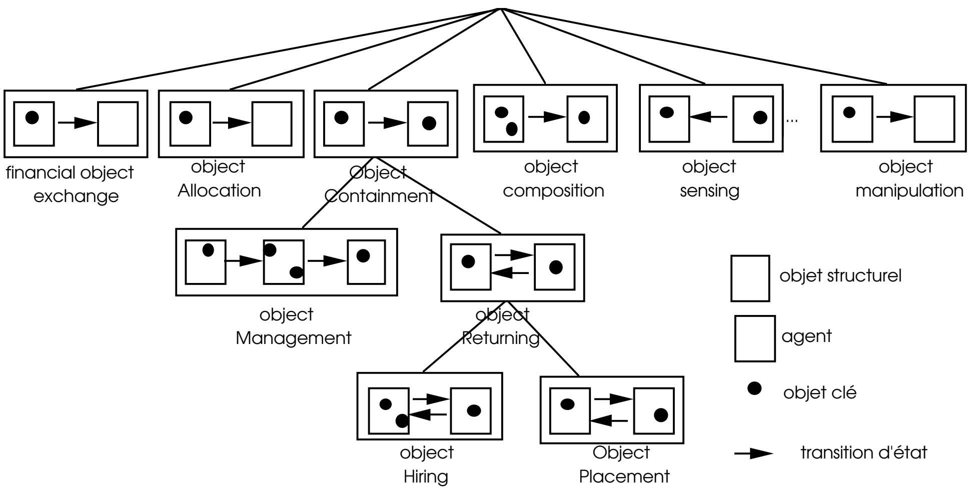

Figure 6 : Hierarchy of object class models

The library is used in the NATURE toolset by the matcher to identify the models relevant for the application at hand and therefore, identify the generic requirements inferred by the domain and that must be part of the system requirements specification.

Several authors have proposed generic knowledge for requirements engineering e.g. clichés in the Requirements Apprentice [Reubenstein91] and generalised application frames in the ITHACA project [Fugini90]. In software engineering, libraries of abstractions have been described at the level of design architectures [Shaw91] and reusable functions, although these have limited success in practice [Pietro-Diaz91], [Harandi91]. Similar abstract classes have also been proposed in Artificial Intelligence, [Greuber92], [Chandrasekaran92]. Knowledge engineering methods, notably KADS [Wilenga93] have also espoused the use of generic domain classes, although only a small number of classes have been described in detail.

4. Requirements engineering process support

We now turn our attention to the third interesting aspect of requirements engineering namely, that of providing sufficient process support for arriving at an accurate and complete requirements specification. Whereas it was recognised in conceptual modelling that development had two aspects, the product and process aspects, it was only the former that was really investigated. Thus, methods for conceptual modelling are a maze of steps, guidelines, checklists, heuristics etc. that are far to provide the help and guidance to build good products [Wynekoop93]. It was assumed that the process of development was linear, Cartesian in nature. Therefore, it was quite usual to base methods and tools on process models like the Waterfall model [Royce70].

In contrast, requirements engineering has explicitly considered the issue of the process support to be provided. This is because, as discussed earlier, requirements engineering is an exploratory activity driven by the objectives and activities carried out by the stakeholders.

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* must be actively supported by *computer assisted tools.* We comment these three aspects of process guidance, process tracing and tool support in turn.

4.1 Guidance

In so far as the issue of guidance is concerned, some experience 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. Current workflow and software process models [Osterweil87], [Finkelstein94] are not directly applicable [Lehman87] as they assume broad coverage and fairly strict control of the process. *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 performed transformations and their 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 thus better reuse of results [Potts89] but the flexibility and situatedness of requirements engineering processes is not adequately covered in existing decision-oriented models.

The importance of situatedness in process modelling is also acknowledged by the software engineering community where it was soon found that a strict prescription of the process model led to dissatisfaction. This was because, in actual practice, departures from the prescription occurred [Russo95]. Therefore, 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 [Cugola95], Cugola96], [Cugola98]. 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 [Dowson94], [SiSaid96], [Jacherri92], [Finkelstein94], [Bandinelli93, [Belkhatir94]. 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. Thus, efforts were made to avoid prescription of a process model and to focus on alternative generation. Since real requirements engineering processes also need prescriptive components, facilities to build these were provided. However, the focus was on alternative generation. This can be seen in the IBIS process model [Potts89], [Ramesh93b] 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], [Rolland94], [Rolland95], [Polh96] 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 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]. There were decisions of three types, atomic, complex and abstract related to each other by two different kinds of dependencies. The first of these identified which decisions can be performed after a given one whereas the second one identified those that must be performed after it, though not necessarily immediately. Recognising the crucial role played by the product situation, the interest was in generating the set of decisions that were applicable to a given product situation. Prescriptive capability was provided through the notion of complex decisions which could be built out simpler decisions whose order of execution was prescribed. Finally, abstract decisions could be built as abstractions and provided high level abstract choices for application engineers.

Experience with the contextual and decisional models showed that a key discriminant factor in real processes is the product situation. This situation has a strong bearing in selecting the *task* best suited to handle it and also the *strategy* to be adopted in carrying out this task. These strategies need to be reflected in the process model so that the right one can be dynamically chosen. A recent attempt to model the strategic dimension of the requirements engineering process through a set of strategies to select tasks adapted to situations was made in [Rolland99] and [Ralyte99]. The process model is represented as a labelled directed graph called a map. The map uses two fundamental notions, *intention* and *strategy*. An intention captures in it the notion of a task that the requirements engineer intends to perform whereas the strategy is the manner in which the intention can be achieved. The nodes of the map are intentions whereas its edges are labelled with strategies. The directed nature of the map identifies which intention can be done after a given one. The only way in which a process can be built is dynamically, through the use of guidelines for selection among alternatives. Only after the intention and the strategy have been decided is there a need for a guideline to achieve the intention. There are three guidelines associated with the map :
- *intention selection guidelines* for determining all succeeding intentions of a given one,
- *strategy selection guidelines* for determining the strategies from which one is selected,
- *intention achievement guidelines* for defining the way in which an intention can be achieved.
Thereafter, the enactment mechanism is invoked to actually carry out the tasks.

It can thus be seen that the requirements engineering community has made a conscious effort towards developing guidance to meet its two most basic needs :

(a) generating the set of alternatives applicable to a given product situation and
(b) reducing the amount of prescription to only those parts of the process where it is essential.

4.2 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 [Gotel94], [Polh94a], [Ramesh93a], [Wright91] 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]. Among others, these reports indicate that requirement traceability is a vital component in implementing a quality system, essential for consistent change integration, lead to less errors during system development, play an important role in contract situations, and improve system acceptance.

Through experience one has discovered that there is a true need for RE processes which are traceable. To make a process traceable means to capture knowledge during the progress of a process in order to later be able to restore the entire or certain decided on aspects of this process. The key issue is that in large projects, over time and with changing development groups, the rationale for and the context of key decisions are confused and even lost [Curtis88]. Process traceability can be divided in three parts [Pohl96] :

- Process execution traceability, i.e. the recording of data that enables the reassembly of the sequence of steps that makes out the executed process.
- Product evolution traceability, i.e. the recording of data that enables you 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 down to design and 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. Thus, there is a differentiation between these two kinds of traceability : the former is referred to as *post-traceability* whereas the latter is called *pre-traceability* [Gotel94], [Davis90], [IEEE-830].

There exists some object oriented methods which recommend product traceability among others Class/Relation [Desfray94], OOSE [Jacobson92] and rAdar [Bari90]. There are some approaches [Alfort90], [Flynn90] and even some commercial tools (e.g. RT from Teledyne Brown Engineering, RMT from Marconi Systems Technology , RDD100 from Ascent Logic Corporation) which support post-traceability. In contrast there are only recent research contributions [Gotel94b], [Kaindl93], [Pohl96], [Ramesh93b], [Ramesh95] and systems (e.g. DOORS from QSS) focusing on pre-traceability. However, the results gained from the surveys of [Ramesh93b], and [Gotel94] indicate that requirements pre-traceability is at least as important as requirements post-traceability ; especially for systems which are embedded in a constantly changing environment.

An interesting framework for requirements pre-traceability was provided by Pohl [Pohl94a] which described the requirements engineering process in a three dimensional space (see Figure 7). The framework assumes 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. These three facets lead to a *three dimensions framework* in which the process of requirements engineering can be traced :

-The complexity of the individual/cognitive aspect of the RE process leads to the *specification dimension* which describes the degree of completeness of the requirements specification.

-The *social usage* aspect leads to the agreement dimension which describes to what degree the members of the RE team agree on the requirements specification.

-The *system aspect* leads to the representation dimension which describes how requirements are technically described, e.g. their degree of formal semantics.

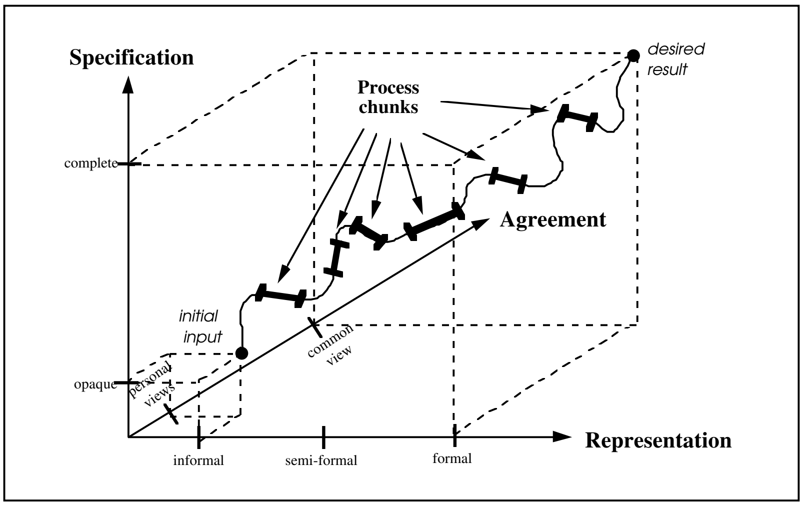

Figure 7 : The three dimensions of the RE process.

As shown in Figure 7 the trace of the requirements engineering process is modelled as a curve within the three dimensions, 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 the actual process performance i.e. to the process steps and the agents by which it was produced.

Therefore it can be seen that the requirements engineering community has made a conscious effort to understand the various facets of traceability, to define the information that make a process traceable and to identify the roles that trace can play in the development of systems.

4.3 Computer tool support

Conceptual modelling as part of system development is facilitated by the use of automated support in the form of CASE tools. A wide variety of CASE tools and CASE environments called Integrated CASE (ICASE) or Integrated Project Support Environment (IPSE) have been built to support specific methods. It has been pointed out [Norman92] that CASE tools have been successful in automating many routine tasks of system development. Wijers [Wijers91] says that though the possible list of things that CASE tools can do is quite large, they have been essentially successful in providing documentation and verification support. Today's tools therefore have excellent facilities for editing and maintenance of graphical specifications but they lack many of the functional features as projected in CASE architectures like that of Bubenko [Bubenko92]. Some of these are to give support to distributed co-operative work, supporting integration in CASE, guiding the process of application development, incorporating reusable specification components etc. Huang [Huang98] has also suggested some possible features for the next generation of CASE tools like process modelling, cross-platform portability, learning, standardisation, and access through the Internet.

Traditionally, each method came with its own CASE tool. Application engineers were expected to select the method they wished to follow and used the associated CASE tool. If their application required even minor modifications in the methods/tools they had selected then it was not possible to make these changes. Thus, CASE technology was basically resistant to change.

To sum up, tool support has been lacking in two main directions :
(a) Providing process support
(b) Adapting to the needs of specific systems.

However, as brought out in this paper, requirements engineering lays emphasis on both process and change management. It is evident that a new approach to computer based tools is needed to support requirements engineering.

Figure 8 presents an architecture for process oriented RE support inspired from [Jarke94]. The architecture is *repository based*. The repository extends the one advocated in Information Resource Dictionary Framework Standard [IRDS90]. Similarly to the IRDS repository, it consists of three levels. However, whereas the IRDS deals with levels of *product* description, the repository deals with levels of *product and process* descriptions [Brinkkemper90].

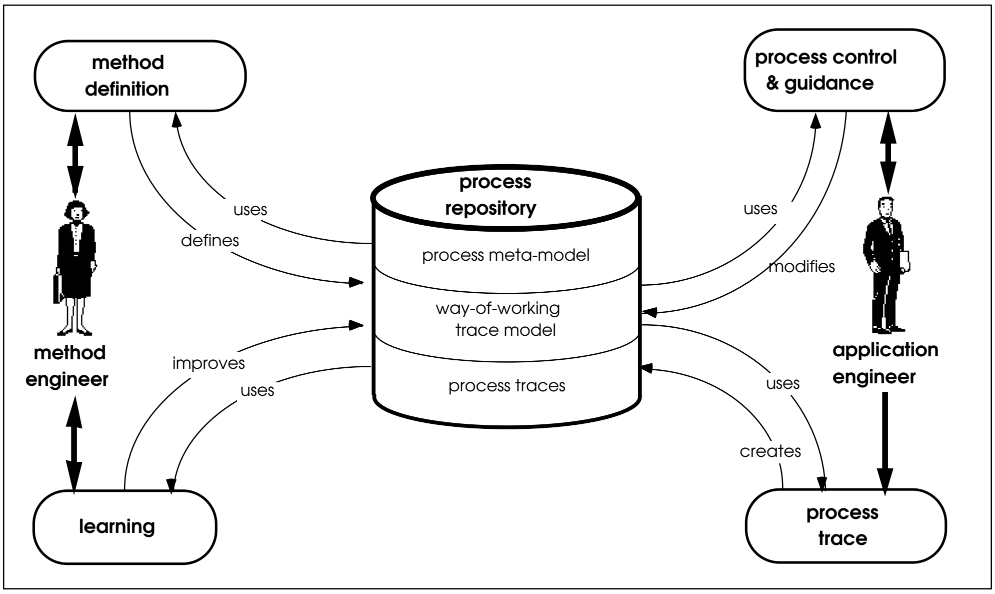

Figure 8 : Repository based and process-integrated environment support

The environment is composed of two sub-environments, the *application engineering environment* in which the process is guided, executed, and traced, and the *method engineering environment* in which the process is defined and improved. These two environments use the *process repository* which contains the information necessary to provide the intended functionality.

Thus, it can be seen, that the architecture of Figure 8 provides process support to application engineers and, additionally, establishes a link between *application* and *method engineering* through the repository.

Tool support for requirements engineering is clearly a complex task. From the point of view of *application engineering*, it involves a number of different problems such as *guidance*, *tracing, repository structuring and management*, *enactment mechanisms*, *efficient interpretation/execution* of process modelling languages, *configuration management*, *view integration*, and *co-operative development*. The application engineering environment needs to support an exploratory process in such a way as to automate all routine, repetitive tasks so that attention can be devoted to exploring alternatives in elaborating the usage world. Additionally, it must help in bringing together the different stakeholders so that the final requirements can be arrived at in a collaborative manner. Finally, the environment must aid in the visualisation of the future system. Only a few of these functions have been implemented in prototypes such as PRO-ART [Pohl94b] to support pre-traceability, MENTOR [SiSaid96] which is a generic tool in the sense that it can function both as a method engineering tool and as a CASE tool depending upon the nature of the process model, PRIME-CREWS [Haumer98] which offers a whiteboard editor for creating fine-grained traceability between goal models and multimedia artefacts, SAVRE [Sutcliffe98] which guides in the generation of requirements to deal with system exceptions and the CREWS-L'Ecritoire [Tawbi98] RE environment which has been built to provide guidance features as described in the goal-scenario coupling approach dealt with above.

Over the last decade, *method engineering* has arisen as a separate field of study in itself. A number of CASE shells have been defined which consist of two parts, the Computer Assisted Method Engineering (CAME) and CASE generator part [Martiin94]. A method is engineered by appropriate instantiation of the meta-model used in the CAME part. The CASE part uses this method to define the application engineering environment. Early CAME parts were organised around product meta-models and MetaEdit [Kelly96] is an example of this. Efforts have been made to include process aspects in CAME using activity meta-models. When these are instantiated then the
activities, tasks etc. performed by methods are also defined for the CAME tool and Meta-Edit + [Kelly96], is an example of that. More recently, a meta-model has been defined in terms of method components. In [Harmsen93] there are two kinds of components called fragments, namely product and process fragments whereas chunks in [Rolland96] realise a tight process and product coupling. Method engineering is seen as a process of assembling together a method from its different fragments [Harmsen93], [Plihon98], [Ralyte99]. From the point of view of *method engineering*, tools must support the

- selection of situated methods i.e. methods meeting some contingency factors
- creation of new methods rapidly when a completely new situation occurs,
- modification of existing methods to handle minor changes in methods, and
- assembly of situated methods from off the shelf method components to gain from past experience.

Finally, requirements engineering tools must support a *feedback mechanism* that allows application engineers to influence method definition and tool construction.

5. Conclusion

.

The thrust areas in requirements engineering are :

- Embedding of systems in their larger usage context, and
- Change management

The former is made possible by stepping back from merely anticipating the functionality that a system must provide (as done in conceptual modelling) to the determination of this functionality in a systematic manner. This is done by identifying the aims and objectives of different stakeholders and the activities they carry out to meet these objectives. This stakeholder driven approach leads to better change management capabilities than found in conceptual modelling. This is because the RE product keeps track of the conceptual link between objectives, activities and system requirements.

As a consequence of the shift to objectives and activities performed to meet them, almost all aspects of information systems engineering get affected. There is a new range of product models to directly represent these. The engineering processes involved are less prescriptive thereby supporting higher creativity and place an emphasis on learning from past experience. The supporting tools are directed on one hand, towards automation of routine tasks and towards providing direct guidance and support for discovering objectives and activities and on the other hand, towards process tracing in order to benefit from past experience. Guidance and tracing support needs to be provided in as transparent a way as possible, it should be effective without being obtrusive. An environment is needed that provides a judicious mix of automated and semi-automated tools that perform routine, humdrum tasks while leaving important decision-making tasks to be done by the requirements engineer.

4. References

[Alfort90] : Alfort M.W., 'Software Requirements Engineering Methodology (SREM) at the Age of Eleven : Requirements Driven design', in Modern Software Engineering, Van Nostrand Reinhold, 1990.

[Anton 96] : A.I. Anton, *Goal based requirements analysis*. Proceedings of the 2nd International Conference on Requirements Engineering ICRE'96, pp. 136-144, 1996.

[Bandinelli93] S. Bandinelli, A. Fugetta, S. Grigoli, *Process Modelling in the large with SLANG*. Proceedings of the 2nd International Conference on « Software Process », Berlin, GERMANY, 1993, pp 75-93.

[Bari90] : M. Bari. And C. Rolland "A Methodology for the Object-Oriented Design", in: Prakash, N. (ed.), Data Management, Current Trends, Tata-McGraw-Hill, New-Delhi, Proc. of the Intern. Conf. on Management of Data, pp 244-259, 1990.

[Belkhatir94] N. Belkhatir, W. L. Melo, *Supporting Software Development Processes in Adele2.* The Computer Journal, vol 37, N°7, 1994, pp 621-628.

[Borgida84] : Borgida, A., Mylopoulos, J., Wong, H.K.T., "Generalization/ specialization as a basis for software specification". In Brodie/Mylopoulos/Schmidt (eds.): On Conceptual Modeling, Springer Verlag 1984, 87-114.

[Brinkkemper90] : S. Brinkkemper, "Formalisation of Information Systems Modelling ", Ph.D. thesis, University of Nijmegen, Thesis Publishers, Amsterdam, 1990.

[Brodie.82] : M.L. Brodie, E. Silva, "Active and Passive Component Modelling: ACM/PCM", in IFIP WG8.1 Working Conference on Information System Design Methodologies : a Comparative Review, North-Holland, 1982.

[Brunet90] : J. Brunet, Cauvet C., Lasoudris L., 'Why using Events in a High-level Specification', in Proceedings of the International Conference on 'Entity-Relationship Modelling', ER90, Lausanne, Switzerland, 1990.

[Bubenko77] Bubenko J. A., "The Temporal dimension in information modelling", Architecture and Models in Data Base Management Systems, North Holland (pub), the Netherlands, pp.93-118, 1977.

[Bubenko92] : Bubenko J.A., and Wangler B., Research Directions in Conceptual Specification development in Conceptual Modelling, Databases, and CASE, Loucopoulos P., and Zicari R. (eds.), 389 – 412, Wiley

[Bubenko94a] : J. Bubenko Jr., Marite Kirikova. 'Worlds' in Requirements Acquisition an Modelling ", 4th European - Japanese Seminar on Information Modelling and Knowledge Bases, Kista, Sweden, Kangassalo and Wangler (Eds.), IOS (pub),1994.

[Bubenko94b] : Bubenko J., Rolland C., Loucopoulos P., De Antonnellis V., "Facilitating Fuzzy to Formal Requirements Modelling", Proc. Int. Conf. on Requirements Engineering (ICRE), Colorado Springs, USA, 1994.

[Caroll95] : J. M. Caroll, "*The Scenario Perspective on System Development*", in Scenario-Based Design: Envisioning Work and Technology in System Development, Ed J.M. Carroll, 1995.

[Chandrasekaran92] : Chandrasekaran B., Keuneke A., Tanner M., "Explanation in knowledge systems : The roles of the Task Structures and Domain Functional Models", in Proc. of Workshop on Task Explanation, Samos, Greece, University of Aegean, 1992.

[Chung96] : Chung L., Nixon B., Yu E., 'dealing with Change : an Approach Using Non-Functional Requirements', The Requirements Engineering Journal, Vol 1, Nb 4, pp 238-260, 1996.

[Clifford83] : Clifford J., Warren D., "Formal semantics for time in databases", ACM Trans. on Database Systems, Vol 6, N 2, 1983 .

[Cockburn95] A. Cockburn, *Structuring use cases with goals*. Technical report. Human and Technology, 7691 Dell Rd, Salt Lake City, UT 84121, HaT.TR.95.1, http://members.aol.com/acocburn/papers/usecases.htm, 1995.

[Cugola95] G. Cugola, E. Di Nitto, C. Ghezzi, and M. Mantione, *How to deal with deviations during process model enactment*. Proceedings of 17th International Conference on « Software Engineering » (ICSE17), Seattle, Washington, USA, April 1995.

[Cugola96] G. Cugola, E. Di Nitto, A. Fuggetta, and C. Ghezzi, *A Framework for Formalizing Inconsistencies and Deviations in Human-Centered Systems*. ACM Transactions on « Software Engineering and Methodology » (TOSEM), vol. 5, num. 3, July 1996.

[Cugola98] G. Cugola, *Inconsistencies and Deviations in Process Support Systems*. Ph.D. Thesis Politecnico di Milano, February 1998.

[Curtis88] : Curtis B., Krasner H. & Iscoe N., "A Field Study of the Software Design Process for Large Systems"', Comm. ACM Vol 31,No11, 1268-1287, 1988.

[Dano97] B. Dano, H. Briand, F. Barbier, *A use case driven requirements engineering process*. Third IEEE International Symposium On Requirements Engineering RE'97, Antapolis, Maryland, IEEE Computer Society Press, 1997.

[Dardenne91] : Dardenne, A., Fickas, S., van Lamsweerde, A., "Goal-directed concept acquisition in requirements elicitation", Proc. 6th IEEE Workshop System Specification and Design0 , Como, Italy, 1991, 14-21.

[Dardenne93] :Dardenne, A., A. v. Laamsweerde and S. Fickas, ”Goal Directed Requirements Acquisition”, Science of Computer Programming, 20(1-2), pp3-50, 1993).

[Davies93] : Davies A.M. : "Object functions and states", Prentice Hall, Englewood Cliffs, NJ, 1993.

[Davis90] : Davis A.M. ,'The Analysis and Specification of Systems and Software Requirements', in systems and Software Requirements Engineering, Tayer R.H. and Dorfmann M. (eds), IEEE Computer Society Press, pp119-134, Tutorial, 1990.

[Desfray94] : Desfray, P., "Object Engineering, the Fourth Dimension", Addison-Wesley/Masson, 1994.

[DoD-2667A] : Military Standard : Defense System Software Development, Department of Defense, 1988.

[Dowson94] : M. Dowson, C. Fernstrom, "Towards requirements for Enactement Mechanisms", Proc. of the th European Workshop on Software Process Technology, 1994.

[Dubois89] : E. Dubois, J. Hagelstein, A. Rifaut, "Formal Requirements Engineering with ERAE", Philips Journal of Research, Vol 43, No 4, 1989.

[Dubois94] : Eric Dubois, Philippe Du Bois, Frédéric Dubru, Michaël Petit, “ Agent-Oriented Requirements Engineering: A case Study using the ALBERT Language ”, Proc. of the Fourth International Working Conference on Dynamic Modelling and Information System - DYNMOD-IV, A. Verbraeck, H.G. Sol, and P.W.G. Bots (editors), Noordwijkerhoud, The Netherlands, september 28-30, 1994.

[ELEKTRA97] ELEKTRA consortium, *Esprit Program 7.1, Technologies for business processes, best business practice pilots. Elektra : Electrical Enterprise Knowledge For Transforming Applications.* (N°22927) http ://www.singular.gr/elektra, January 1997 to June 1999.

[ELEKTRA98] : ELEKTRA consortium, *DEMETRA : System Design Specification for PPC*, ELEKTRA deliverable, March 1998.

[Erickson95] : T. Erickson, "Notes on Design Practice: Stories and Prototypes as Catalysts for Communication", in Scenario-Based Design: Envisioning Work and Technology in System Development, Ed J.M. Carroll, 1995.

[Feather92] : Feather M.S, Fickas S. : "Coping with requirements freedoms", Workshop notes, Int. Workshop on development of Intelligent Information Systems, Niagara, Ontario, 1992.

[Feiler93] Feiler P.H. and Humphrey W.S., 'Software Process Development and Enactment: Concepts and Definitions', Proceedings of the. Second Intl. Conf. On Software Process, 1993.

[Finkelstein90] : Finkelstein A., Kramer J., Goedicke M., "ViewPoint Oriented Software Development", Proc. Conf "Le Génie Logiciel et ses Applications", Toulouse, p 337-351, 1990.

[Finkelstein94] : A. Finkelstein, J. Kramer, B. Nuseibeh (eds), "Software Process Modelling and Technology", John Wiley (pub), 1994.

[Firesmith94] : D. G. Firesmith, "Modelling the dynamic Behaviour of Systems, Mechanisms, and Classes with Scenarios", In Software DevCon '94, pages 73-82. SIGS Publications, NY, 1994.

[Flynn90] : Flynn R.F., Dorfmann D., 4The Automated Requirements Traceability System (ARTS) : an experience of eight years', in Systems and Software Requirements, Tayer R.H. and Dorfmann M. (eds), IEEE Computer Society Press, Tutorial, PP423-428, 1990

[Franckson91] : M. Franckson, C. Peugeot, "Specification of the Object and Process Modeling Language ", ESF Report n° D122-OPML-1. 0, 1991.

[Fugini90] : M.G. Fugini, B. Pernici I, "RECAST: A Tool for Reusing Requirements", Conf. CAISE'90, Stockholm, May, 1990.

[Glinz95] : M. Glinz, "An integrated formal Model of Scenarios based on Statecharts", Lecture Notes in Computer Science '95 , pages 254-271, 1995.

[Gotel94] : Gotel O., Finkelstein A., 'An Analysis of the Requirements Traceability Problem', in Proceedings of the First Ont. Conf. On Requirements Engineering, Colorado Springs, CP, pp94-102, IEEE Computer Society Press, 1994.

[Gotel94b] : Gotel O., Finkelstein A., 'Modelling the Contribution Structure Underlying Requirements', in Proc. of the first Int. Workshop on Requirements Engineering : Foundation of Software Quality', Utrech, The Netherlands, 1994.

[Gougen93] : Gougen J.A., "Social issues in Requirements Engineering", In Proceedings of IEEE Symposium on Requirements Engineering, IEEE Computer Society Press 194-198.

[Greuber92] : Greuber T., "Ontolingua : A mechanism to support portable ontologies", version 3.9. Technical report, Knowledge systems laboratory, Stanford University, CA, 1992.

[Harandi91] : Harandi S.D.P, Lee M.Y., "Acquiring software design schema : A machine learning perspective", Proc. 6th Knowledge Based Software Engineering Conference, IEEE Computer Society press, 188-197, 1991.

[Harel87] : D. Harel, "Statecharts: a Visual Formalism for Complex Systems", Sci. Computer Program, 8, pp. 231-274, 1987.

[Harker93] : Harker S.D.P., Eason K.D. & Dobson J.E., "The Change and Evolution of Requirements as a Challenge to the Practice of Software Engineering", IEEE Symposium on Requirements Engineering, RE'93, San Diego, CA, Jan. 4-6, 1993, 266-272.

[Harmsen93] : F. Harmsen, S. Brinkkemper, "Computer Aided Method Engineering based on existing Meta-Case technology," Proc. of the fourth Workshop on the Next Generation of CASE tools, (NGCT93), Paris, France, 1993.

[Haumer98] : Peter Haumer, Klaus Pohl, Klaus Weidenhaupt, *Requirements Elicitation and Validation with Real World Scenes*, to appear in IEEE Transactions on Software Engineering, Vol. 24, No. 12, Special Issue on Scenario Management, December 1998.

[Holbrook90] : C. H. Holbrook, *A scenario - based methodology for conducting requirements elicitation*. ACM SIGSOFT, Software Engineering Notes Vol. 15, N° 1, pp. 95-104. January 1990.

[Hsia94] : P. Hsia, J. Samuel, J. Gao, D. Kung, Y. Toyoshima, C. Chen, "*Formal Approach to Scenario Analysis*", IEEE Software, pp. 33-41, 1994.

[Huang98] Huang R., Making Active CASE TOOLs – Towards the Next Generation of CASE Tools, Software Engineering Notes, 23, 1, 47 - 50, 1998.

[Humphrey89] : Humphrey W.S., Kellner M.I.; "Software Process Modeling: Principles of Entity Process Models", Proc. 11th Int. Conf. on Software Engineering, 1989.

[IEEE-830] : Guide to Software Requirements Specification, ANSI/IEEE std 830, 1984.

[IRDS90] : Information Technology - Information Resource Dictionary System (IRDS) - Framework, ISO/IEC International Standard.

[Jacherri92] L. Jacherri, J. O. Larseon, R. Conradi, *Sotware Process Modelling and Evolution in EPOS*. Proceedings of the 4th International Conference on « Software Engineering and Knowledge Engineering » (SEKE'92), pp574-589, Capri, ITALY, 1992.

[Jackson 93] : Jackson M., Zave P., 'Domain descriptions', IEEE symposium on requirements Engineering, IEEE Computer Society press, 56-64, 1993

[Jackson94] : Jackson M., Problems, Methods and Specialisation. Contributions to the Special Issue of the SE Journal on 'Software Engineering in the Year 2001', 1994.

[Jacobson et al., 1992] I. Jacobson, M. Christerson, P. Jonsson, G. Oevergaard, "Object Oriented Software Engineering: a Use Case Driven Approach", Addison-Wesley, 1992

[Jacobson95] : I. Jacobson, *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 and Sons, 309-336, 1995.

[Jarke93] : Jarke, M., Pohl, K., "Establishing visions in context: towards a model of requirements processes". Proc. 12th Intl. Conf. Information Systems, Orlando, Fl, 1993.

[Jarke94] : Jarke M., Pohl, K.,Rolland C ;, Schmitt J.R., 'Experience-based Method Evaluation and Improvement : a Process Modelling Approach', in Proceedings of the Intl. IFIP Conf. On 'Methods and Associated Tools for the Information Systems Life Cycle', pp 1-28, Maastrich, The Netherlands, 1994.

[Johnson 95] J. Johnson, *Chaos : the Dollar Drain of IT project Failures*. Application Development Trends, pp.41-47, January 1995.

[Kaindl93] : Kaindl H., 'The Missing Link in Requirements Engineering', ACM SIGSOFT Software Engineering Notes, 19(2) :30-39, 1993.

[Kardasis98] : P. Kardasis, P. Loucopoulos, "Aligning Legacy Information Systems to Business Processes". Proceedings of the 10^{th} Conference on Advanced Information Systems Engineering, CAiSE'98. Pisa, Italy, 8-12 June, 1998.

[Kelly96] : kelly S., Lyyttinen K., Rossi M., 'Meta-Edit+ : a fully configurable, multi-user and multi-tool CASE and CAME environment', Proceedings of the CAISE'96 Conference, Springer Verlag, 1996.

[Kirikova94] : Kirikova M., Bubenko J., 'Enterprise modelling : improving the quality of requirements specifications', Information Research Seminar, IRIS 17, Olou, Finland, August 1994.

[Kyng95] : M. Kyng, "Creating Contexts for Design", In John M. Carroll, editor, Scenario-Based Design: Envisioning Work and Technology in System Development, pages 85-107. John Wiley and Sons,1995.

[Lalioti95] : V. Lalioti and B. Theodoulidis, "Use of Scenarios for Validation of Conceptual Specification", Proceedings of the Sixth Workshop on the Next Generation of CASE Tools, Jyvaskyla, Finland, June 1995.

[Lehman87] : M.M. Lehman ; "Process Models, Process Programs, Programming Support "; Proc. 9th Int. Conf. on Software Engineering ; 1987

[Leite97] J.C.S. do Prado Leite, G. Rossi, F. Balaguer, A. Maiorana, G. Kaplan, G. Hadad and A. Oliveros, *Enhancing a requirements baseline with scenarios*. In Third IEEE International Symposium On Requirements Engineering RE'97, Antapolis, Maryland, IEEE Computer Society Press, pp. 44-53, 1997.

[Loucopoulos94] : P. Loucopoulos, *The F^3 (From Fuzzy to Formal) view on requirements engineering.* Ingénierie des Systèmes d'Information, Vol. 2, N° 6, pp. 639-655, 1994.

[Loucopoulos98] : Loucopoulos, P., Kavakli, V., Prekas, N., Dimitromanolaki, I. Yilmazturk,C., Rolland, C., Grosz, G., Nurcan, S., Beis, D., and Vgontzas, G., 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-CSC'98), Athens, Greece, 1998 , pp. 352-357.

[Lubars93] : Lubars M., Potts C., Richer C., "A review of the state of the practice in requirements modeling",. Proc. IEEE Symp. Requirements Engineering, San Diego 1993.

[Macaulay93] : Macaulay L., 'Requirements capture as a cooperative activity', IEEE symposium on Requirements Engineering, IEEE Computer Society Press, 174-181, 1993.

[Maiden94a] : Maiden N.A.M., Sutcliffe A.G., "Computational mechanisms for reuse of domain knowledge during requirements engineering", ICSE-Workshop on research issues in the intersection between software engineering and artificial intelligence, Sorrento, Italy, May 1994.

[Maiden94b] : Maiden N.A.M., Rugg G., "Knowledge Acquisition Techniques for Requirements Engineering", Proc. Workshop on Requirements Elicitation for System Specification, Keele UK, 12-14 July, 1994.

[Martin92] : Martin J., Odell J., 'Object Oriented Analysis &Design', Prentice Hall, Englewoods Cliffs, NJ07632, 1992.

[Marttiin94] : P. Marttiin, "Methodology Engineering in CASE shells : Design Issue and current Practice ", PhD thesis, Computer science and information systems reports, Technical report TR-4, 1994.

[McGraw97] : Karen Mc Graw, Karan Harbison, *User Centered Requirements, The Scenario-Based Engineering Process*. Lawrence Erlbaum Associates Publishers, 1997.

[Mylopoulos92] : Mylopoulos J., Chung, L., Nixon, B., "Representing and using non functional requirements: a process-oriented approach", IEEE Trans. Software Eng. Vol 18, N 6, 1992.

[Nardi92] : B. A. Nardi, "The Use of Scenarios in Design", SIGCHI Bulletin, 24(4).

[Norman92] : Norman R. and Forte G.(eds.), CASE in the 90's, Special section of the CACM, 1992.

[Nurcan99a] : S. Nurcan, J. Barrios, G. Grosz, C. Rolland, 'Change process Modelling using the EKD-Change management Method', in proceedings of the European Conference on information Systems, ECIS'99, 1999.

[Nurcan99b] : Selmin Nurcan, Colette Rolland, 'Using EKD-CMM Electronic Guide Book for Managing Change in Organisations', Submitted to European-Japanese Conference 1999.

[Olle88] : Olle, T.W., Hagelstein, J., MacDonald, I.G., Rolland, C., Sol, H.G., Van Assche, F.J.M., Verrijn-Stuart, A.A., "Information Systems Methodologies: A Framework for Understanding", Addison-Wesley, 1988.

[Osterweil87] : L. Osterweil, "Software processes are software too"; Proc. 9th Int. Conference on Software Engineering, IEEE Computer Society, Washington, DC, 1987, pp2-13, 1987.

[Pietro-Diaz91] : Pietro-Diaz R., "Implementing faceted classification for software reuse", Communications of ACM, Vol 34, No 5, 88-97, 1991.

[Plihon98] : V. Plihon, J. Ralyté, A. Benjamen, N.A.M. Maiden, A. Sutcliffe, E. Dubois, P. Heymans, *A reuse-oriented approach for the construction of scenario based methods*. Proceedings of the International Software Process Association's 5th International Conference on Software Process (ICSP'98), Chicago, Illinois, USA, 14-17 June 1998.

[Pohl96] : K. Pohl,. 'Process Centered Requirements Engineering', J. Wiley and Sons Ltd., 1996.

[Pohl97] : K. Pohl, P. Haumer, *Modelling contextual information about scenarios*. Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona, pp.187-204, June 1997.

[Pohl94a] : Pohl, K., "The Three Dimensions of Requirements Engineering: a framework and its application", Information Systems Vol 19, N 3, pp 243-258, 1994.

[Pohl94b] : Pohl K., Doemges R. Jarke M.; "PRO-ART: An environment for Enabling Requirements Traceability" Technical Report, Informatik V, RWTH-Aachen, 1994.

[Potts 97] : C. Potts, *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, June 1997.

[Potts89] : Potts C., "A Generic Model for Representing Design Methods", Proc. 11th Int. Conf. on Software Engineering, 1989.

[Potts94] : C. Potts, K. Takahashi, A.I. Anton, *Inquiry-based requirements analysis*. In IEEE Software 11(2), pp. 21-32, 1994.

[Prakash97] : Prakash N., 'Towards A Formal Definition of Methods', the Requirements Engineering Journal, 23 – 50, Springer Verlag, 1997.

[Prat97] : N. Prat, *Goal formalisation and classification for requirements engineering*. Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona, pp. 145-156, June 1997.

[Ralyte99] : Ralyte J., Rolland C., Plihon V., '' , accepted for publication in the Proceedings of the 11th Intl. Conf. On Advanced Information Systems Engineering (CAISE'99), Springer Verlag, 1999.

[Ramesh93a] : Ramesh B., 'A Model of Requirements Traceability for Systems Development', Technical Report, naval Postgraduate School, Monterey, CA, 1993.

[Ramesh93b] : Ramesh, B., Edwards, M., "Issues in the Development of a Requirements Traceability model", Proc. IEEE Symp. on Requirements Engineering, IEEE Computer Society Press, San Diego, Ca, 1993.

[Ramesh95] : Ramesh, B., Powers T., Stubbs C. and Edwards, M., ' implementing Requirements Traceability : A Case Study', in Proceedings of the 2nd Symposium on Requirements Engineering (RE'95), pp89-95, York, UK, 1995.

[Rawsthorne96] : D. A. Rawsthorne, "Capturing functional Requirements through Object Interactions", in Proceedings of ICRE '96, pages 60-67. IEEE, 1996.

[Reubenstein91] : Reubenstein H.B., Waters R.C., "The Requirements Apprentice : Automated assistance for the requirements acquisition", IEEE Transactions on Software Engineering, Vol 18, No 3, 226-240, 1991.

[Rolland79] : Rolland C., Leifert S., Richard C, "Tools for information systems design and management", Proc. 5th Int. Conf. on VLDB, 1979.

[Rolland82] : C. Rolland, C. Richard, "The REMORA Methodology for Information Systems Design and Management", in IFIP WG8.1 Working Conference on Information System Design Methodologies : a Comparative Review, North-Holland, 1982.

[Rolland91] : Rolland C., Cauvet C., "ALECSI : An Expert System for Requirements Engineering", Proc. 3th Int. Conf. on Advanced Information Systems Engineering (CAISE'91), Springer Verlag (Pub.).

[Rolland93] : Rolland C., "Modelling the Requirements Engineering Process", Information Modelling and Knowledge Bases, IOS Press.

[Rolland94] : Rolland C., Prakash N., "A Contextual Approach for the Requirements Engineering Process", Proc. Int. IEEE Conf. on Software Engineering and Knowledge Engineering (SEKE94), Riga, 1994.

[Rolland95] : Rolland C., Souveyet C., Moreno M, "An Approach for Defining Ways-of-Working ", Information Systems Journal, Vol. 20, No 4, pp337-359 ,1995.

[Rolland97a] C. Rolland, C. Ben Achour, *Guiding the construction of textual use case specifications*. Data & Knowledge Engineering Journal Vol. 25 N° 1, pp. 125-160, (ed. P. Chen, R.P. van de Riet) North Holland, Elsevier Science Publishers. March 1997.

[Rolland97b] : C. Rolland, S. Nurcan, G. Grosz "A way of working for change processes" in International Research Symposium '97 - Effective Organisations, 4-5 Sep 1997, Dorset, UK, p.201-204

[Rolland98a] : C. Rolland, C. Souveyet, C. Ben Achour. "*Guiding Goal Modelling using Scenarios*", IEEE Transactions on Software Engineering, Special Issue on Scenario Management, Vol. 24, No. 12, 1055- 1071, Dec. 1998.

[Rolland98b] : C. Rolland, P. Loucopoulos, V. Kavakli, S.Nurcan "Intention based modelling of organisational change", under publication

[Rolland98c] : C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N.A.M. Maiden, M. Jarke, P. Haumer, K. Pohl, Dubois, P. Heymans, "A Proposal for a Scenario Classification Framework". Requirements Engineering Journal, Vol; 3, No. 1, pp. 23-47, 1998.

[Rolland 99] : Rolland C., Prakash N., Benjamen A., ‘A Multi-model View of Process Modelling’, The Requirements Engineering Journal, (under communication).

[Royce70] : Royce W. W., "Managing the Development of Large Software Systems", Proc. IEEE WESCON 08.

[Rubin92] : Rubin K.S., Golberg A., *Object Behavior Analysis*, Communications of the ACM, 35(9), Sept 1992, pp 48-62.

[Rumbaugh91] : J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *‘‘Object-oriented modelling and design’’*. Prentice Hall, 1991.

[Russo95] Russo, *The use and adaptation of system development methodologies*, Proceedings, 1995, International Resources Management. Association Conference, Atlanta, USA, 1995

[Shaw91] : Shaw M., "Heterogeneous design idioms for software architecture", 6th International Workshop on Software Engineering and Design, IEEE Computer Society Press, 158-165, 1991.

[SISaid96] : Si-Said S., Rolland C., Grosz G., "MENTOR : A Computer Aided Requirements Engineering Environment", in Proc 8th Int. Conf. on Advanced Information Systems Engineering (CAISE'96), Springer Verlag, 1996 .

[Some96] : S. Some, R. Dssouli, J. Vaucher, "Toward an Automation of Requirements Engineering using Scenarios", Journal of Computing and Information, Special issue: ICCI'96, 8th International Conference of Computing and Information, Waterloo, Canada,2(1) pp 1110-1132, 1996.

[Standish 95] : The Standish Group, *Chaos*. Standish Group Internal Report, http://www.standishgroup.com/chaos.html, 1995.

[Sutcliffe94] : Sutcliffe A.G., Maiden N.A.M., "A theory of domain knowledge for requirements engineering", Nature Report Deliverable D-D-2 August 1994.

[Sutcliffe98] : Sutcliffe A.G., Maiden N.A.M., Minocha S , Manuel D. : ‘Supporting scenario-based requirements engineering’, IEEE transactions on Software Engineering, special issue on ‘Scenario Management’, Vol 24, Nb 12, pp 1072-1088, Dec.1998.

[Tawbi98] : M. Tawbi, C. Souveyet, C. Rolland, *L'ECRITOIRE a tool to support a goal-scenario based approach to requirements engineering*, Submitted to Information and Software Technology journal, Editor : Martin Shepperd, Publishers : Elsevier Science B.V, 1998

[van Lamsweerde95] : A. Van Lamsweerde, R. Dairmont, P. Massonet, *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, IEEE, 1995, pp 194 –204.

[van Lamsweerde98] : Van Lamweerde A., Willemet L., ‘Inferring Declarative Requirements Specification from Operational Scenarios’, IEEE Transactions on Software Engineering, Special Issue on Scenario Management, Vol. 24, No. 12, 1089- 1114, Dec. 1998.

[Weidenhaupt98] : K. Weidenhaupt, K. Pohl, M. Jarke, P. Haumer, *Scenario usage in system development : a report on current practice*. IEEE Software, March 1998.

[Wijers91] Wijers G.M.,' Modeling Support in Information Systems development', PhD. Thesis, Thesis Publishers, Ansterdam, 1991.

[Wilenga93] : Wilenga B., Van de Velde, Schreiber G., Akkermans H., "Expertise model definition document"; KADS project document KADS-II/M2/UvA, University of Amsterdam, 1993.

[Willars92] : Willars H., 'Amplification of business cognition through modelling techniques' 11th IEA Congress, Paris, 1992

[Wood94] : D.P. Wood, M. G. Christel, S. M. Stevens, "A Multimedia Approach to Requirements Capture and Modelling", Proceedings. ICRE'94, Colorado Springs, 1994.

[Wright92] : Wright P., "What's in a Scenario", SIGCHI Bulletin, Volume 24, Number 4, October 1992.

[Wright91] : Wright S ;, 'Requirements Traceability : What ? Why ? and How ?', in Proceedings. of the Colloquium on Tools and Techniques for Maintaining Traceability during Design, IEEE Professional Group C1 (SE), London, UK, 1991.

[Wynekoop93] J. D. Wynekoop, N. L. Russo, "System Development methodologies: unanswered questions and the research-practice gap,", Proc. of 14th ICIS (eds. J. I. DeGross, R. P. Bostrom, D. Robey), Orlando, USA, 1993, pp. 181-190.

[Young87] M. R. Young, P. B. Barnard, "*The Use of Scenarios in Human-Computer Interaction Research: Turbocharging the Tortoise of Cumulative Science*", CHI + GI 87 Human Factors in Computing Systems and Graphics Interface, Toronto, 1987.

[Yourdon 89] : Yourdon E.E, 'Modern structured analysis', Prentice Hall, 1989.

[Yu94a] Eric SK Yu, John Mylopoulos. " Understanding 'Why' in Software Process, Modelling, Analysis, and Design ". In Proc. of the 16th International Conference on Software Engineering _ ICSE'94, Sorrento (Italy), May 16-21, 1994. IEEE & ACM.

[Yu94b] Eric SK Yu, John Mylopoulos. " From ER to AR_ modelling strategic Actor Relationships for Business Process Reengineering ". In Proc. of the 13th International Conference on the Entity-Relationship Approach _ ER'94, Manchester (UK), December 13-16, 1994.

[Yu94c] Eric SK Yu, John Mylopoulos. " 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 (WITS'94), Vancouver, B. C., Canada, December 17-18, 1994.