UFCE4S-10-3: Requirements Engineering
Faculty of Computing, Engineering and Mathematical Sciences
University of the West of England

Goal-Oriented Approaches to Requirements Engineering

by

Stewart Green

List of Figures

List of Tables

Chapter 1

Deriving requirements from goals

1.1 Introduction

Each of the socio-technical approaches to requirements engineering described in Chapter Three—Checkland's Information Systems Development [WBPB95], Holtzblatt's and Beyer's Contextual Inquiry [HB96, HB98], and Eason's socio-technical approach [Eas88]—featured, to a greater or lesser extent, the use of current goals and knowledge of current organisational activities to derive requirements for computer-based systems. However, if the goal-oriented and the related process-oriented approaches were fully articulated in these socio-technical approaches, then it is likely that they would become even more effective at tackling some of the problems associated with the traditional approaches, for example at tackling the problem of high-level organisational objectives often being ignored. So the goal-oriented and process-oriented approaches to deriving requirements for computer-based systems need to be fully understood, and this chapter now reviews both. The chapter begins by discussing the rationale for the goal-oriented approach. This is followed by a basic description of this approach. Then existing work on goal-oriented approaches is reviewed, and advantages and problems of the goal-oriented approach are listed. Goals are seen to have a close relationship to processes in that processes are enacted to achieve goals. The chapter ends with a review of process-oriented methods associated with the development of computer-based systems, and, therefore, with the derivation of the requirements for such systems. The reviews highlight the salient features of process-oriented approaches, and surface problems associated with each approach considered. The detailed description of goal-oriented and process-oriented approaches presented in this chapter are used to inform the design of the synthesised approach in the next. And the problems with the goal-oriented and process-oriented approaches encountered here are taken to be problems that need to be addressed by the new, synthesised approach that is presented in Chapter Six.

1.2 Rationale for a goal-oriented approach to requirements engineering

Asking and answering the "why" questions during software development is viewed as important by a number of researchers in the field. For example, McDermid provides a strong justification for this position [McD94]. He first observes that, traditionally, requirements specifications are characterised by the description of a system's functions. Then, after enumerating a number of problems perceived with such specifications, he says "perhaps most significantly, such specifications may violate the 'high level objectives' of the organisation the system is intended to serve" [McD94] [1]. The violation occurs because "the process which leads to such specifications is effectively 'starting at the wrong place', that is with the derived requirements not fundamental ones. By this is meant *looking at the desired properties at a presupposed system boundary instead of starting with business needs or objectives of the embedding system*" [McD94][2] (thesis author's italics). In other words, McDermid is saying that much system development starts with too

[1] Page 25
[2] Page 26

narrow a system boundary; and that this is unhelpful. Instead, it should start by considering the goals of the business or embedding system, in other words the goals of a served system, in Checkland's terms [WBPB95].

McDermid's position is shared by other researchers [LK95a] [DBCS94] [3]. For example, Loucopoulos and Karakostas state that "it is important to establish at the outset the significance that requirements have *in an organisational context*" [LK95a] [4] (thesis author's italics). His justification for this assertion is that "understanding the organisational setting is crucial to developing a more complete understanding of requirements for Information Systems. Information Systems and their formal descriptions exist for some reason—they serve strategic, tactical, and operational objectives of the enterprise" [LK95a] [5].

So, Loucopoulos asserts that the purpose of Information Systems is to serve objectives of an enterprise, and McDermid warns that if account of them is not taken during systems development, then they are liable to be violated. Thus, for both, the starting point of systems development is the determination of the goals of a served system. But how are such goals determined and represented, and, more importantly, how are they used to produce requirements for computer-based systems? The next section attempts to answer these questions.

1.3 The goal-oriented approach to requirements engineering

The most important feature of the goal-oriented approach is that it can be used to derive requirements for computer-based systems from the goals of a served system, for example an organisation or department. In addition, requirements derived in this way are considered to be superior to requirements derived from more orthodox approaches. This is because the computer-based systems implemented from them will, at worst, be unlikely to violate the high-level goals, and, at best, will conform well to them.

In one of its simplest forms, the approach involves decomposing high-level goals to lower-level sub-goals, and then decomposing these in turn, until leaf-goals are reached that express requirements for computer-based systems [LK95a] [6]. This process is illustrated in the following example adapted from [LK95a] [7]. Consider the diagram in figure 1.1. It shows how the high-level goal "Increase Profits" has been successively decomposed through three levels to the leaf-goals "Automate Task XYZ" and "Automate Task PQR". Both these leaf-goals express requirements for computer-based systems. So one high-level goal has produced two possible sets of requirements for computer-based systems that will achieve it. This illustrates an important feature of the goal-oriented approach: it normally produces a number of alternative sets of requirements for computer-based systems.

The preceding paragraph has introduced a number of concepts: goal, sub-goal, leaf-goal, goal decomposition, and alternative sets of requirements.

[3] Pages 93 - 94
[4] Page 2
[5] Page 2
[6] Pages 45 - 47 and 84 - 85
[7] Pages 45 - 47

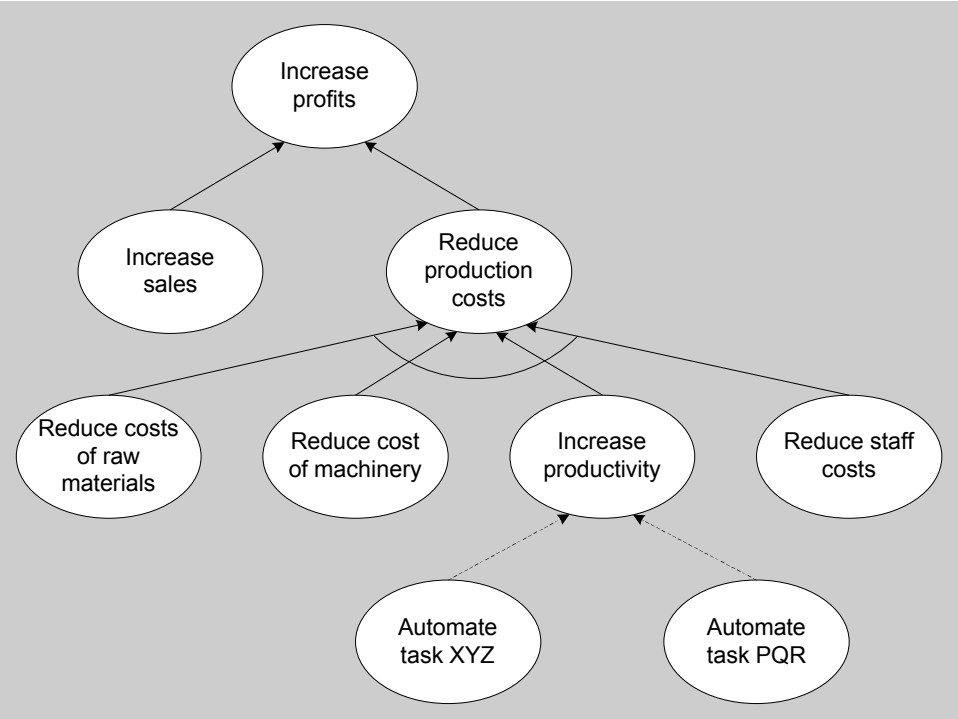

Figure 1.1: Goal decomposition (after [LK95a])

These are now discussed in more detail. Goals are usually defined in terms of states to be reached, maintained, or avoided (see, for example, [McD94][8]). Loucopoulos provides a good definition [LK95a] [9]: "A goal is defined as a defined state of the system. Since a state is described in terms of the values of a number of parameters, a goal can be alternatively defined as a set of desired values for a number of parameters". Loucopoulos provides the following example of a goal: "To make profits of $1M in the next financial year". Here, the two goal parameters are "profits" and a time-frame, and the desired values are "$1M" and "within the next financial year" respectively.

Both in the literature and in the world of work, there are a number of common synonyms for the term goal. These include: objective, aim, end, target, and purpose. Often the terms "end" and "objective" are used to stand for goals that are either vaguely expressed or long-term or to be achieved by multiple actions (see, for example, [McD94][10]. Another term that is often used instead of goal is "constraint". However, constraints and goals are taken here to refer to different phenomena as will be explained in the sequel.

In the goal-oriented approach to deriving requirements for computer-based systems, high-level goals of a served system are first identified. These are then decomposed successively in order to derive requirements for computer-based systems [DvF93, McD94, LK95a] [AP98][11]. The notation that is often used to capture the high-level goals and the result of their decomposi-

[8]Page 28
[9]Page 44
[10]Page 27
[11]Page 158

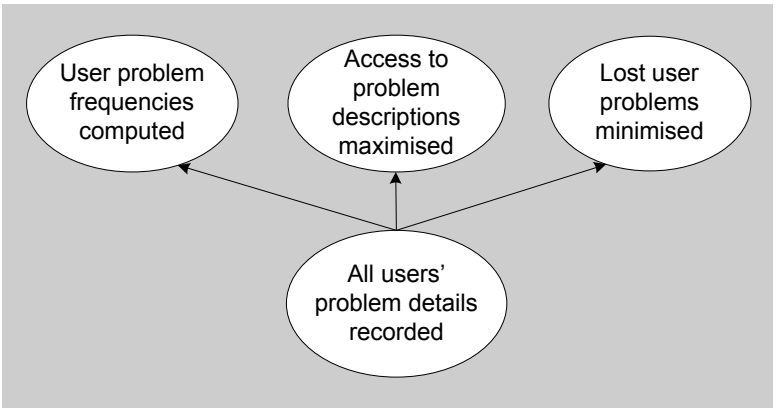

Figure 1.2: Goal lattice - a fragment

tion is derived from Nilsson [Nil71][12]. The original notation has been re-used by a number of researchers employing a goal-oriented approach, but with a variety of adaptations [Chu93, Nix93, DvF93] [AMP94] [13]. The basic idea is described and exemplified in Loucopoulos et al. [LK95a][14]. A goal may be decomposed in two main ways. Either a goal, G1 say, may be decomposed into a set of conjunctive sub-goals, all of which must be attained in order to attain goal G1. Or it may be decomposed into a set of disjunctive sub-goals, at least one of which must be attained in order to attain goal G1. These ideas are illustrated in figure 1.1. The goal "Increase Profits" is attained if either of the goals "Increase Sales" or "Reduce Production Costs" is attained. And the goal "Reduce production Costs" is attained only if all of the sub-goals "Reduce Cost of Raw Material", Reduce Cost of Machinery", "Increase Productivity", and "Reduce Staff Cost" are attained.

Diagrams like the one in figure 1.1 are referred to as goal hierarchies. However, goal lattice would be a more appropriate name [LK95a][15], as, in general, a sub-goal may support the attainment of more than one high-level goal. This is exemplified in the fragment from a goal lattice shown in figure 1.2. It was taken from a goal-hierarchy that was generated in the case study described in Chapter Seven. The case study describes how a goal-oriented approach is used to generate requirements for computer-based systems that are intended to improve the working of a university faculty help-desk. Recording the details of all problems reported to the help-desk by the users helps to attain the three higher level parent goals that are shown. For example, recording problem details helps to minimise the occurrence of "lost" problems, in other words problems reported by users for which the help-desk can find no record.

So far, applications of the goal-oriented approach have been considered that have involved neither conflicting goals nor multiple views of a served system's goals. In practice, the situation is often more complicated. Some sub-goals in a goal-hierarchy usually conflict with others. And different stakeholders frequently have different views about the goals of a served sys-

[12]Pages 22 - 23 and 87 - 90
[13]Pages 100 and 102
[14]Page 45
[15]Page 45

tem. Dardenne et al. [DvF93] [16] assert that two goals conflict with each other if they cannot be achieved together. They illustrate this kind of conflict with an example taken from a library system. One goal is "Maintain Long Borrowing Period". This is defined to mean that a borrower may borrow a book for as long as they need it. Another goal is "Maintain Regular Availability". This is defined to mean that books may be borrowed only for a fixed period. The two goals cannot be achieved together. The first one allows a book to be retained indefinitely. But, should this happen, the second would probably be violated eventually. A slightly weaker form of conflict is proposed by Loucopoulos and Karakostas [LK95a][17]. They assert that "mutually conflicting goals affect negatively the attainment of each other". Thus, when two goals conflict, it is possible that both may be achieved in part. This way of defining conflict is incorporated by Bolton et al. in the GMARC goal-oriented approach [BJT+94][18] in the "is_undermined_by" relationship between goals. In one of their examples of such conflict, the goal "profits made" is undermined by the goal "refunds" allowed, though both can be achieved together to some extent. In the goal-oriented approach to deriving requirements for computer-based systems from goals it is necessary to analyse goal hierarchies for conflicting goals [McD94, LK95a, DvF93]. Conflicting goals need to be identified in order to fully evaluate the effect of alternative solutions, in other words of alternative sets of requirements for computer-based systems. Once conflict has been identified it may be reduced or eliminated by various means. For example, Dardenne et al. propose favouring higher priority goals over lower priority goals [DvF93]. Another way is to encourage stakeholders to negotiate the definitions of conflicting goals until partial, contemporaneous achievement of the goals is enabled.

Conflicting goals complicate the simple picture that was presented earlier. This picture is also complicated in practice by the existence of different views of the main goals and, by implication, the associated goal hierarchies for a served system that are held by the various stakeholders [Kee92][19] [LK95a][20] [DBCS94] [21] [CH98][22] [WKRG99][23] [Jac91]. For example, Loucopoulos states that "the process of eliciting goals is far from straightforward since the process is likely to involve multiple participants who will hold multiple perspectives on a single domain" [LK95a] [24]. Different stakeholders often hold different views about the main goals of a served system. Frequently the goals in different views conflict with one another. In order to progress a development, the differences between views need to be resolved. "The private goals of members of an organisation (its owners, employees, and so on) must all be considered and harmonised" [LK95a] [25]. However, while some workers seem sanguine about the feasibility of this task, and provide detailed mechanisms for combining goal hierarchies [Kee92] [26]

[16] Page 23
[17] Page 46
[18] Page 145
[19] Page 295 - 301
[20] Pages 80,84
[21] Page 93
[22] Pages 83 - 86
[23] Page 142
[24] Page 84
[25] Page 84
[26] Pages 301 - 304

and resolving conflict [DvF93][27], others, for example [Jac91], highlight some of the major difficulties for this endeavour in some common organisational contexts.

1.4 Goal-oriented approaches in the literature

The previous section has described the goal-oriented approach to deriving requirements for computer-based systems. It has presented the main concepts of the approach: goal, goal hierarchy, goal decomposition, and the idea that alternative sets of requirements may satisfy given goals. It has also discussed two inter-related factors that complicate the approach: different stakeholders may hold very different views about what are the main goals of a served system, and conflicting goals may exist both within one view and across different views.

This section reviews research that has been reported in the literature on goal-oriented approaches to deriving requirements for computer-based systems. Research has been categorised according to where its authors have taken the boundary of the served system to lie. Different researchers have defined their served system boundary on either a structural basis or a processual basis. The structural categories are:

- Served system scope defined by the enterprise boundary
- Served system scope defined by the application boundary

The processual category is:

- The focus is on processes

Each subsection below discusses work associated with a particular category and presents representative pieces of research in greater depth.

Served system scope defined by the enterprise boundary

A number of projects have taken the boundary of an enterprise to be the limit of the served system, including the following: GMARC [BJT+94], ORDIT [DBCS94], F3 [BNG94, BRLD94], and [Eas88][28]. Loucopoulos and Karakostas overview a number of approaches to enterprise goal modelling in [LK95a][29], where they also provide a characterisation of enterprise goal modelling and a tutorial example. A representative and relatively complete description of the goal-oriented approach to deriving requirements in the context of enterprises is also provided by Loucopoulos and Kavakli [LK95b]. Their fundamental belief is that "the purpose of building a software system is to be found outside the system itself in the enterprise" [LK95b] [30], and that, therefore, knowledge of the enterprise should be captured in a formal model and used to derive requirements for computer-based systems. Their interpretation of an enterprise model comprises knowledge about goals, roles, actors, processes, and resources in an enterprise. This knowledge is held in

[27] Pages 33 - 35
[28] Pages 83, 85, and 87 - 90
[29] Pages 82 - 87
[30] Page 47

three different views: the technical, the social, and the teleological. The "technical perspective describes a number of different aspects of the business: business processes, flow of information and data across business processes, where resources or organisations are located, where activities take place, etc." [31] The "social viewpoint reasons about policies, structures and work roles; validates and maintains their established order"[32]. The teleological viewpoint is deemed the most important:"Goals are recognised as the primary factors that govern and explain the current and potential enterprise configuration" [33]. The teleological viewpoint is ideally developed in the manner described in the description of the goal oriented-approach given in section 1.3 (see page 6): goals of the stakeholders are determined, along with the constraints of the enterprise; goals are decomposed into more specific sub-goals; this leads to alternative approaches to their satisfaction, each of which must be evaluated; in addition, conflicting goals should be identified and resolved with the stakeholders before the alternatives are evaluated. The paper reviews an application of their approach to an Air Traffic Control (ATC) enterprise. Although an explicit process model for the approach is not given, the one presented below seems to be implicit in the paper:

1. Investigate the enterprise
2. Develop models of the technical and social viewpoints
3. Develop a model of the teleological viewpoint
   (a) Identify goals, constraints, and problems
   (b) Construct goal hierarchies
   (c) Construct a goal graph
   (d) Identify conflicting goals
   (e) Identify goals to automate
   (f) Define automation strategies
   (g) Evaluate automation strategies
   (h) Select best automation strategy
   (i) Define Information System goals
   (j) Define functional requirements to satisfy IS goals
   (k) Define non-functional requirements to satisfy IS goals

Stage-three is illustrated in detail below using fragments of Loucopoulos's and Kavakli's example. In step 3 (a) a requirements engineer should identify goals, constraints, and problems. Some of the high-level goals, constraints, and problems of the ATC enterprise are given below:

- High-level goals
  - Safety (management, customers)
  - Efficiency (management, customers)

[31] Page 52
[32] Pages 52 - 53
[33] Page 53

  - Avoiding boredom (controllers)
  - Reducing demands on controllers (controllers)
  - Minimize risk of aircraft accidents (management, customers)

- Constraints
  - Vertical separation between aircraft cannot be less than 2000 feet
  - Horizontal separation between aircraft cannot be less than 5 miles

- Problems
  - Reduce delays to aircraft
  - Manage problems and emergencies

In 3 (b) the requirements engineer constructs goal hierarchies. Each goal, constraint, and problem is treated as a high-level goal, and decomposed into a goal hierarchy. Three of the four resulting higher-level goals are shown in figure 1.3, which also shows part of the goal decomposition of one of them. In 3 (c) the requirements engineer constructs a goal graph by merging together the goal hierarchies. A part of the result is shown in figure 1.4.

In 3 (d) the requirements engineer analyses the goal graph for conflicting goals. Figure 1.4 shows that two conflicting goals have been identified: "automate sequencing and planning" conflicts with "manage problems and emergencies". In 3 (e) the requirements engineer scrutinizes the leaf goals of a goal graph in order to identify areas which might benefit from partial or complete automation through computer-based systems. Here, two have been identified: "automate sequence planning" and "automate aircraft sequencing" (see figure 1.4). In 3 (f) the requirements engineer describes the automation strategies that are identified in the previous step. Here, "Automate sequencing and planning" will be completely automated. The associated computer-based system "will automatically detect conflicts and divergences and order the aircraft in time and space in the most efficient way" [LK95b][34]. "Automate aircraft sequencing" will be partially automated: the associated computer-based system "manages only aircraft sequencing" while the controller selects among alternate sequences, updates flight plans and so on. In 3 (g) the requirements engineer predicts the effects of each automation strategy and categorise them on a scale ranging from beneficial to harmful. Here, the completely automatic "Automate sequencing and planning" is likely to minimize the risk of human error. However, it cannot handle abnormal situations. But they can be handled by the partially automatic "Automate aircraft sequencing". In 3 (h) the requirements engineer chooses as the best automation strategy the one that evaluates as most beneficial in the previous step. Here, "Automate aircraft sequencing" is chosen. In 3 (i) the requirements engineer determines what goals for information systems are motivated by the leaf goals of the chosen automation strategy. Here, the goal "Visualise air traffic scenarios in real-time" motivates the introduction of the following goal for an information system: "The system should display scenarios" (see figure 1.5). In 3 (j) the requirements engineer determines what functional requirements for computer-based information systems are motivated by the information systems goals. Here, functional requirements

[34] Page 66

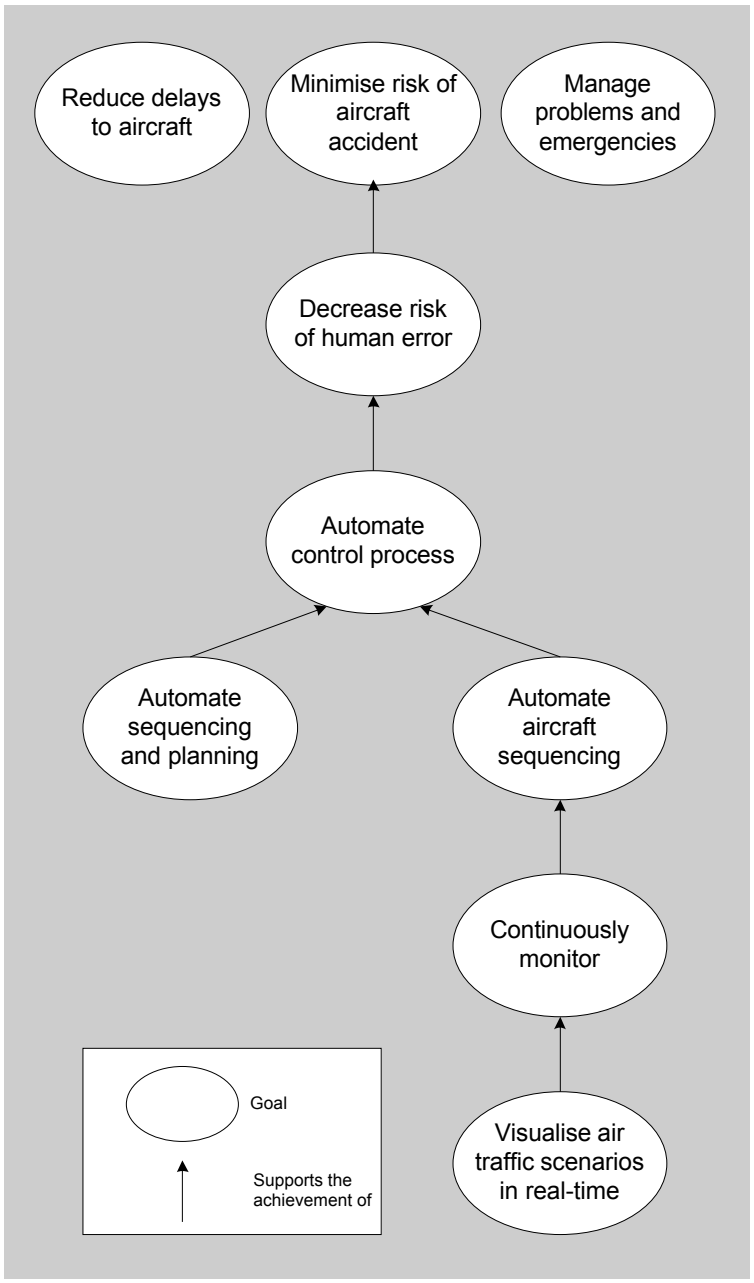

Figure 1.3: Air Traffic Control goal hierarchies (after [LK95b])

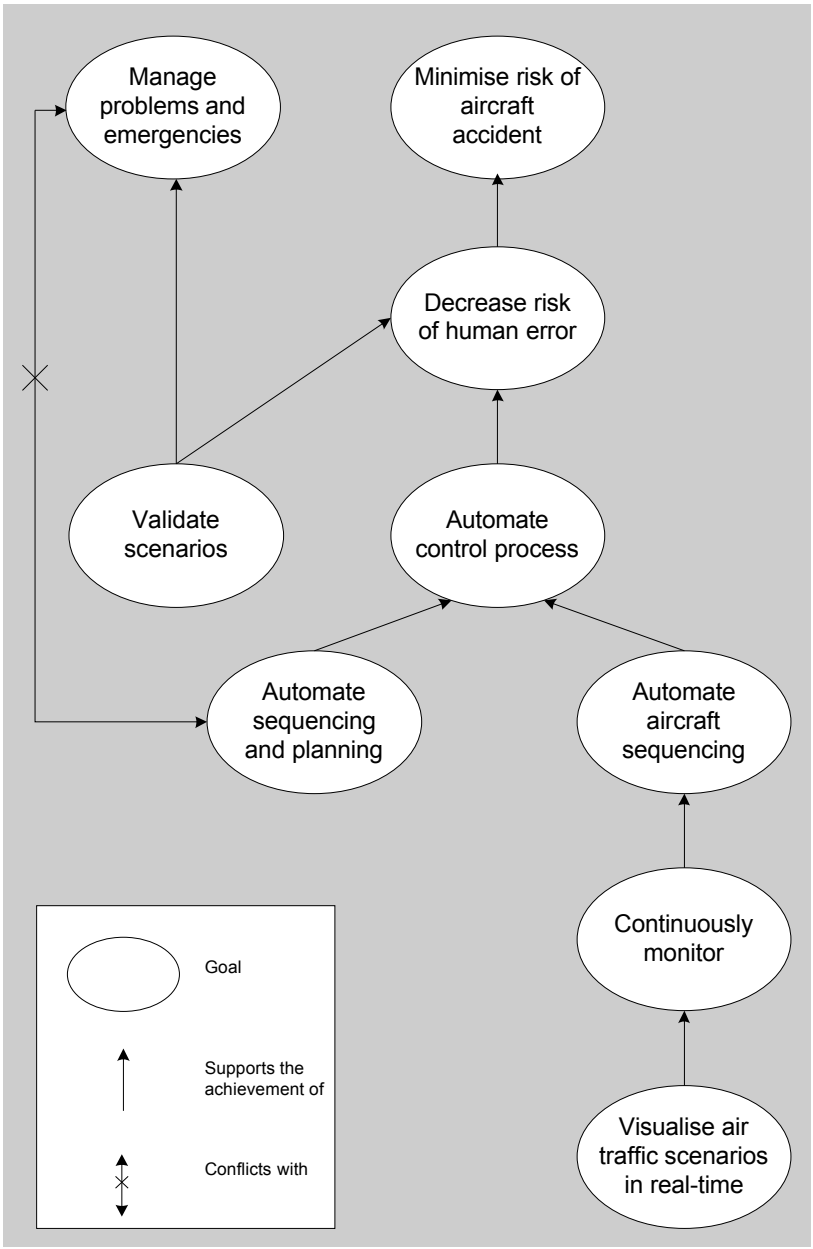

Figure 1.4: Air Traffic Control goal graph (after [LK95b])

include the following: "Display tracks", "Display maps and airways", and so on (see figure 1.5).

In step 3 (k) the requirements engineer determines what non-functional requirements for computer-based information systems are motivated by the information systems goals. Here, the goal "Visualise air traffic scenarios in real-time" motivates the introduction of the following non-functional requirement for computer-based information systems: "The system should perform in real-time". This abstract requirement, in turn, motivates the introduction of other non-functional requirements, which, in turn, motivate the introduction of others, until the following detailed requirement is motivated: "Display aircraft in $< 3/16$ sec of sweep period".

A critical evaluation of Loucopoulos's and Kavakli's approach yields two potential problems. First, Loucopoulos and Kavakli seem to assume that there is a close relationship between the goals of an enterprise and the personal goals of its members: "The members of the enterprise work toward the joint enterprise objectives, perhaps different but related to their personal objectives". However, a number of writers, for example Checkland [CH98] and Jackson [Jac91] challenge this view: they maintain that in many organisations individual members and groups of members have goals that differ both from the official enterprise goals and from each other. And second, although Loucopoulos's and Kavakli's approach is very comprehensive, it may be difficult to apply because it is not described in great detail.

Served system scope defined by the application boundary

The boundary of a served system may be taken to encompass a whole enterprise. On the other hand it may be taken to encompass just some part of an enterprise. Parts may be determined on the basis of structural division. In this case, example parts might include, for example, division or function or project or team. Or they may be defined in terms of some service or function required throughout an enterprise. Internal e-mail and meeting scheduling are examples of such parts of an enterprise. A number of goal-oriented approaches to deriving requirements for computer-based systems have been developed that are capable of working with the served system defined as an application service or function with an enterprise-wide scope [Fea87, FH92, DvF93, BJT+94, Chu93]. These approaches are illustrated in [Gre93], where they are applied to the classic "meeting scheduler" problem (see Appendix **??**). One of these approaches—KAOS—is described again here, since it provides an illustration of the most formal application of the goal-oriented approach to deriving requirements for computer-based systems.

The aim of the KAOS approach [DvF93] is to derive a description of a system's behaviour and an initial analysis of its structure through acquiring and formalising functional and non-functional requirements (or goals) for a composite system. Figure 1.7 shows the inputs to KAOS.

The meta model is actually an integral part of KAOS, but is included here as an input as it may be changed occasionally. The meta model is like an ERA schema; here it characterises composite systems. As such it comprises meta concepts, for example goal, constraint, agent, action, and so on; meta relationships, for example agent-performs-action, action-ensures-constraint, constraint-operationalises-goal, and so on; meta attributes characterising

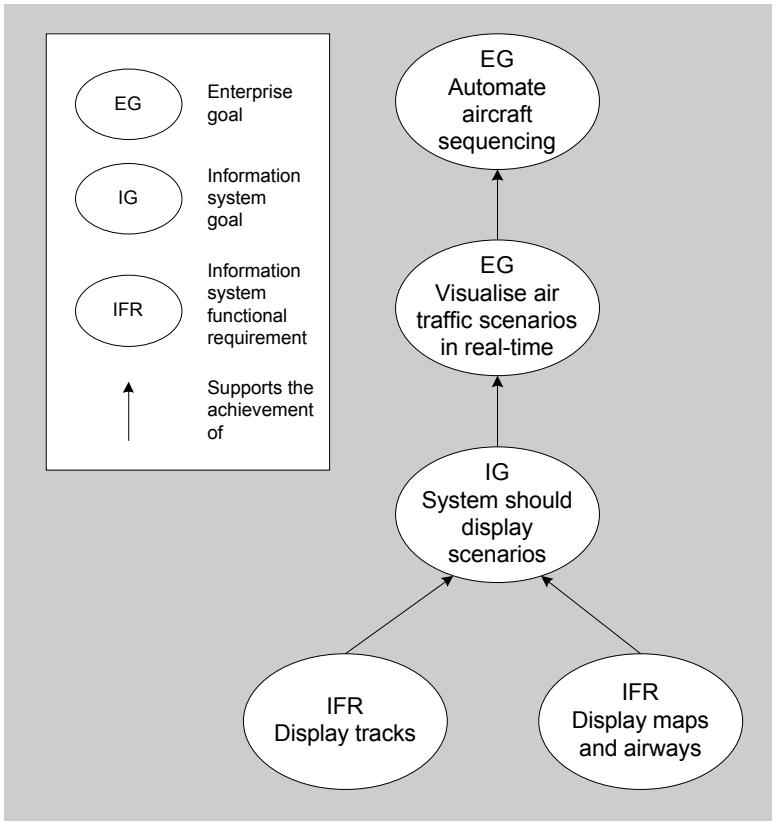

Figure 1.5: Air Traffic Control functional requirements (after [LK95b])

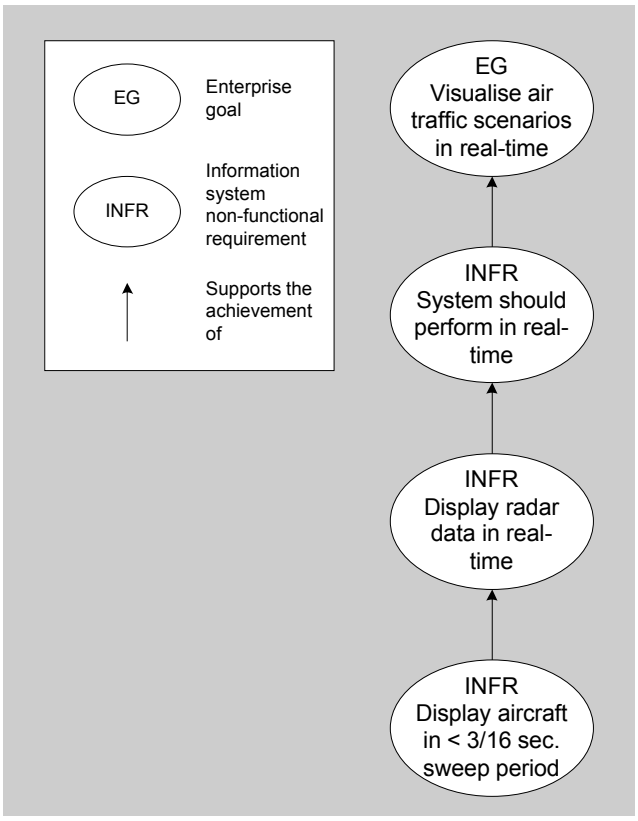

Figure 1.6: Air Traffic Control non-functional requirements (after [LK95b])

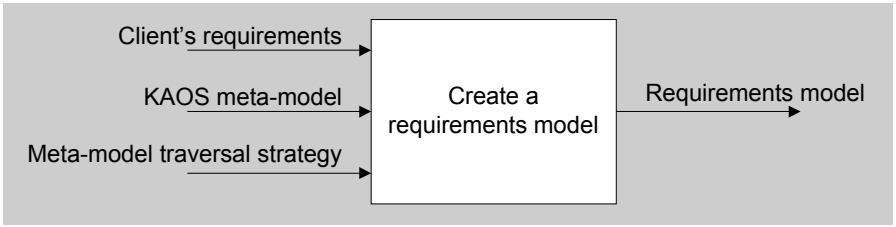

Figure 1.7: Overview of the KAOS approach

both, for example pre-condition is a meta attribute of the action meta concept and strengthened pre-condition is a meta attribute of the ensures meta relationship; and meta constraints, for example constraints defining the cardinality of a meta relationship. Figure 1.8 shows a part of a KAOS meta model. The order and manner in which meta objects are acquired is determined by the meta model traversal strategy (or acquisition strategy). The meta model could be traversed backwards, either from identified agents, or from client supplied scenarios; however, for the remainder of this section, only backwards traversal from a client's requirements (or goals) is considered. Knowledge about traversal strategies is included in meta knowledge used by a computer based acquisition assistant to guide an actual traversal. The assistant also uses domain knowledge of varying degrees of abstraction in order to perform analogical reasoning.

The initial basic source of information for this approach, using a goal-driven strategy, is a client's expression of functional and non-functional requirements for a composite system. These are input to the seven-stage KAOS process, which is shown in figure 1.9. During the KAOS process, instances of each of the meta objects discussed above are acquired, as a requirements model is gradually created. During stage-one, the highest level goals of a system are identified from the requirements. Goals are deemed to be system objectives which cannot be met by the actions of just one agent. Such goals are reduced to an equivalent set of objectives each of which can be met by the actions of one agent. These objectives are called constraints and are formally defined during stage-three. The reduction is achieved by decomposing goals into one or more sets of sub-goals, where each member of a set contributes to the satisfaction of a parent goal, and satisfying all the sub-goals in a set completely satisfies the parent goal [Nil71]. If it is possible, goals should be expressed formally using a first-order temporal logic (currently a formalism inspired by ERAE [Dub91] is used by the KAOS group). As the goal decomposition proceeds, objects (agents, entities, relationships, and events) referred to in formal and informal goal descriptions are identified and abstracted for use in stage-two.

During stage-two, objects associated with goals are reviewed, and agents (objects which control state transitions) are identified along with their actions and any pre-conditions, post-conditions, and trigger-conditions for their actions. This knowledge of agents and actions is then used in stage-one to identify when a goal may be reduced to a constraint, in other words to identify leaf-goals. Thus, stages one and two may be viewed as co-routining stages. In stage-three, each leaf-goal of a goal structure is converted into a constraint whose formal definition is expressed in terms of objects and

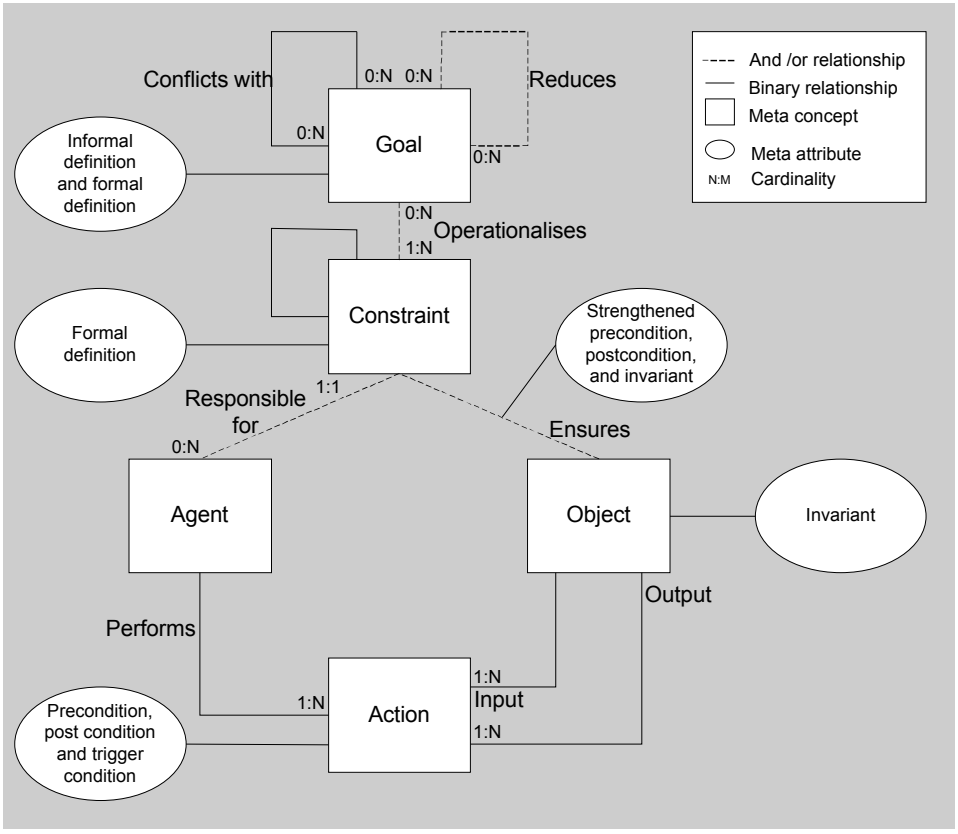

Figure 1.8: A portion of a KAOS meta model

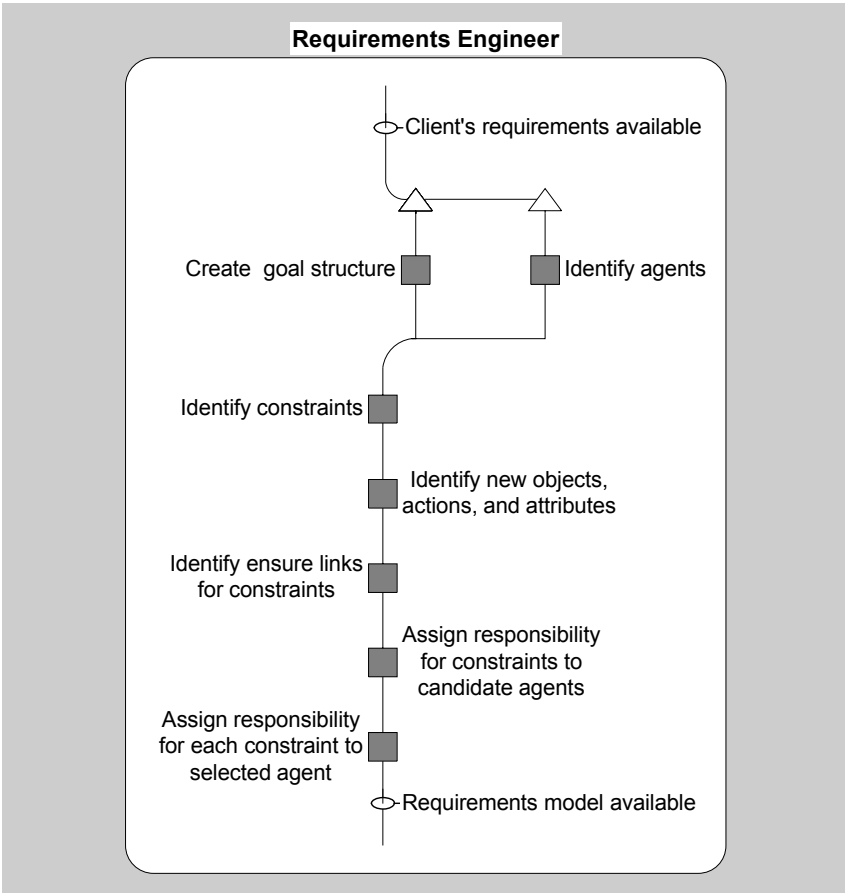

Figure 1.9: The KAOS process

actions available to some agent identified during stage-two.

In stage-four, new actions and objects described in the formal definitions of constraints during stage-three are identified and acquired. In addition, new characteristics of already acquired concepts, such as new meta attributes or new pieces of invariants, pre-conditions, post-conditions, and trigger-conditions are also identified. In general, actions, pre-conditions, post-conditions, and trigger-conditions identified in stage-two are not necessarily sufficient to ensure that each constraint will be satisfied. So, in stage-five, each constraint is examined and consequently, to ensure that each constraint will be met, some actions and objects may be modified, and new actions and objects may be introduced. Such modifications might involve strengthening an action's pre-condition, for example, or strengthening an object's invariant. This new information is held in a requirements model as meta attribute values of the ensures meta relationship. In addition, new actions are acquired which allow soft constraints (see below) which may have been violated to be restored.

In a complete requirements model each constraint is the responsibility of a single agent. During stage-six, each constraint is reviewed in turn, and agents which might be made responsible for it are linked to it by instances of the responsibility meta relationship. Finally, in stage-seven, the actions required to satisfy a constraint are assigned to the "best" agent from among the candidate agents for satisfying that constraint.

The rationale for the meta model and the acquisition process is as follows: the system goals correspond to a client's requirements; these are satisfied if all the leaf goals of the generated goal structure can be satisfied; these, in turn, can be satisfied if the constraints operationalising them can be met. The formal definition of each constraint defines a set of sequences of states in terms of properties of objects in the states, that is to say in terms of patterns of states. In order to move from one state in a sequence to the next, actions must be performed on one or more of the objects in the input state. In general, a set of actions (with appropriate pre-conditions, post-conditions and trigger conditions) acting on objects (conforming to appropriate invariants) will be needed to meet a constraint. A number of agents may be able to perform actions which will enable them to satisfy the constraint. One of these is selected as the agent which is responsible for meeting the constraint; this one has the appropriate actions assigned to it. The approach is used to create a requirements model for a composite system. This model comprises, among other things, a set of agents, where each agent has a set of actions and a set of constraints for which it has been assigned responsibility. Different constraints may conflict with one another.

The KAOS approach is illustrated here by applying it to the Meeting Scheduler problem (see Appendix **??**). The initial basic source of information for this approach, using a goal-driven strategy, is a client's expression of functional and non-functional requirements for a composite system. For example, for the meeting scheduler system, such goals include the following: "to determine, for each meeting request, a meeting date and location so that most of the intended participants will effectively participate", and "privacy rules should be enforced; an ordinary participant should not be aware of constraints stated by other participants".

A client's requirements are input to the seven-stage KAOS process (see

figure 1.9). Figure 1.10 shows the results of partially decomposing one of the meeting scheduler system's functional goals during stage-one. From complete goal descriptions (not given) for the goals referred to in figure 1.10, it is possible to identify from the Schedule Meeting goal the following objects: meeting request, date, location, meeting scheduler, attendee (in other words meeting participant); and actions: participate. During stage-two, these objects and actions are reviewed, and Attendee and Meeting Scheduler are identified as agents, and Participate (at meetings) is identified as an action of the Attendee agent. Similarly, an Authorised User agent, and Request Meeting and Accept Meeting actions are also identified during stage-two.

Continuing the example, in stage-three, the Attendees' Preferences Known system goal is operationalised into two constraints: a hard constraint (hard constraints must not be violated), Request For Attendees' Preferences Made, and a soft constraint, Request For Attendees' Preferences Satisfied. This is shown in figure 1.11. The first constraint may be satisfied by a Request Attendees' Preferences action available to the Meeting Scheduler agent; the second may be satisfied by the Submit Preferences action available to Attendee agents. It is assumed that both actions were acquired from the client during stage two.

In stage-four, it is determined that the two constraints in the example contain no new objects or actions, nor do they contain anything which completes the description of objects and actions already identified. In stage-five it is recognised that Request For Attendees' Preferences Satisfied is a soft constraint which signifies that all the requests for attendee preferences will be satisfied within a predefined time period. If the constraint is violated, actions are required to help to restore it. One such action might be Send Preference Request Reminder; in the meeting scheduler requirements model, this action would be linked to the constraint by an instance of the restoration meta relationship.

Stage-six and seven are concerned with allocating agents. In the example, the Attendee agent is linked by a responsibility meta relationship to the Request For Attendees' Preferences Satisfied constraint, and the System Scheduler agent is similarly linked to the Request For Attendees' Preferences Made constraint. Had other agents been identified earlier which might also have been made responsible for either of these constraints, then they would also have been similarly linked. Finally, in stage-seven the Request For Attendees Preferences Made is assigned to the Meeting Scheduler agent by an instance of the performs meta relationship link. Figure 1.12 shows a fragment of the requirements model acquired for the meeting scheduler system by the approach just described.

Critical evaluation of the KAOS approach identifies two potential problems: the lack of tool support for KAOS combined with its use of formal notations (temporal logic) may make non-experts reluctant to use it in real-world organisational contexts.

1.5 Advantages and problems of the goal-oriented approach

The main advantages of the goal-oriented approach to requirements engineering have been summarised by Axel van Lamsweerde [vL01]. First, the

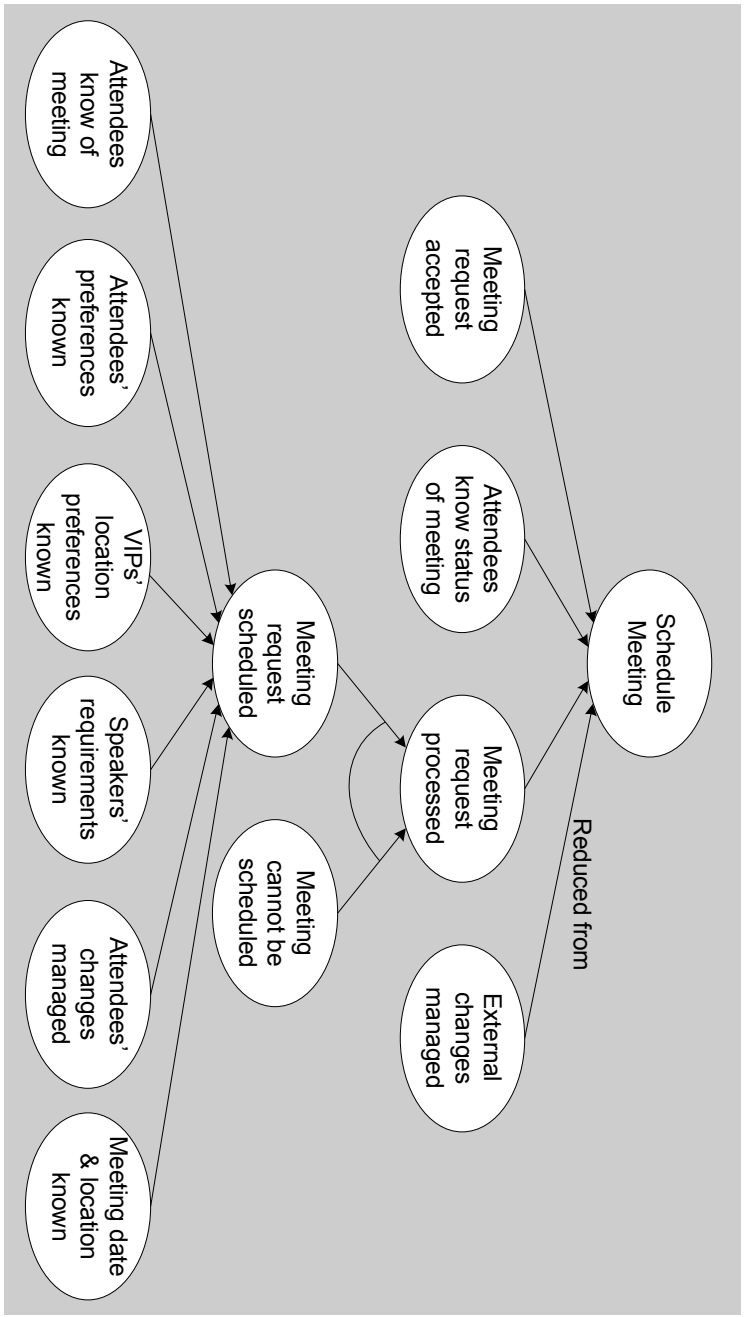

Figure 1.10: Partial decomposition of a meeting scheduler system goal

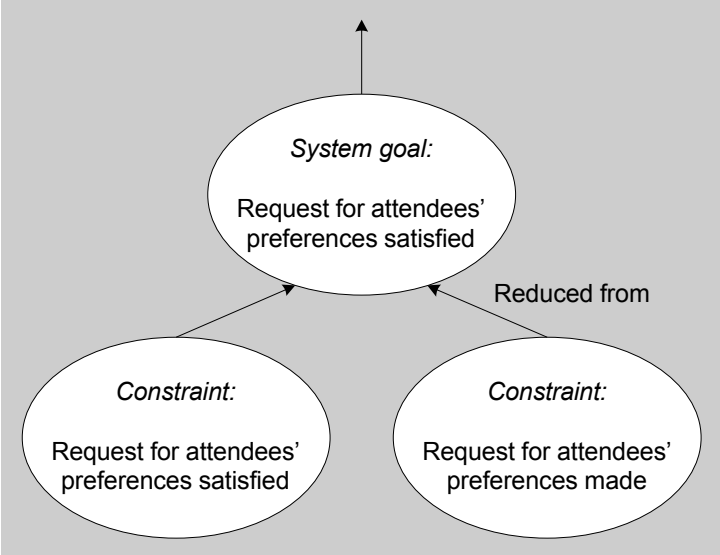

Figure 1.11: Operationalising goals into constraints

goal-oriented approach is characterised by a systematic requirements elaboration process: "Goals drive the identification of requirements to support them." Second, the approach facilitates the detection and resolution of intrinsic conflict: "Goals have been recognised to provide the roots for detecting conflict among requirements and for resolving them eventually." Third it facilitates the exploration of alternative "solutions": "Alternative goals provide the right level of abstraction at which decision makers can be involved for validating choices being made or suggesting other alternatives overlooked so far." Looking for alternative solutions means that more solutions, and, in particular, more radical solutions are often found. Fourth, the approach is associated with achieving requirements completeness: "Goals provide a precise criterion for sufficient completeness of a requirements specification." Fifth, goals may provide a rationale for requirements: links from requirements to goals may be used to explain and justify requirements to stakeholders. Sixth, the goal-oriented approach helps the requirements enginer and the stakeholders to avoid irrelevant requirements: "Goals provide a precise criterion for requirements pertinence." And seventh, goals may help to structure requirements documents: "Goal refinement provides a natural mechanism for structuring complex requirements documents for increased readability."

In addition to the advantages identified by van Lamsweerde, there are other advantages. First, the goal-oriented approach may facilitate the understanding of a system. The goal-oriented approach generates a set of goals for a system, and reading goal descriptions may help to provide a high-level understanding of the purpose and nature of the system. Second, the approach may facilitate evaluation of a system, since goals constitute criteria against which an implemented system may be judged. Third, the approach encourages implementation detail independence by making goals (relatively abstract), rather than activities (relatively concrete), the centre of focus.

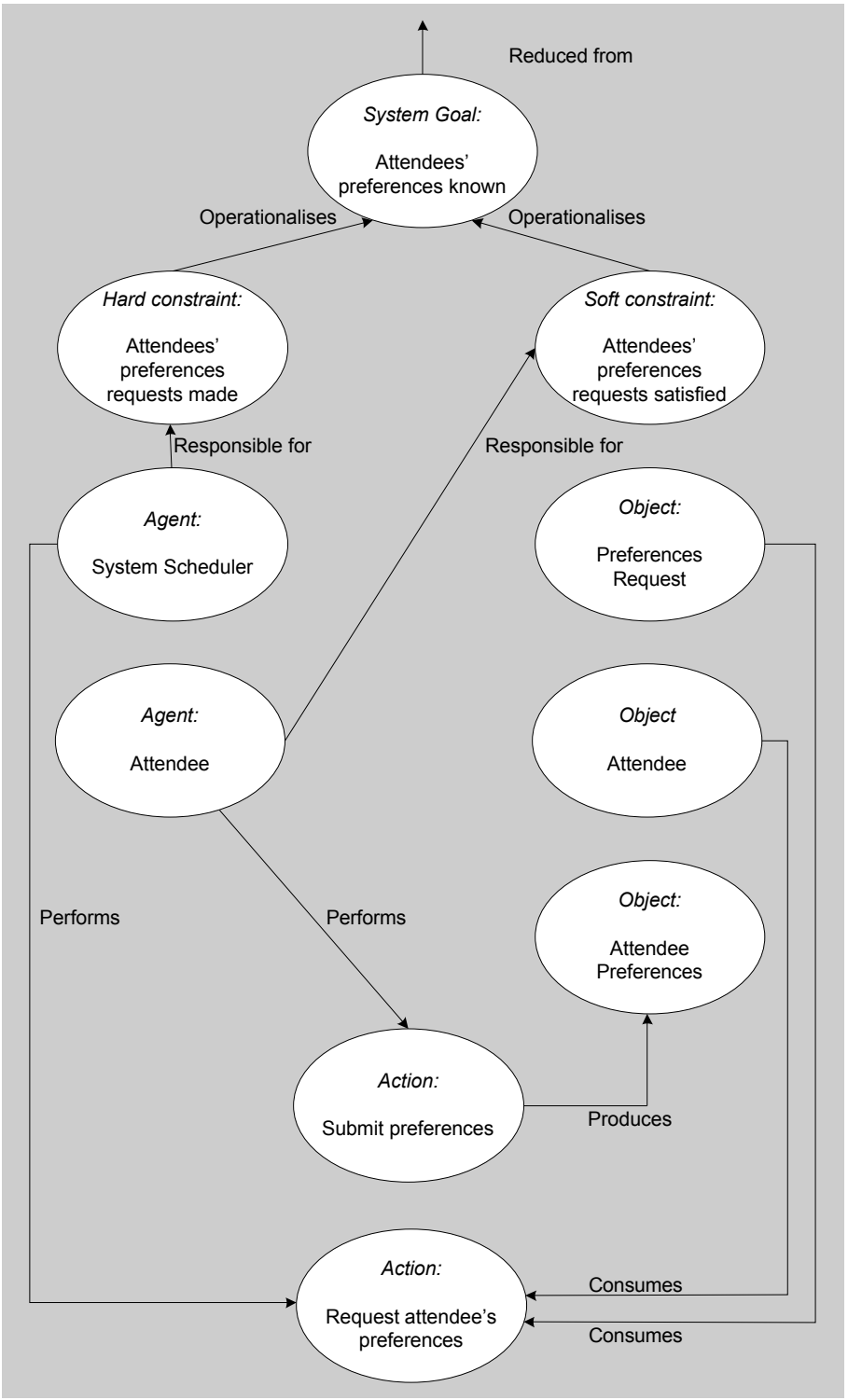

Figure 1.12: Fragment of a requirements model for a meeting scheduler system

Fourth, it facilitates the provision of re-usable structures: established goal structures may be re-used in similar domains; see, for example, GMARC in [BJT+94]. Fifth, the approach provides, in goals, a starting point for traceability links between all the artefacts created during the development of a piece of software [GF96] and [Bir99]. Sixth, the approach may facilitate the evaluation of designs: a design may be judged according to how well it satisfies a set of goals [Chu91]. And seventh, the approach may be used to faciltate maintenance: in an approach developed by Karakostas [Kar90] requirements that need changing "point to" the system components that may be impacted by the change.

However, although the goal-oriented approach has a number of advantages, there are also a number of potential problems associated with it. For example, one problem is associated with decomposing goals: how can one be sure that a goal is precisely satisfied by its sub-goals? Another problem related to the difficulty that sometimes arises over distinguishing goals from solutions: "Distinguishing between primitive goals (leaf-goals) and the means to achieve them is not always clear" [LK95b][35]. While a third problem concerns the dynamic nature of many organisations. In such organisations the goals may change so quickly that the goal-oriented approach is rendered less useful.

1.6 Goals and processes

This section reviews work that exemplifies the process-oriented approach to requirements engineering, an approach that focuses upon processes as well as upon goals. A process is taken to comprise a collection of activities undertaken by one or more agents (people or computer-based systems) some of which involve interaction between the agents. Processes are normally intended to achieve goals, in other words the activities of a process are normally intended to produce a state corresponding to a goal to be achieved. Process models are used to describe processes. They may represent both the activities of a single agent, and activities involving two or more agents.

Once a process has been characterised and expressed as a process model, a number of uses may be made of it. First, it may be analysed to see, for example, how well it meets its intended goals; second, it may be modified in order to try to produce process performance improvements. Provision of IT support for a process constitutes one kind of modification. Appropriate IT support, and thus the requirements for it, may be determined in at least two distinct ways: either the existing process may be analysed and and areas which might benefit from IT support may be identified (see figure 1.13); or, alternatively, the goals of a process might be reviewed in parallel with an appraisal of potentially relevant IT. Following this, a new process could be designed that would exploit the latest IT in order to achieve the goals (see figure 1.14). One or other variant of this method is found in recent work on Business Process Engineering (BPR) [Dav93, Ham90, HC93, Oul95]. It is also found in more recent work that focuses upon providing computer-based support for business processes [WKRG99].

Three process-oriented approaches are reviewed here, with a view to surfacing ideas that might be incorporated in the synthesised approach to

[35] Page 60

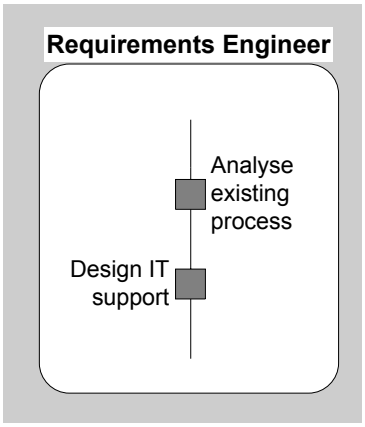

Figure 1.13: Provision of IT support (one)

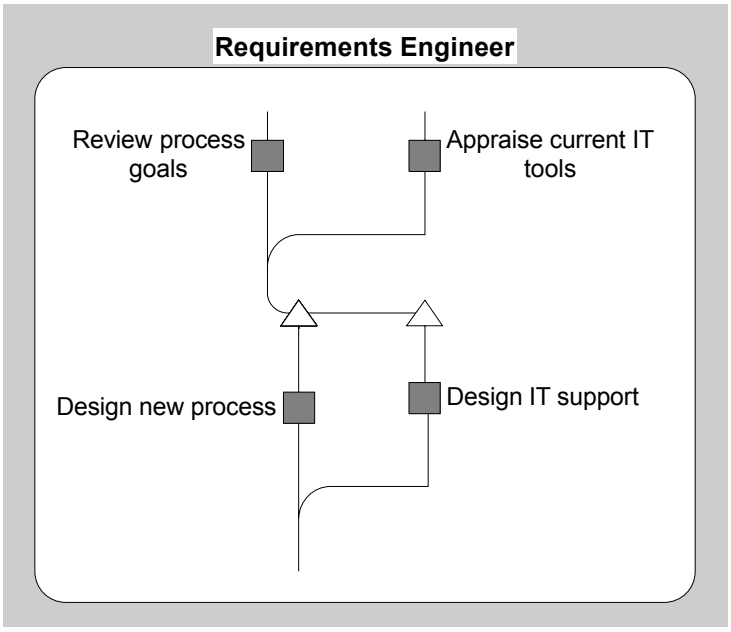

Figure 1.14: Provision of IT support (two)

goal-oriented requirements engineering that is described in the next chapter. The three approaches are: Davenport's [Dav93], Ould's [Oul95], and the approach of Warboys and his colleagues [WKRG99]. The reviews also identify problems with each approach that the new, synthesised approach is intended to eliminate or attenuate.

Davenport's approach

Among the BPR initiatives, Davenport's [Dav93] method conforms closely to the first variant of the goal-oriented method outlined above. Davenport states: "The objectives of business process innovation fall into three groups, which tend to lead into one another: strategic objectives lead to process-related objectives, which lead to information (computer-based) systems-

related objectives" [Dav93] [36]. Davenport has designed a method for deriving requirements for IT process support from captured strategic objectives. This method is first presented in outline, and then elaborated.

- Identify processes to innovate
- Identify IT enablers for new process design
- Define business strategy and process vision
- Understand the existing process
- Design and prototype the new process

For the first stage, Davenport suggests that the processes to innovate could be chosen informally, based upon, for example, their perceived centrality in relation to the firm's business strategy or upon their health, in other words upon whether or not they were exhibiting serious problems. He also sketches a more formal procedure for selecting processes to innovate: first, identify and prioritise the goals of an organisation; then identify its main processes; and, finally, choose process(es) to innovate based upon the number and importance of the goals that they support.

For stage-two, Davenport recommends that, before undertaking process redesign, a process redesigner should be aware of the range of IT tools that have potential for supporting the process:"A process designer pursuing innovation should consider all the tools that can be used to shape or enable the process, and IT and the information it provides are among the most powerful" [Dav93]. Knowledge of the potential of such IT tools may then actually influence design decisions. This method is arguably better than the method in which a process is redesigned in the absence of such knowledge, followed by a search for IT tools to support the resulting process. Better because, in the former case, potentially applicable, powerful tools may be more easily incorporated into the process, and, in the latter, they are much less likely to be. Davenport suggests three mechanisms that are intended to facilitate the selection of potentially useful IT tools:

- Maintain an index of IT-supportable areas
- Map IT tools to generic business areas
- Carry out benchmarking, in other words examine how the same or similar processes have been implemented in other organisations or contexts

Stage three of the approach is concerned with defining a business strategy and process vision. Davenport identifies a series of "drivers" for his method—strategy, process vision, process goals, actions—where each driver is used to derive the next one in a series. So, the key driver is the first one: a well-defined strategy. It has already seen that this may be used to select a process to innovate. Here it can be seen that it may be used in "the development of process visions for processes to be innovated". For Davenport, a strategy and a process vision are closely related phenomena: but "we

[36] Page 215

view strategy as long-term directional statements on key aspects of a firm or business unit, and vision as a detailed description of how, and how well, a specific process should work in future". Strategy includes process-oriented, product-oriented, and financial goals that are measurable and to be achieved in a time-frame of five to ten years. On the other hand, a process vision which is derived from a strategy comprises the main goal of the process and the other process objectives, preferably quantified, and is to be achieved in a shorter time-span. A typical process objective might be: "reduce new drug-development cycle time by 50% in three years".

The existing process should be characterised in stage-four. This involves both modelling its structure and flows, and (ideally) measuring its performance. For Davenport, the main reason for the former is to understand the cause of a process's problems. The main reason for the latter is to make performance comparisons between the old and new process.

When stage-five is reached, a process re-designer should have knowledge about the current process, a vision for the new process, and IT tools offering the potential to support the new process. Guided by the process vision, the process redesigner redesigns a new process. He or she includes in the design appropriate IT tool support and thus produces, if only in embryonic form, descriptions of the requirements of such tools. Davenport recommends that the new process should be prototyped and the prototype tested by the users. Such prototypes are intended to include prototype versions of the supporting IT tools.

Critical evaluation of Davenport's approach yields potential problems that might make it difficult to apply. First, it is worth noting that each of the three mechanisms in stage-one is based, at best, upon the current manifestation of a process (including its goals). However, a new process is not designed until later, and its constituent activities might be different from the activities of the current process. But IT tools chosen for their relevance to the current process manifestation might not prove to be so relevant to a redesigned process. Second, it is worth noting that since a strategy is used to select processes to innovate, then strategy definition should be performed prior to the first step, Identify Processes to Innovate. Third, the approach's steps are not expressed in sufficient detail to allow unambiguous interpretation. And finally, Davenport has not provided a process modelling language. The third problem is tackled in the approach synthesised here (see next chapter). The latter problem has been addressed by a number by researchers who have created languages or notations for modelling processes: activity diagrams in UML [Fow97], and Role Activity Diagrams (RADS) [Oul95].

Ould's approach

Two criticisms of Davenport's method—insufficiently detailed process steps, and lack of a process modelling language—are addressed successfully by Ould [Oul95]. Ould's method is worked out in more detail than Davenport's, and, in addition, Ould provides a language for modelling processes: Role Activity Diagrams (RADs) (See Appendix **??** for details of the RAD notation). Ould [Oul95][37] presents an eight stage model. Here, for simplic-

[37] Page 182

ity, a condensed version of his method is presented. It has four main stages, as follows:

- Decide on the objectives of modelling
- Investigate and model the current process of interest
- Analyse the current process
- Respond to the analysis

This method may be specialised in a number of ways depending upon the objectives of the modelling. However, here, the interest is in both improving the current process, and in providing IT support for either an existing or an improved process. Hence, in this case, the method given above specialises to the following:

- Investigate and model the current process of interest
- Analyse the current process in order to improve it
- Support the improved process with IT

The details of these stages are given in [Oul95][38]. In the first stage various sources of information—people, documents, terms of reference, meetings—are accessed. For example, people involved in the process are questioned about it individually and in groups. The following kinds of information is elicited:

- Process goals
- Roles
- Interactions
- Triggers
- Process problems
- Suggested process improvements

In stage-two, the current process of interest is analysed with a view to improving it. The most important mechanism for achieving improvements is based upon a movement from the current concrete process to an abstract process, then another movement to a new concrete process. First, one abstracts away from the "concrete" current process and produces an "abstract" process. As Ould notes: "in an abstract model we look at intent or purpose". In other words, the abstract model is providing a view of the goal(s) of the process. From the abstract process model, a new concrete process is designed. The differences between the original and new concrete processes may include changes to individual actions, changes to interactions, changes to role structure, or changes to the assignment of roles to parts of the organisation.

In stage-three, the improved process is examined to see which of its areas can be supported by IT, and which of these areas is amenable to which sort of technology. Ould identifies three basic kinds of IT support:

[38] Chapters Nine and Ten

- Traditional information systems for supporting information needs of individuals
- Workgroup computing for supporting soft complex interactions between people
- Workflow management for supporting the overall process and flow of activity

Ould also provides heuristics for generating requirements for supporting computer-based systems that support an improved process:

- Review black-box activities in RADs to see whether they may be:
  - fully automated
  - performed by humans with computer-based system support
- Review interactions in RADs to see whether they may be supported by:
  - workflow management systems
  - workgroup computing systems

Critical evaluation of Ould's process-oriented approach yields two potential problems. First, the new process is derived from the abstract form (goals) of the current process. It seems that both users' problems with the current process, and their new goals for the new process, although perhaps elicited during the "investigate and model the current process" stage, are not explicitly used to derive the new process. Second, whereas in Davenport's method, IT tools that might support a new process are identified either before or while the new process is being designed, in Ould's method, IT tool support is determined after the new process has been designed. But, as has been already observed, because the design of the process is not influenced explicitly by potentially appropriate IT tools, it seems less likely that the most appropriate tools, once identified, will be easy to integrate into the process. In the approach presented in the next chapter, both of these criticisms are addressed. It explicitly elicits stakeholders' views of the current process problems and new goals, and uses both in developing a new process. Second, like Davenport, it suggests that the identification of appropriate IT tools and the design of the new process should proceed in parallel.

Organisational Process Modelling (OPM)

Kawalek et al. [WKRG99] have developed the Organisational Process Modelling (OPM) approach to deriving software. Organisational Process Modelling focuses upon goals and processes and is used to derive requirements for computer-based systems intended to support organisation processes. Such computer-based systems, termed active models, are designed to coordinate the actions of people and software tools. They are also intended to coordinate interaction between people, between tools, and between people and tools. Thus such active models are intended to guide and control process enactment within organisations.

The work of Kawalek et al. extends that of Ould through both the introduction of new models in addition to the Role Activity Diagram models, and the introduction of new notation to the existing Role Activity Diagram notation. Kawalek et al. also focus mainly upon the production of active models, unlike Ould, who focuses equally upon workflow systems, groupwork systems and traditional IT support (database and personal productivity tools). Indeed, Kawalek et al. have designed a language, the Process Modelling Language (PML) for specifying active models.

Organisational Process Modelling conceptualises organisations in terms of people who interact both with each other and with software tools in order to achieve organisational goals. Through considering the interactions between people, the goals that they are seeking to carry out, and the activities that they carry out, a user of Organisational Process Modelling identifies software systems intended to support the behaviour, and thus to achieve the goals [WKRG99][39]. Organisational Process Modelling "is concerned with the description of operational goals of users and their mapping to technical capabilities. Thus, it seeks to describe why people collaborate, and how their goals are fulfilled"[WKRG99][40].

Figure 1.15 summarises Organisational Process Modelling in a Role Activity Diagram. Using the notation's magnifying glass symbol for identifying goal states, the diagram shows that the output from the Organisational Process Modelling process is an active model. This comprises one or more computer-based systems (and therefore their corresponding requirements) that are intended to support organisational processes. The support constitutes managing the actions and interactions of people and tools participating in the processes. The active model is produced during the last stage of Organisational Process Modelling, that is to say during the stage called "Design IT support". The "Design IT support" activity works upon both a description of the organisational context and problems encountered in that context in order to derive an active model. The organisational model is expressed in terms of three different models: a system (role) model, a goal model, and one or more method models. The first represents people working in context; the second their organisational goals; and the third, one or more ways through which the people might achieve their goals. The latter model is expressed in an augmented Role Activity Diagram.

The first stage, Investigate Context, focuses first upon the investigation as an intervention in the organisation, and second upon the process of interest. The intervention will probably take the form of a project. Issues of concern might include, for example, timescales, budget, resources, aims, scope, deliverables, etc. The focus upon the process of interest involves first of all investigating the process—its objectives, boundary, inputs and outputs, participants, performance problems, and so on. Second, it involves a thorough capture and analysis of the objectives of the process [WKRG99][41]. Methods used to capture and analyse the process objectives are listed below. The main outputs of stage-one are details of the process of interest: its goal(s), its boundary, and its participants.

- Discover goals from documents and from stakeholders

[39] Pages 4, 11, 15, 68, and 72
[40] Page 68
[41] Page 92

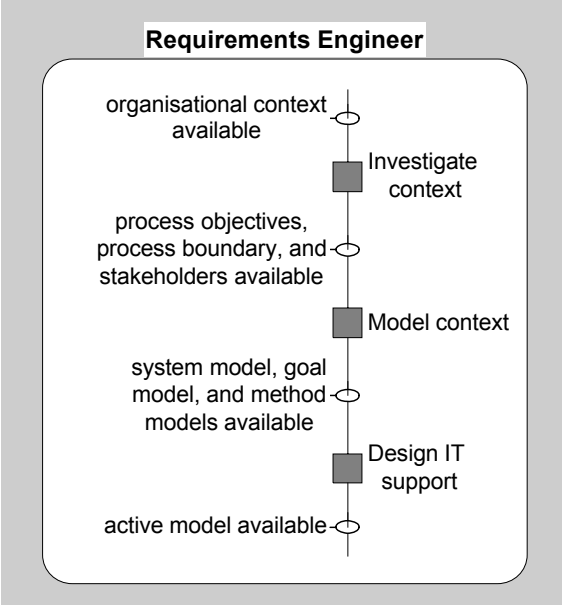

Figure 1.15: Organisational Process Modelling: a Role Activity Diagram model

- Cross-check goals between process participants and other stakeholders
- Build goal hierarchies using top-down decomposition
- Identify conflicting goals
- Assess how well goals are satisfied by the process
- Prioritise goals
- Identify low-level operational goals of agents participating in the process

Stage-two of Organisational Process Modelling is concerned to identify interacting agents, to describe their goals, and to describe how the goals may be achieved. Three different models—system model, goal model, and method model—are used to model these different aspects of the process of interest. The first model, the system model, depicts interaction between agents. For example, in an organisational context where IT users report IT problems to a Helpdesk, the role model shown in figure 1.16, might be obtained. It shows that in general a number of User agents, or people in other words, interact with a single Helpdesk agent, a person or a group of people.

The second model, the goal model, shows what goals are obtained by each agent in a system model. Continuing the example, figure 1.17 shows the goals that are obtained by the User and the Helpdesk agents respectively. It shows that the User agent initiates interaction with the Helpdesk agent in order to obtain a solution to an IT problem, in other words to achieve the goal "solution obtained". The Helpdesk agent's part in the interaction is complete when a solution has been communicated to a user, that is to

Figure 1.16: Organisational Process Modelling: system model for the Helpdesk context

say when the goal "solution given" has been obtained. This goal model also expresses the non-functional goal that the "solution given" goal should be obtained within two weeks.

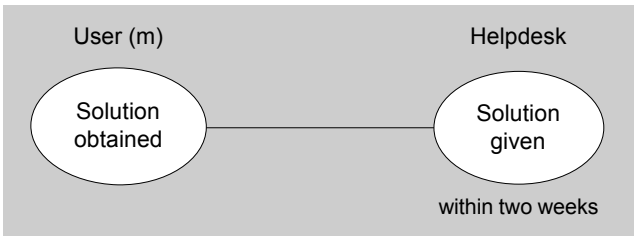

Figure 1.17: Organisational Process Modelling: goal model for the Helpdesk context

The third model, the method model, shows one way in which the goals of interacting agents might be achieved. The modelling formalism is a superset of the notation used in Role Activity Diagrams [Oul95]. Completing the example, figure 1.18 shows how the "solution obtained" and "solution given" might be achieved. It shows that for the User to achieve the goal "solution obtained", they must first communicate the problem to the Helpdesk agent. That agent logs the problem, solves the problem, and communicates the solution back to the User. This activity achieves both the User's and the Helpdesk's goals of interaction.

The main outputs from the stage are the system, goal, and method models. These are input to stage-three along with the context problems uncovered during stage-one.

The purpose of stage-three is to create software that will support the co-ordination of interactions between people, between tools, and between people and tools, as they participate in enacting a process. But which parts of a process should receive such support? In Organisational Process Modelling, process problems that are identified during stage-one (Investigate context) are used to identify parts of the process to investigate for process support by IT. Following reflection upon a process problem and upon the models (system, goal, and method) associated with that problem, the designers suggest ways of eliminating or attenuating the problem using IT support, and in particular workflow techniques. The solution is represented in a new

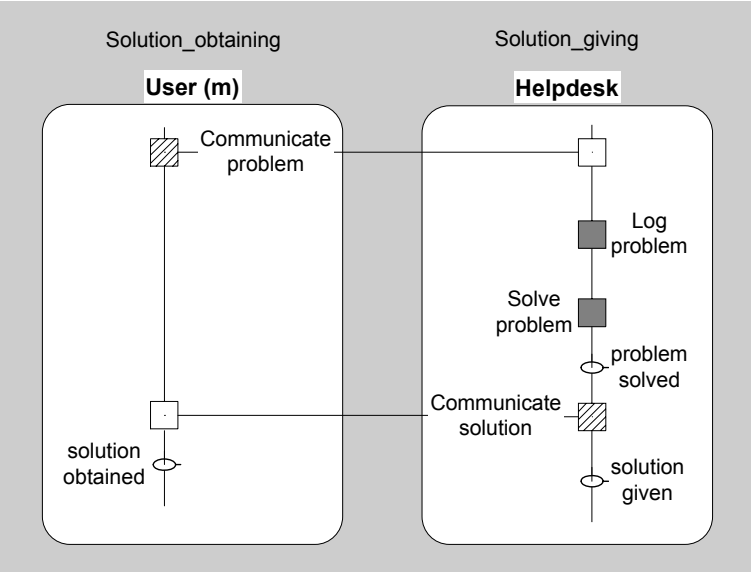

Figure 1.18: Organisational Process Modelling: method model for the Helpdesk context

method model. This model shows just those roles relevant to solving the problem. The IT solution is shown on the model as one or more annotations. The final stage is to use the new method module to design PML code to implement the IT solution featured in the model. Later the PML code may be compiled to create (part of) an active model that will support the process in the intended way. The Role Activity Diagram diagram in figure 1.19 summarises the description just given.

The description above illustrates what Kawalek et al. call refinement. Refinements involve planned change in order to solve problems or to find new ways of meeting existing goals. Kawalek et al. also note that two other categories of change might occur during stage-three, viz reflections and re-inventions. Reflections are the relatively small, unplanned changes that people suggest might be made to processes. They are the kind of ideas that manifest themselves spontaneously when processes are being investigated, talked about, and modelled. Re-inventions, on the other hand, involve either major change that "substantially eradicates existing patterns of behaviour" [WKRG99] or the introduction of a process and its IT support into a "green-field" situation.

Critical evaluation of Organisational Process Modelling yields three potential problems. First, during process capture, it is not clear how the goal hierarchies captured in stage-one are used. In particular it is not clear what use is made of higher-level goals. Second, stage-two seems to model the current organisational processes, and stage-three produces IT support for them (active model). This may lead to the approach providing IT support for just the status quo, rather than for improved or new organisational processes. The latter seems more desirable. And third, stage-three focuses on the goals of agents doing the work. It seems to largely ignore higher-level business

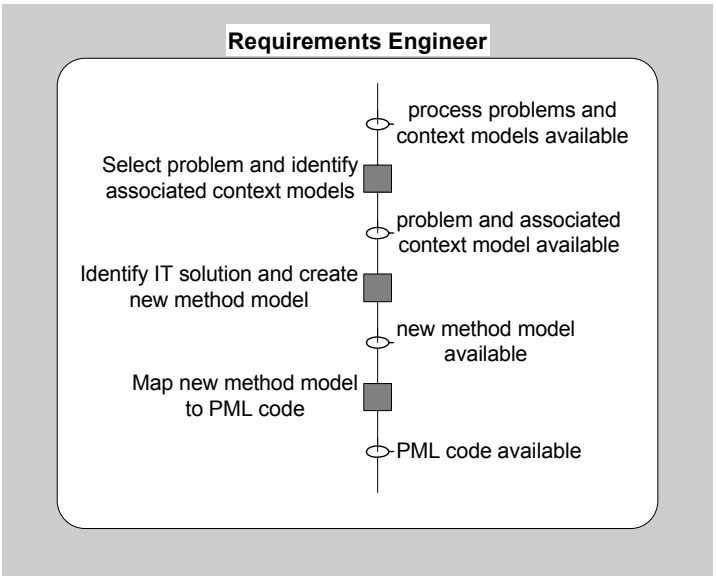

Figure 1.19: Organisational Process Modelling: creating the active model

goals of other agents, for example, managers and process owners. Each of these potential problems is addressed in the the synthesised goal-oriented approach that is presented in the next chapter.

1.7 Summary

This chapter has reviewed goal-based approaches to requirements engineering and process-based approaches to requirements engineering. Problems with all the approaches considered have been identified. In the next chapter a new goal-based approach to requirements engineering is presented. The new approach is largely based upon the principles, ideas, and techniques that are embodied in the approaches that are considered in Chapters Three and Four. The new approach is intended to resolve the problems associated with the approaches already considered.

Bibliography

[AMP94] A. Anton, W. McCracken, and C. Potts. Goal decomposition and scenario analysis in business process re-engineering. In *6th International Conference on Advance Information Systems Engineering*, pages 94—104. Springer Verlag, 1994.

[AP98] A. Anton and C. Potts. The use of goals to surface requirements for evolving systems. In *International Conference on Software Engineering*, pages 157—166. IEEE Computer Society Press, 1998. "Kyoto, Japan".

[Bir99] B. Birkem. Traceability management from business processes to use cases with uml. *Journal of Object Oriented Programming*, September 1999.

[BJT+94] D. Bolton, S. Jones, D. Till, D. Furber, and S. Green. Using domain knowledge in requirements capture and formal specification construction. In M. Jirotka and J. Goguen, editors, *Requirements Engineering: Social and Technical Issues*, pages 141–162. Academic Press, 1994.

[BNG94] J. Bubenko, J. Nellborn, and M. Gustafsson. Enterprise modelling - the key to capturing requirements for information systems. Technical report, Swedish Institute for Systems Development, PO Box 1250, S-164 28 Kista, Swede, 1994. Deliverable 3-1-3-R1 Part A, Espirit III Project 6612.

[BRLD94] J. Bubenko, C. Rolland, P. Loucopoulos, and V. DeAntonellis. Facilitating 'fuzzy to formal' requirements modelling. In *Proceedings of the 1st International Conference on Requirements Engineering*, pages 154—157. IEEE Computer Society Press, 1994.

[CH98] P. Checkland and S. Holwell. *Information, Systems, and Information Systems*. John Wiley and Sons, 1998.

[Chu91] L. Chung. Representation and utilisation of non-functional requirements for information system design. In *Proceedings of CAiSE*, pages 3–50, 1991.

[Chu93] L. Chung. *Using Non-Functional Requirements: A process Oriented Approach*. PhD thesis, Department of Computer Science, University of Toronto, 1993.

[Dav93] T. Davenport. *Process Innovation: Reengineering Work through IT*. Harvard Business School Press, 1993.

[DBCS94] J. Dobson, A. Blyth, J. Chudge, and R. Strens. The ordit approach to organisational requirements. In M. Jirotka and J. Goguen, editors, *Requirements Engineering: Social and Technical Issues*, pages 87—106. Academic Press, 1994.

[Dub91] E. Dubois. A formal language for the requirements engineering of computer systems. In A. Thayse, editor, *Introducing a Logic Based Approach to AI*, volume 3, pages 357–433. Wiley, 1991.

[DvF93] A. Dardenne, A. vanLamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20:3–50, 1993.

[Eas88] K. Eason. *Information Technology and Organisational Change*. Taylor and Francis, 1988.

[Fea87] M. S. Feather. Language support for the specification and development of composite systems. *ACM Transactions on Programming Languages and Systems*, 9(2):198–234, April 1987.

[FH92] S. Fickas and R. Helm. Knowledge representation and reasoning in the design of composite systems. *IEEE Transactions on Software Engineering*, 18(6):470–482, April 1992.

[Fow97] M. Fowler. *UML Distilled*. Addison Wesley, 1997.

[GF96] O. Gotel and A. Finkelstein. Revisiting requirements production. *Software Engineering Journal*, May 1996.

[Gre93] S. J. Green. Technical report 1993/43: Goal-driven approaches to requirements engineering. Technical report, Department of Computing, Faculty of Engineering, Imperial College, London, 1993. Available at `http://www.doc.ic.ac.uk/research/technicalreports/1993/`.

[Ham90] M. Hammer. Re-engineering work: don't automate, obliterate. In *Harvard Business Review*, pages 104—112, March 1990.

[HB96] K. Holtzblatt and H. Beyer. Contextual design: Principles and practice. In D. Wixon and J. Ramey, editors, *Field Methods Casebook for Software Design*, pages 301–333. John Wiley, 1996.

[HB98] K. Holtzblatt and H. Beyer. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann, 1998.

[HC93] M. Hammer and J. Champy. *Reengineering the Corporation*. Harper Collins, 1993.

[Jac91] M. C. Jackson. Social systems theory and practice: The need for a critical approach. In R. Flood and M. Jackson, editors, *Critical Systems Thinking: Directed Readings*, pages 117–137. John Wiley, 1991.

[Kar90] V. Karakostas. Modelling and maintenance software systems at the teleological level. *Software maintenance: research and Practice*, 2:47—59, 1990.

[Kee92] R. Keeney. *Value Focused Thinking*. Harvard University Press, 1992.

[LK95a] P. Loucopoulos and V. Karakostas. *Systems requirements Engineering*. McGraw-Hill, 1995.

[LK95b] P. Loucopoulos and E. Kavakli. Enterprise modelling and the teleological approach to requirements engineering. *International Journal of Cooperative Information Systems*, 4(1):45—79, 1995.

[McD94] J. McDermid. Requirements analysis: Orthodoxy, fundamentalism and heresy. In M. Jirotka and J. Goguen, editors, *Requirements Engineering: Social and Technical Issues*. Academic Press, 1994.

[Nil71] N. J. Nilsson. *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill Computer Science, 1971.

[Nix93] B. Nixon. Dealing with performance requirements during the development of information systems. In *International Symposium on Requirements Engineering*, pages 42—49. IEEE Computer Society Press, 1993.

[Oul95] M. Ould. *Business Processes: Modelling and Analysis for Reengineering and Improvement*. John Wiley and Sons, Chichester, 1995.

[vL01] A. van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *5th International Symposium on Requirements Engineering*. IEEE Computer Society Press, 2001.

[WBPB95] M. C. Winter, D. H. Brown, and Checkland P. B. A role for soft systems methodology in information systems development. *European Journal of Information Systems*, 4:130–142, 1995.

[WKRG99] B. Warboys, P. Kawalek, I. Robertson, and M. Greenwood. *Business Information Systems: A Process Approach*. McGraw Hill, 1999.