9 Modeling Goals and Reasoning with Them

Colette Rolland and Camille Salinesi

Abstract. The concept of goal has been used in many domains such as management sciences and strategic planning, artificial intelligence and human computer interaction. Recently, goal-driven approaches have been developed and tried out to support requirements engineering activities such as requirements elicitation, specification, validation, modification, structuring and negotiation. This chapter first review various research efforts undertaken in this line of research and presents the state-of-the-art in using goals to engineer requirements. It then presents a particular goal model, the goal/strategy map, and shows that maps can help with facing the challenge of new emerging multi-purposes systems, i.e. systems imposing variability in requirements elaboration and customization in the requirements engineering process.

Keywords: Goal, Goal modeling, Goal specification, Reasoning with goals, Elicitation, Variability, User, Scenario.

9.1 Introduction

Goals have long been recognized to be an essential component involved in the Requirements Engineering (RE) process. In their seminal paper, Ross and Schoman stated "requirements definition must say why a system is needed, based on current and foreseen conditions, which may be internal operations or external market. It must say what a system features will serve and satisfy this context. And it must say how the system is to be constructed" [77]. Typically, the current system is analyzed; problems are pointed out and opportunities are identified; high level strategic goals are elicited and refined to address such problems and meet such opportunities; requirements are then elaborated to meet these goals. Goals are thus the driving force of the requirements engineering process.

Goal-driven approaches have proved to be an effective way to elicit requirements [64, 76] and also to support a systematic exploration of design choices [41, 74, 90] to check requirements completeness [91], to ensure requirements pretraceability [26, 66] and to help in the detection of threats [31] such as conflicts [68] and obstacles [41, 64] and their resolution. The leading role played by goals in the RE process led to a whole stream of research on goal modeling, goal specification/formulation and goal-based reasoning for the multiple aforementioned purposes.

This chapter aims first to provide a state-of-the-art review in the three key topics of goal modeling, goal specification and reasoning with goals. Thereafter, we will discuss a particular goal model, the *goal/strategy map* [73] and show how comprehensive guidelines, drawn from our research and our practical experience,

help to model and specify maps and to reason with them. A special emphasis will be put to demonstrate how goal/strategy maps are well suited to deal with new challenges raised by the emerging conditions of systems development leading to variability in requirements capture and customization in the requirements process. Variability is imposed by the *multi-purpose* nature of software systems of today. These systems must meet the purpose of several organizations and must be adaptable to different usage situations sets of customers. In contrast, earlier software systems were concerned with the purpose of a single organization and of a single set of customers. Variability is defined in software development as the ability of a software system to be changed, customized or configured to a specific context [87]. Therefore, it can be seen that variability affects both goal models, which must make variability explicit, and the process of goal-based reasoning that must help selecting the right variant for the project at hand.

The rest of this chapter is organized in two main sections. Section 2 is an overview of the state-of-the-art in using goals to engineer requirements. Section 3 presents the goal/strategy map model and its contribution to deal with variability requirements.

9.2 State-of-the-Art Review

According to Axel van Lamsweerde [40], RE is "concerned with the identification of goals to be achieved by the envisioned system, the operationalization of such goals into services and constraints, and the assignment of responsibilities of resulting requirements to agents as humans, devices, and software". In this view which is largely shared by the RE community, goals drive the RE process which focuses on goal centric activities such as goal elicitation, goal modeling, goal operationalization and mapping goals onto software objects, events and operations. This section provides an overview of research efforts undertaken in this line. It is organized in three parts. The first one provides the "big picture", the second overviews contributions of goal modeling approaches and the third one discusses their weaknesses.

9.2.1 The Big Picture

This section presents a motivation for goal-driven RE, briefly defines what a goal is and introduces the roles of goals in the RE process and the difficulties encountered in their use.

9.2.2 Motivation for Goal-Based RE Approaches

Goal-driven RE approaches have emerged as a means to overcome the major drawback of traditional approaches, that is, to lead to systems technically good but

unable to respond to the needs of users in an appropriate manner. Indeed, several field studies show that a requirement misunderstanding is a major cause of system failure. For example, a survey of 800 projects undertaken by 350 US companies revealed that one third of the projects were never completed and one half succeeded only partially; poor requirements were identified as the main source of problems [81]. Similarly, a survey over 3800 organizations in 17 European countries shows that most of the perceived problems are related to requirements specification (>50%), and requirements management (50%) [23]. More recently, a 2003 survey of the Meta Group [54] shows even more pessimistic figures attributing 60 to 70% of system failures to poor requirements capture, validation and management. If we want better quality systems to be produced, i.e. systems that meet the requirements of their users, RE needs to explore the objectives of different stakeholders and the activities carried out by them to meet these objectives in order to derive purposeful system requirements. Goal-driven approaches aim at meeting this objective.

The framework of Fig. 9.1 shows that goal-based RE approaches are motivated by establishing an *intentional relationship* between the usage world and the system world [34]. The *usage world* describes the tasks, procedures, interactions etc. performed by agents and how systems are used to do work. It can be looked upon as containing the objectives that are to be met in the organization and achieved by the activities carried out by agents. The *subject world* contains knowledge of the real world domain about which the proposed system has to provide information. Requirements arise from both of these worlds. However, the subject world imposes domain-requirements, which are facts of nature and reflect domain laws, whereas the usage world generates user-defined requirements, which arise from people in the organization and reflect their goals, intentions and wishes.

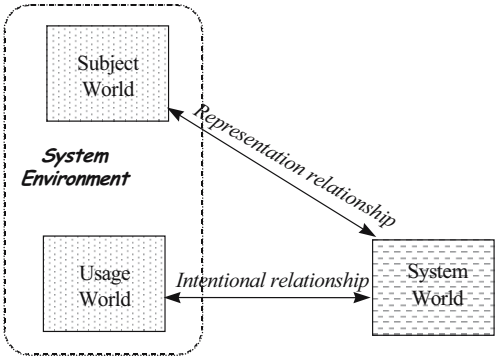

Fig. 9.1 Relationships between the worlds of usage, subject and system

The *system world* is the world of system specifications in which the requirements arising from the other two worlds must be addressed. These three worlds are interrelated as shown in Fig. 9.1. User-defined requirements are captured by

the intentional relationship. Domain-imposed requirements are captured by the representation relationship. Understanding the *intentional relationship* is essential to comprehend the reason why a system should be constructed. The usage world provides the rationale for building a system. The purpose of developing a 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 relationship between the usage world and the system world addresses the issue of the system purpose and relates the system to the goals and objectives of the organization. This relationship explains why the system is developed. Modeling this establishes the conceptual link between the envisaged system and its changing environment.

Goal-driven approaches have been developed to address the semiotic, social link between the usage and the system world with the hope to construct systems that meet the needs of the organization and fulfill their purpose.

9.2.2.1 What Are Goals?

According to Axel van Lamsweerde [43] "a goal corresponds to an objective the system should achieve through the cooperation of agents in the software to be and in the environment". Goals refer to intended or *optative* [32] properties of envisioned system or of its environment. They are expressions of intent and thus declarative with a *prescriptive* nature, by opposition to descriptive statements [32] which describe real facts. For instance, *Transport passengers fast* is a goal whereas *If doors are closed, they are not open* is a descriptive statement. Goals can be formulated at different levels of abstraction ranging from high-level, e.g. strategic results that an enterprise wants to achieve, down to low-level, e.g. technical concerns on precise situations that a system component should help to reach. *Transport passengers safely* is an example of a high level goal whereas *Keep doors closed when moving* is a goal of a lower level of abstraction.

Goals cover different types of concerns, functional and quality (also called non functional). *Functional goals* refer to services that will be provided by the system or its environment whereas *quality goals* refer to qualities of the system behavior in its environment. *Provide cash* is a functional goal whereas *Serve customer quickly* is a quality goal.

Unlike requirements, goals are usually achieved by the cooperation of multiple agents. The goal *Transport passengers safely* requires, for example, the cooperation of multiple agents such as the train transportation system, the software system, the tracking system and the passengers. A goal under the responsibility of a single agent in the software becomes a requirement. One important decision in the RE process is therefore to decide which goals will be automated and which ones will not. Whereas the actual situations met in the system environment (e.g. physical laws, regulations, norms and behaviors, etc) are usually not controlled by the system, it is possible to control the satisfaction of requirements by implementing them into the system. *Maintain doors closed while moving* is a goal leading to a requirement for the system that will ensure its satisfaction whereas *Get in when doors open* is an assumption [15] about agents out of the system control. Such a statement cannot be used as a requirement.

9.2.2.2 Roles of Goals

As a driving force of the requirements engineering process, goals play a number of roles which are introduced in the following.

- *Requirements elicitation*: goal modeling proved to be an effective way to elicit requirements [4, 15, 20, 35, 43, 64, 76]. The pros of goal-based requirements elicitation being that the rationale for developing a system must be found outside the system itself, in the enterprise [49] in which the system shall function.
- *Exploration of design choices*: RE assumes that the envisioned system might function and interact with its environment in many alternative ways. Alternative goal refinement proved helpful in the systematic exploration of system choices [30, 43, 64, 74].
- *Requirements completeness* is a major RE issue. Yue [91] was probably the first to argue that goals provide a criterion for requirements completeness: the requirements specification is complete if the requirements are sufficient to achieve the goal they refine.
- *Requirements traceability*: goals provide a means to ensure requirements pretraceability [26, 60, 66]. They establish a conceptual link between the system and its environment, thus facilitating the propagation of organizational changes into the system functionality. This link provides the rationale for requirements [11, 56, 64, 77, 80] and facilitates the explanation and justification of requirements to the stakeholders.
- *Requirements negotiation*: Stakeholders provide useful and realistic viewpoints about the system to be. Negotiation techniques have been developed to help choosing the prevalent one [9, 29]. Prioritization techniques aim at providing means to compare the different viewpoints on the basis of costs and value [36, 55]. Chapters 7 and 4 respectively provide a more detailed survey of requirements negotiation and prioritization methods.

Conflicts detection and resolution: Multiple viewpoints are inherently associated to conflicts [59] and goals have been recognized to help in the detection of conflicts and their resolution [41, 68, 70, 78].

9.2.3 Contributions of Goal Modeling Approaches

For goals to play the aforementioned roles, a whole stream of research led to contributions on goal modeling, goal formulation and goal-based reasoning that we review in turn.

9.2.3.1 Modeling Goals

Goal modeling is central to RE Goal-driven approaches; its benefit are to support heuristic, qualitative or formal reasoning schemes during the RE process. Goals are modeled by intrinsic features such as types and by links with other goals or other elements in the requirements model. We consider them in turn.

Goal Taxonomies: Goals can be of different types. Several classification schemes have been proposed in the literature. Functional versus non-functional is the first one. Functional goals underlie services that the system is expected to deliver whereas non-functional goals refer to expected system qualities such as security, safety, performance, usability, flexibility, customizability, interoperability, and so forth. A rich taxonomy for non-functional goals can be found in [12]. Another distinction often made in the literature is between *soft goals*, whose satisfaction cannot be established in a clear-cut sense [57], and *hard goals* whose satisfaction can be established through verification techniques [7, 11, 16]. Soft goals are especially useful for comparing alternative goal refinements and choosing one that contributes the "best" to them.

Another classification axis is based on types of temporal behavior prescribed by the goal. In [15], achieving (respectively cease) goals generates system behaviors; maintaining (respectively avoid) goals restricts behaviors; optimizing goals compares behaviors to favor those, which better ensure some soft target property. In a similar way, [82] proposes a classification according to desired system states (e.g., positive, negative, alternative, feedback, or exception-repair) and to goal level (e.g., policy level, functional level, domain level). In [6] Antòn makes a distinction between objective goals that refer to objects in the system, and adverbial goals, that refer to ways of achieving objective goals. Goal types and taxonomies are used to formulate a goal [2, 22, 57, 76] and to define heuristics for goal acquisition, goal refinement, requirements derivation, and semi-formal consistency/completeness checking [2, 5, 12, 15, 82].

Goal Links: Many different types of relationships among goals have been introduced in the literature. They can be classified in two categories to relate goals: (1) to each other and (2) with other elements of requirements models. We consider them in turn in the next sub-sections. Chapter 5 of this book deals with similar expressions.

a) Goal Links Among Goals: The most common form of a goal model is an AND/OR graph. AND/OR relationships [11, 15, 50, 58, 76] inspired from AND/OR graphs in Artificial Intelligence are used to capture goal decomposition into more operational goals and alternative goals, respectively. In the former, all the sub-goals must be satisfied for the parent goal to be achieved, whereas in the latter if one of the alternative goals is achieved, then the parent goal is satisfied. For example, in a book lending system, the goal *Satisfy borrower request* is *ANDed* (has an AND relationship) with *Satisfy Bibliography request, Satisfy book request* and *Provide long borrowing period*. These three goals are sub-goals of the former that will be satisfied if its sub-goals are themselves satisfied. *Maintain as many copies as needed* and *Maintain regular availability* are alternatives to satisfy the goal *Satisfy customer request*. The former is *ORed* (has an OR relationship) with the latter and will be satisfied if one of the two alternative goals is satisfied.

In [12, 57, 58], the inter-goal relationship is extended to support the capture of negative/positive influence between goals. A sub-goal is said to contribute partially to its parent goal. This leads to the notion of goal *satisfycing* instead of goal

satisfaction. For example, *Ensure confidentiality of accounts* and *Ensure security of accounts* are ANDed to *Secure accounts*. Both contribute positively to satisfycing the parent goal *Secure accounts*. By opposition to goal satisfaction, which can be verified quantitatively, using some criterion [69], goal satisfycing cannot be established in a clear-cut sense. Goal satisfaction expressed in AND/OR graphs of hard goals is referred to as the *quantitative framework* whereas goal satisfycing expressed with soft goals is part of the so-called *qualitative framework*. The "motivates" and "hinders" relationships among goals in [11] are similar in the sense that they capture positive/negative influences among goals.

In [76], goal-scenario pairs (called requirement chunks, RC) can be assembled together through *composition, alternative and refinement* relationships. The first two lead to AND and OR structures of RCs whereas the last leads to the organization of the collection of RCs as a hierarchy of chunks of different granularity. *AND relationships* among RCs link complementary chunks in the sense that every one requires the others to define a completely functioning system. RCs linked through *OR relationships* represent alternative ways of fulfilling the same goal. RCs linked through a *refinement relationship* are at different levels of abstraction. The goal *Fill the ATM with cash* is an example of ANDed goal to *Withdraw cash from the ATM* whereas *Withdraw cash from the ATM with two invalid code capture* is ORed to it. Finally *Check the card validity* is linked to the goal *Withdraw cash from the ATM* by a refinement relationship.

Conflict relationships are another kind of relationship among goals. These relationships have been introduced [11][15][59][21] to capture the fact that one goal might prevent the other from being satisfied. For example, in the book lending system considered above, *Provide long borrowing period* which is a sub-goal of *Satisfy borrower request* in the AND/OR graph has a conflict relationship with the alternative goal *Maintain regular availability* of the parent goal *Satisfy customer request* in the same goal graph.

b) Goal Links with Other Elements of Requirements Models: In addition to inter-goal relationships, goals are also related to other elements of requirements models. In his keynote talk [37], Lamsweerde introduced the magic RE triangle as composed of goal, scenario and agent. Obviously goals have privileged relationships with the two other concepts of scenario and agent. Many authors suggest combining goals and *scenarios* [2, 13, 28, 35, 38, 46, 62, 85]. This is understandable because scenarios and goals complement each other. Goals are declarative whereas scenarios are procedural. Intentions are made explicit by goals whereas they are implicit in scenarios. Goals are abstract whereas scenarios are concrete. Combining goals and scenarios can be therefore, seen as a way to mitigate limitations that each concept has when used in isolation. Potts [62] for example, says that it is "unwise to apply goal based requirements methods in isolation" and suggests complementing them with scenarios. This combination has been used mainly, to make goals concrete: scenarios can be interpreted as containing information on how goals can be achieved. In [14, 33, 46, 61], a goal is considered as a contextual property of a use case [33] i.e. a property that relates the scenario to its organizational context. Therefore, goals play a documenting role only. [13] goes

beyond this view and suggests to use goals to structure use cases by connecting every action in a scenario to a goal assigned to an actor. In this sense a scenario is discovered each time a goal is. Clearly, all these views suggest a unidirectional relationship between goals and scenarios. [76] further extends this view by suggesting a "bi-directional relationship between goals and scenarios". In the forward direction from goal to scenario, the scenario represents a possible behavior of the system to achieve the goal, and therefore, scenarios help make the goal concrete and detect unrealistic goals. In the backward direction, from scenario to goal, the relationship is used to discover new goals using mining techniques. As the scenario represents a concrete, realistic behavior of the system to be, the goals inferred from it should themselves be realistic ones.

As mentioned before, goal satisfaction requires cooperation among agents. Relationships with *agents* have been emphasized in [89, 90] where a goal is the object of the dependency between two agents. Such type of link is introduced in other models as well [15, 42, 47] to capture who is responsible of a goal. Aside from the golden relationships with scenarios and agents, goals might have links with other concepts of requirements models. For example, as a logical termination of the AND/OR decomposition, goals link to *operations* which operationalize them [2, 15, 35, 38]. Relationships between goals and system *objects* have been studied in [45] and are for instance, inherently part of the KAOS model [15, 42]. In [11] goals are related to a number of concepts such as *problem*, *opportunity* and *threat* with the aim to better understand the context of a goal. Finally the interesting idea of *obstacle* introduced by [62] leads to obstructions and resolution relationships among goals and obstacles [41, 85].

9.2.3.2 Formulating Goals: Goal formulation is necessary to document the goal model and to support some form of reasoning. Goal formulation can be informal, semi-formal or formal. Goal statements are often texts in natural language [7, 13] and may be supplemented as suggested by [92] with an informal specification to make precise what the goal name designates.

The motivation for semi-formal or formal goal expressions is to support some form of automatic analysis. Typical *semi-formal formulations* use some goal taxonomy and associate the goal name to a predefined type [2, 15]. This helps clarifying the meaning of the goal. For instance, in [57] a non-functional goal can be specified. *Accuracy[account.balance]* is an example of such a goal formulation. Similarly, in Elektra [22], goals for change are pre-fixed by one of the seven types of change: *Maintain, Cease, Improve, Add, Introduce, Extend, Adopt* and *Replace*. Graphical notations [12][57][43] can be used in addition to a textual formulation. L'Ecritoire [76] proposes to formulate each goal as a clause with a main verb and several parameters, where each parameter plays a different role with respect to the verb. For example in the goal statement *Withdraw* $_{verb}$ $(cash)_{target}$ $(from\ ATM)_{means}$, *Withdraw* is the main verb, *cash* is the parameter target of the goal, and *from ATM* is a parameter describing the means by which the goal is achieved. The linguistic approach of Fillmore's Case grammar [24], and its extension [19] was used to define goal parameters [65]. Each type of parameter corresponds to a case and plays a different role with respect to the verb, e.g. target entities affected by the goal,

means and manner to achieve the goal, beneficiary agent of the goal achievement, destination of a communication goal, source entities needed for goal achievement etc.

Formal specifications of goals like in Kaos [15][43] require a higher effort but yield more powerful reasoning. Achieve [BookRequestSatisfied]: ($\forall$bor: Borrower, b: Book, lib: Library) Requesting (bor, b) $\wedge$ b.subject $\in$ lib.coverageArea $\Rightarrow \Diamond$ ($\exists$ bc:BookCopy) (Copy(bc, b) $\wedge$Borrowing(bor, bc)) is an example of such formal specification.

9.2.3.3 Reasoning with Goals: The ultimate purpose of goal modeling is to support some form of goal reasoning for RE sub-processes such as requirements elicitation, consistency and completeness checking, obstacle discovery, conflict resolution and so forth. We consider some of these in the following.

a) Eliciting Goals by Reuse: Although goals can sometimes be spontaneously expressed by stakeholders and therefore available to requirements engineers at early phases of the requirements process, most goals are implicit. Therefore, eliciting goals is not always an easy task, and reasoning techniques can be usefully employed for better performance. *Reuse techniques* are some of these. Chap. 2 is devoted to elicitation problems. For example, Massonet [53] proposes to retrieve goals that have semantically and structurally similar specifications in a repository of reusable specification components, and then transpose the specifications found according to the matching that emerged from the retrieval process. An attempt to retrieve cases from a repository of process cases was developed in [44]. The software tool captures traces of RE processes using the NATURE contextual model [44] and develops a case-based technique to retrieve process cases similar to the situation at hand.

b) Eliciting Goals from Scenarios: A goal inductive elicitation technique based on the analysis of conceptualized scenarios is proposed in [76]. Scenarios can be conceptualized owing to powerful analysis and transformation linguistic techniques based on a Case Grammar inspired by Fillmore's Case Grammar [19, 24]. The pay-off of the scenario conceptualization process is the ability to perform powerful induction on conceptualized scenarios. In [38], a similar approach is developed that takes scenarios as examples and counter examples of the intended system behavior and generates goals that cover positive scenarios and exclude the negative ones. [5] takes similar position to derive goals from use-case specifications.

c) Eliciting Goals by Refinement: Many approaches suggest formulating goals at different levels of abstraction. By essence, goal centric approaches aim to help in the move from strategic concerns and high level goals to technical concerns and low abstraction level goals. Therefore, it is natural for approaches to identify different levels of goal abstraction where high level goals represent business objectives and are refined in system goals [2, 3] or system constraints [41]. Inspired by

cognitive engineering, some Goal-driven RE approaches deal with means-end hierarchy abstractions, where each hierarchical level represents a different model of the same system. The information at any level acts as a goal (the end) with respect to the model at the next lower level (the means) [48, 67, 88]. In [76] the refinement strategy helps discovering goals at a lower level of abstraction. This is a way to support goal decomposition. Another obvious technique to perform refinement is to decompose it by asking the HOW question [39]. Other decomposition based goal elicitation heuristics have been developed in [50] and [47].

d) Obstacle Driven Elaboration: Goal models seem to be powerful instruments to perform hazard reasoning. Several RE approaches have already been developed to deal with obstacles and conflicts [4, 31, 41]. Both concepts relate to the goals that users have in mind when they use the facilities offered by software systems. An obstacle is defined as a phenomenon that occurs in the system and/or its environment and obstructs the achievement of the goal [4, 41]. A conflict is when the achievement of two different goals obstructs each other [21, 68]. A similar principle is used to build misuse case descriptions. A *misuse case* is as a use case described from the point of view of a hostile actor. The goal of this actor is to use the system functions for a different purpose than the one initially intended [1, 79].

e) Conflict Resolution: Reasoning with goals can also help to resolve conflicts among stakeholders. A conflict is when the achievement of two different goals obstructs each other. [59, 68, 78] explain how conflicts arise from multiple viewpoints and concerns. Various forms of conflict have also been studied in [17]. Ivankina [31], and Sutcliffe [83, 84], generalize the notions of obstacle, conflict and other system menace into the notion of threat because they all correspond to the partial or total hindering of one or several system goals.

9.2.4 Weaknesses of Goal-Driven Approaches

Despite their contributions to the performance of a number of RE activities, several authors [39][2][28] also acknowledge the fact that dealing with goals is not an easy task. This sub-section discusses weaknesses of goal driven approaches.

- *Mitigating goal abstractness*: Our own experience in several domains such as air traffic control, electricity supply, human resource management, tool set development is that it is difficult for domain experts to deal with the abstract concept of a goal [75]. Scenario authoring is one of the rare ways used in goal driven approaches to make a goal more concrete. More mechanisms are needed to mitigate the abstract nature of a goal.
- *Finding the right goal*: It is often assumed that systems are constructed with some goals in mind [18]. However, practical experiences show that goals are not given and therefore, the question of where they originate from [2] acquires importance. In addition, enterprise goals, which initiate the goal discovery process, do not reflect the actual situation, but an idealized one. Therefore, pro-

ceeding from spurious goals may lead to ineffective requirements [63]. Thus, finding the right goal is rarely an easy task and more support is needed.

- *Removing goal fuzziness*: The initial goal statement is usually rather imprecise and sketchy and can be interpreted in many ways. The exact meaning of the goal gets clearer and clearer as the elicitation process proceeds. However, experience shows [72] that it is best to make a precise, formal statement of the goal as early as possible in the RE process and that the informal goal statement must be brought into a form that is conducive to performing goal analysis. Goal driven approaches must better support goal formulation avoiding nevertheless the burden of formal languages.
- *Supporting goal operationalization*: Additionally, it has been shown that the application of goal reduction methods to discover the components goals of a goal, is not as straight-forward as literature suggests [15][7]. Our own experience in the F3 [11] and ELEKTRA [75] projects confirms this. It is thus evident that help is needed to achieve meaningful goal modeling.
- *Guiding alternative goals discovery*: Finding alternative goals to a parent goal is crucial for the envisionment of the future system and therefore, crucial to RE. However, experience shows that the process is manual, adhoc and unsatisfactory. This is similar to observations made in the discovery of use case variants [13]. Providing automated support is needed to facilitate the discovery of a large number of alternative designs as an exhaustive generation of alternatives is very difficult to practice manually.

9.3 Goal/Strategy Maps

In this section, we discuss the case of particular type of goal model, the goal/strategy map. We first justify the move from traditional AND/OR goal models to goal/strategy maps as a response to the challenge posed by new multi-purpose emerging systems and by the need to swerve from goal modeling to model goal achievement through strategies to fulfill goals. We introduce the concept of map, illustrate it with an ERP system example and discuss how the model meets the aforementioned challenge. Thereby we consider the customization process implied by multi-purpose systems and discuss the way it can be handled with maps.

9.3.1 Facing the Multi-Purpose System Challenge with Maps

9.3.1.1 Motivations for Maps

Goal modeling approaches have been conceived with the traditional software system life cycle in mind: high strategic goals are captured to elicit software requirements and build the software functionality that fulfils these requirements. However, in recent years, development “from scratch” became the exception and a new

context in which software systems are developed has emerged. Whereas earlier, a system met the purpose of a single organization and of a single set of customers, a system of today must be conceived in a larger perspective, to meet the purpose of several organizations and to be adaptable to different usage situations/customer sets. The former is typical of an ERP-like development situation whereas the latter is the concern of product-line development [86], [10] and adaptable software [30]. In the software community, this leads to the notion of software variability, which is defined as the ability of a software system to be changed, customized or configured to a specific context [87]. Whereas the software community studies variability as a design problem and concentrates on implementation issues [8], [10], [86], we believe like Halmans [27] that capturing variability at the goal level is essential to meet the multi-purpose nature of new software systems.

Our position is that variability implies a move from systems with a *mono-facetted purpose* to those with a *multi-facetted purpose*. Whereas the former concentrates on goal discovery, the multi-facetted nature of a purpose extends it to consider the many different ways of goal achievement. For example, for the goal Purchase Material, earlier it would be enough to know that an organization achieves this goal by forecasting material need. Thus, Purchase material was mono-facetted: it had exactly one strategy for its achievement. However, in the new context, it is necessary to introduce other strategies as well, say the Reorder Point strategy for purchasing material. Purchase Material now is multi-facetted, it has many strategies for goal achievement. These two strategies, among others, are made available, for example, in the SAP Materials Management module[72].

The foregoing points to the need to balance goal-orientation with the introduction of strategies for goal achievement. This is the essence of goal/strategy maps.

A *goal/strategy map*, or *map* for short, is a graph, with intentions as nodes and strategies as edges. An edge entering a node identifies a strategy that can be used for achieving the intention of the node. The map therefore, shows which intentions can be achieved by which strategies once a preceding intention has been achieved. Evidently, the map is capable of expressing goals and their achievements in a declarative manner.

9.3.1.2 The Map Representation Formalism

In this section we introduce the key concepts of a map and their relationships and bring out their relevance to model multi-facetted purposes. A map provides a representation of a multi-facetted purpose based on a non-deterministic ordering of intentions and strategies. The key concepts of the map and their inter-relationships are shown in the map meta-model of Fig. 9.2, which is drawn using UML notations.

- As shown in Fig. 9.2, a map is composed of several sections. A *section* is an aggregation of two kinds of intentions, *source* and *target*, linked together by a strategy.
- An *intention* is a goal, 'an optative' statement [32] that expresses what is wanted i.e. a state that is expected to be reached or maintained. *Make Room Booking* is an intention to make a reservation for rooms in a hotel. The

achievement of this intention leaves the system in the state, *Booking made.* Each map has two special intentions, *Start* and *Stop*, associated with the initial and final states respectively.

- A *strategy* is an approach, a manner, a means to achieve an intention. Let us assume that bookings can be made *on the Internet*. This is a means of achieving the *Make Room Booking* intention, and is a strategy. *by visiting a travel agency* is another strategy to achieve the same intention.
- A *section* is an aggregation of the source intention, the target intention, and a strategy. As shown in Fig. 9.2 it is a triplet <I_{source}, $I_{target,}$ $S_{source\text{-}target}$>. A section expresses the strategy $S_{source\text{-}target}$ using which, starting from I_{source}, I_{target} can be achieved. The triplet <*Start, Make Room Booking, on the Internet*> is a section; similarly <*Start, Make Room Booking, by visiting a travel agency*> constitutes another section.

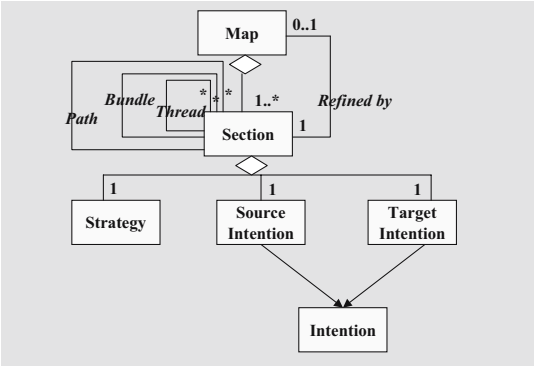

Fig. 9.2 The map meta-model

A section is the basic construct of a map which itself can be seen as an assembly of sections. When a map is used to model a multi-facetted purpose, each of its sections represents a facet. The set of sections models the purpose in its totality and we will see below that the relationships between sections and between a section and a map lead to the representation of the multi-facetted perspective. A facet highlights a consistent and cohesive characteristic of the system that stakeholders want to be implemented in the software system through some functionality. A facet in our terms is close to the notion of feature, which can be defined as a "prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems". We believe that a facet is a useful abstraction to express variability in intentional terms. A map is drawn as a directed graph from *Start* to *Stop*. Intentions are represented as nodes of the graph and strategies as edges between these. The graph is directed because the strategy shows the flow from the source to the target intention (see Fig. 9.5).

- Three kinds of relationships can be defined between sections, namely the thread, path and bundle. These relationships generate multi-thread and multi-path topologies in a map.
- *Thread relationship*: It is possible for a target intention to be achieved from a source intention in many different ways. Each of these ways is expressed as a section in the map. Such a map topology is called a *multi-thread* and the sections participating in the multi-thread are said to be in a thread relationship with one another. Assume that *Accept Payment* is another intention in our example and that it can be achieved in two different ways, *By electronic transfer* or *By credit card*. This leads to a thread relationship between the two sections shown in Fig. 9.3.

It is clear that a thread relationship between two sections regarded as facets represents directly the variability associated to a multi-facetted purpose. Multi-faceting is captured in the different strategies to achieve the common target intention.

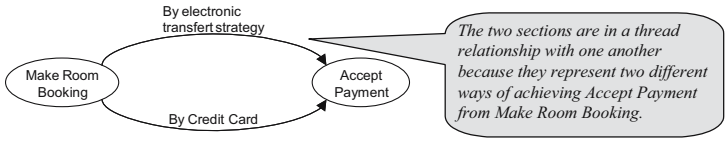

Fig. 9.3 An example of thread relationship

- *Path relationship*: This establishes a precedence/succession relationship between sections. For a section to succeed another, its source intention must be the target intention of the preceding one. For example the two sections <*Start, Make Room Booking, By the Internet Strategy*>, <*Make Room Booking, Accept Payment, By credit card*> form a path.

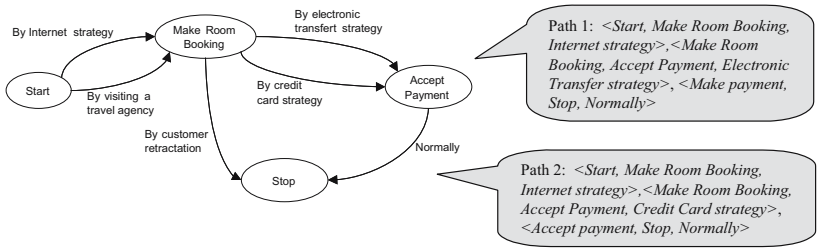

Fig. 9.4 The multi-path of the map *Make Confirmed Booking*

From the point of view of modeling facets, the path introduces a composite facet whereas the section based facet is atomic. Given the thread and the path relationships, an intention can be achieved by several combinations of sections. Such a topology is called a *multi-path*. In general, a map from its *Start* to its *Stop* inten-

tions is a multi-path and may contain multi-threads. Let us assume in our example that it is possible to *Stop* either because a customer retracts from making the booking (*By customer retraction*) or after payment (*Normally*). Fig. 9.4 shows the entire map with the purpose to *Make Confirmed Booking*. This map contains 6 paths from Start to Stop out of which two are highlighted in the Figure.

Clearly, the multi-path topology is yet another way of representing the multi-facetted perspective. Multi-faceting in this case is obtained by combining various sections together to achieve a given intention of the map. Consider for instance the intention Accept payment in Fig. 9.4; there are four paths from Start to achieve it; each of them is a different way to get the intention achieved and in this sense, participates to the multi-faceting. Each path is a composite facet composed of two atomic facets. This can be extended to the full map which can be seen as composed of a number of paths from Start to Stop. This time these paths introduce multi-faceting but to achieve the intention of the map which in our example, is Make Confirmed Booking.

- *Bundle relationship*: Several sections having the same pair $<I_{source}, I_{target}>$ which are mutually exclusive are in a *bundle relationship.* The group of these sections constitutes a *bundle*. Notice that the difference between a thread and bundle relationship is the exclusive OR of sections in the latter versus an OR in the former.
- *Refinement relationship*: The map meta model also shows that a section of a map can be refined as another map through the refinement relationship. The entire refined map then represents the section. Refinement is an abstraction mechanism by which a complex assembly of sections at level i+1 is viewed as a unique section at level i. As a result of refinement, a section at level i is represented by multiple paths & multiple threads at level i+1.

From the point of view of multi-faceting, refinement allows to look to the multi-facetted nature of a facet. It introduces levels in the representation of the multi-facetted purpose which is thus completely modeled through a hierarchy of maps. To sum up:

- The purpose of the system is captured in a *hierarchy of maps*. The intention associated to the root map is the highest level statement about the purpose. Using the refinement mechanism, each section of the root map can be refined as a map and the recursive application of this mechanism results in a map hierarchy. At successive levels of the hierarchy the purpose stated initially as the intention of the root map is further refined.
- At any given level of the hierarchy, the multi-facetted dimension is based on multi-thread and multi-path topologies. Multi-thread introduces local faceting in the sense that it allows to represent the different ways for achieving an intention directly. Multi-path introduces global faceting by representing different combinations of intentions and strategies to achieve a given map intention. Any path from Start to Stop represents one way of achieving the map intention, therefore the purpose represented in this map.

9.3.1.3 Illustrating Map with the SAP R3 Material Management Map

In this section we show the use of the Map to capture the multi-facetted purpose of a system and take the SAP R/3 Materials Management (MM) module to illustrate this. This module provides automated support for the day-to-day operations of any type of business that entails the consumption of materials. It consists of five key components starting from materials planning (MM-MRP Materials Requirements Planning), through purchasing (MM-PUR Purchasing), managing inventory (MM-IM Inventory Management), managing warehousing (MM-WM Warehouse Management), to invoice verification (MM-IV Invoice Verification). It also includes two support components, MM-IS Information System and MM-EDI Electronic Data Interchange.

In its totality, the MM module can be seen to meet the purpose, *Satisfy Material Need Efficiently.* This is the intention of the root map shown in Fig. 9.5. The map shows that to meet this purpose two intentions have to be achieved, namely *Purchase Materi*al and *Monitor Stock*. These reflect the conventional view of materials management as "procuring raw material and ensuring effectiveness of the logistics pipeline through which materials flow" [72]. Evidently, there is an ordering between these two intentions: stock cannot be monitored unless it has been procured. This is shown in the Figure by the section <*Purchase Material, Monitor Stock, Out-In strategy* >.

.

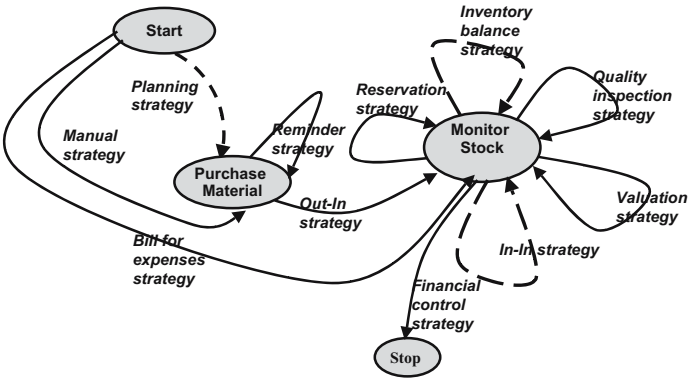

Fig. 9.5 The material management map. Intermittent lines represent bundles.

The map of Fig. 9.5 has 25 paths from *Start* to *Stop*, 5 following the *Bill for expenses strategy*, 10 following the *Planning Strategy*, and 10 following the *Manual strategy*. Thus, the map is able to present a global perspective of the diverse ways of achievement of the main purpose. When a more detailed view is needed, then it becomes necessary to focus more specifically on the multi-facetted nature of each intention found in the "global" map. The detailed view of the intentions contained in Fig. 9.5 is brought out in turn below.

The Multiple Facets of Purchase Material: The multi-facetted nature of *Purchase Material* is shown in Fig. 9.5 by including three strategies for its achievement (a) *Planning strategy*, (b) *Manual strategy* and (c) *Reminder strategy*. The three facets are <*Start, Purchase Material, Planning strategy*>, <*Start, Purchase Material, Manual strategy*> and <*Purchase Material, Purchase Material, Reminder strategy*>. Subsumed in the first facet are two mutually exclusive facets, one that allows purchase to be made when stock falls to the reorder point and the other for purchasing as per the planned material need. These two are captured in a bundle consisting of two strategies not shown in the figure, namely the *Reorder point strategy* and *Forecast based strategy*. The second facet, <*Start, Purchase Material, Manual strategy*>, allows the buyer to manually enter a purchase requisition leading to the generation of the purchase order. The third facet is used to remind the vendor to deliver material when the delivery is not made in due time. The bundled strategies correspond to the SAP functions of MM-MRP Forecast Based Planning and Reorder Point Planning respectively whereas the manual strategy is part of the MM-PUR component. It can be seen that the component structure of SAP does not directly reflect the alternative functionality of achieving the same goal.

The Multiple Facets of Monitor Stock: *Monitor Stock* is the second key intention of the material management map. The intention represents the management goal of ensuring proper posting of procured material and effectiveness of material logistics while maintaining financial propriety. This suggests that Monitor Stock has three classes of facets (a) the procurement/posting class, (b) the logistics class, and (c) the financial class. The facets in each class are as follows:

a) Procurement/Posting Facets
Procurement of material can be done either against a purchase order or without a formal purchase order, directly from the market. In the latter case, material is immediately ready for posting, whereas in the former case, posting is done after delivery is made against the purchase order. Thus, we have two facets of this class:

- Posting of material delivery against a purchase order
- Posting of material procured through direct purchase

These correspond in the map of Fig. 9.5 to the *Out-in strategy* and *Bill for expenses strategy*, respectively. In SAP, the facet represented by the section <*Purchase Material, Monitor Stock, Out-In strategy*> is covered by functions of the MM-IM and MM-WM components whereas <*Start, Monitor Stock, Bill for expenses strategy*> is a function of MM-IV, the Invoice Verification component.

The facet <*Purchase Material*, *Monitor Stock, Out-In strategy*> is, in fact, a compound one. It represents the variety of ways in which compliance of delivered material with the purchase order can be ensured and material posting made. Therefore, its refinement reveals a complex assembly of facets that can be represented through a map at a lower level. This refinement is shown in Fig. 9.6. Since <*Purchase Material*, *Monitor Stock, Out-In strategy*> does not permit stock posting unless material delivery complies with the purchase order, its refinement contains

an ordering of the two intentions, *Accept Delivery* and *Enter Goods in Stock*. The former has four facets, one for the case where delivery is strictly according to the purchase order and three facets that allow delivery to be accepted within specified tolerances from that in the purchase order. The four facets are as follows:

- The delivery complies with the purchase order
- Reconciliation against the purchase order has to be done
- Reconciliation between the different units used by the supplier and the receiver has to be done
- Reconciliation of under/over delivery has to be done

These correspond in Fig. 9.6 to the four multi-threads identified by the strategies Okay strategy, Reconciliation by PO recovery, Reconciliation of unit difference, and Reconciliation of under/over delivery. The nature of the three Reconciliation facets is such that one or more can be simultaneously used. Therefore, these strategies do not form a bundle but are each represented as a thread.

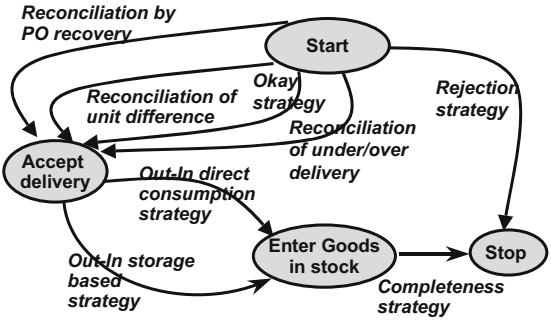

Fig. 9.6 Refinement of <*Purchase Material*, *Monitor Stock, Out-In strategy*>

Now consider the intention *Enter Goods in Stock*. This displays two facets for entering goods in stock (a) when delivery is made directly to the consumption location and (b) when delivered goods are stored in a warehouse. As shown in Fig. 9.6, these two ways of achieving *Enter Goods in Stock* correspond to the two strategies, *Out-In direct consumption* and *Out-In storage based strategy*. The target intention, *Monitor Stock*, of the facet under refinement is achieved in the map when the intention *Stop* is achieved. Evidently, this happens when either the material delivered is rejected and no stock entry is made or when, after entering the accepted delivery in stock, all subsequent housekeeping is done to take into account the consequences of entering goods in stock. These two facets of *Stop* are represented in Fig. 9.6 by *Rejection strategy* and *Completeness strategy* respectively.

b) Material Logistics Facets

Facets in this class enter the picture only after initial posting of stock has been made by the class of procurement/posting facets of *Monitor Stock*. The interesting question now is about the movement of stock and how this movement is kept track of. That is, *Monitor Stock* has to be repeatedly achieved after each movement

to/from warehouses, to consumption points or for quality inspection. This gives us the three facets:

- Control of material movement to/from warehouses
- On-time transfer of material to consumption points
- Quality control of the material transferred

These correspond in the map of Fig. 9.5 to the *In-In*, *Reservation*, and *Quality inspection strategies*. These strategies have *Monitor Stock* as both their initial as well as their target intentions. This represents the repeated achievement of *Monitor Stock*. Of the three foregoing facets, the first, represented by the section <*Monitor Stock, Monitor Stock, In-In strategy*> needs further explanation. In fact, subsumed in this facet are two mutually exclusive facets of *Monitor Stock*. These correspond to the cases when the stock to be moved spends a long time in transit or when immediate transfer is possible. As before, the section <*Monitor Stock, Monitor Stock, In-In strategy*> is represented as a bundle of two sections having strategies *One-step transfer* and *Two-step transfer*. The former corresponds to immediate transfer and the latter to delayed transfer. In SAP, this bundled section is covered partly by MM-IM and MM-WM and has a relationship with Financial Accounting, Assets Management, and Controlling.

c) Financial Propriety Facets
The third class of facets of *Monitor Stock* deals with financial propriety. Not only must it be ensured that stock on hand is physically verified but also it should be financially valued. Thus, we have two facets in this class

- Physical stock taking of the material
- Valuing the stock for balance sheets

These are represented in the map of Fig. 9.5 by the *Inventory balance* and *Valuation* strategies respectively. As for the material logistics class of facets, these are also concerned with the repeated achievement of *Monitor Stock*. Therefore, both the source and target intentions of these strategies is *Monitor Stock*. The facet corresponding to the <*Monitor Stock, Monitor Stock, Inventory balance strategy*> section subsumes three different ways of physical stock taking: by periodic inventory verification, by continuous verification and by verifying a sample of the total inventory. Any of these three can be mutually exclusively deployed. Therefore, we represent it as a bundle of the three strategies, *periodic*, *continuous* and *sampling* strategies. This bundle is handled by the MM-IM component in SAP.

The facet represented in Fig. 9.5 by the section <*Monitor Stock, Monitor Stock, Valuation strategy*> can itself be treated as a bundle of mutually exclusive facets represented by strategies such as LIFO and FIFO. In SAP, only LIFO valuation is available as a function in MM-IM.

Completing Satisfy Material Need Effectively: The complete fulfillment of *Satisfy Material Need Effectively* requires that the financial aspects of material procurement are properly handled. Thus completion, corresponding to the achievement of *Stop* of Fig. 9.5 is done by the *Financial control strategy* allowing the

flow from *Monitor Stock* to *Stop*. In SAP, this takes the form of the Invoice Verification component, MM-IV. When a multi-facetted product like the SAP MM is to be adopted, then the task of the adoption process is to select the facets of the MM map that are of relevance. This leads us to the issue of the process dimension which we consider in the next section.

9.3.2 Matching Maps to Support Multi-Purpose System Customization

The multi-purpose view of emerging systems that leads to the representation of variability in *product models* has a counterpart on the *process dimension* which implies a change of the traditional RE process. Whereas the latter corresponds merely to a move from an *As-Is* to a *To-Be* model (Fig. 9.7a), the former leads to producing the *To-Be* model by a model-match centered process. As shown in Fig. 9.7b the organizational goals are expressed in the *As-Wished* model. The *Might-Be* model reflects the functional capability of the multi-purpose system (e.g. an ERP) and the *To-Be* model needs to be defined as the best match between the *As-Wished* and the *Might-Be*. This process leads to customizing the *Might-Be* model to tailor it to the organizational requirements expressed in the *As-Wished* model.

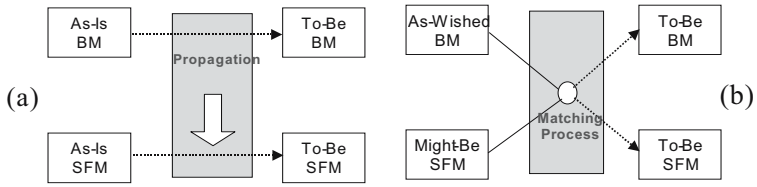

Fig. 9.7 Multi-purpose system customization process (BM stands for Business Models, SFM stands for System Functionality Models)

We believe that maps can help in facing the challenge raised by the customizing activity required in the RE process of multi-purpose systems in two ways: (a) by offering a uniform representation of the involved models, namely the *As-Is*, *As-Wished*, *Might-Be* and *To-Be* and (b) by providing a formalism to model the matching process in a multi-purpose dimension. Our position is that the multi-facetted perspective on product modeling has implications on process modeling as well. First, there cannot be a mismatch between the process modeling paradigm and the product modeling paradigm. Instead, the former must be aligned to the latter. Thus, the process modeling paradigm should be Goal-driven. Secondly, it is unlikely that product variability can be discovered with a monolithic way of working. This implies that the process model should provide many different strategies to achieve the same process goal. The foregoing points to the desirability of the process to be looked upon as a multi-facetted purpose process. This multi-facet aspect implies a process model that has the capability to integrate in it the many strategies found in different methodologies for achieving the same process goal.

For example, to *Elicit a Goal*, different methodologies follow different strategies, top-down, bottom-up, what-if, participative etc. These get integrated in one multi-facetted purpose process model.

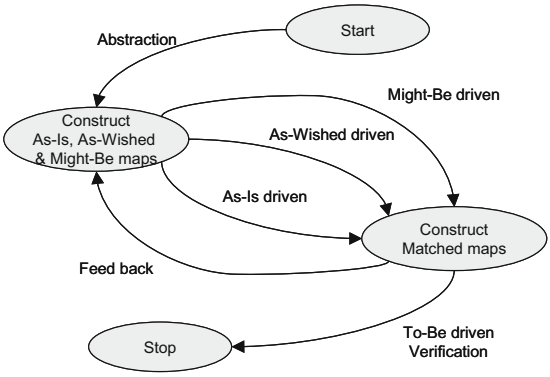

Fig. 9.8 Process model for ERP customization.

This position was confirmed by our experience in different projects where we observed that people have specific expectations and requirements about these process models. First, they are facing an issue and have a goal in mind and would like process models to let them easily situate both and to suggest different alternative paths to achieve the goal and solve the issue. Second, they want freedom and flexibility in their ways of working; one single imposed way-of-working is not acceptable. They expect to learn about the different ways by which each of their goals can be achieved and each issue can be solved. Third, they want advice on how to choose between the different alternative solutions that shall be proposed to solve a given issue. The first two points lead to a multi-purpose driven process model and the third point raises the requirement of a model able to offer guidance in process enactment. Maps can be used to model a methodological process and to capture *process goals* as map nodes and *strategies* to achieve those as edges. For maps to provide guidance we introduced guidelines that can be associated to sections in a process map to guide the selection of process goals as well as to guide strategy selection, situation identification and section achievement.

Fig. 9.8 shows a process model that was developed for an ERP customization project. As the figure shows the process model is represented as a map. The root purpose of this map is *Elicit ERP Installation Requirements*. Achieving the purpose leads to the *Matched-map* which expresses the requirements that the ERP installation shall be met. Many of the intentions/strategies of the *Matched Map* are obtained from the *Might-Be map* (the ERP map) and match the *As-Wished* organizational requirements. Others may not be available in the *ERP map* and will require in-house development. In such a case, the *Matched Map* makes them ex-

plicit. Again, all the intentions and strategies of the *ERP map* may not be included in the *Matched Map*. This corresponds to the ERP functionality that is not matching the requirements in the *As-Wished map*. Thus, the *Matched Map* is the input to the installation process. The multi-facetted nature of the process is shown by the sub-purposes embedded in the map, namely the two main intentions *Construct As-Is, As-Wished, Might-Be maps* and *Construct Matched Map* and the various strategies to achieve them.

There are three ways of achieving it by three different strategic drives, *As-Wished*, *Might-Be* and *As-Is* drives. Each drive considers the intentions and strategies of its corresponding map from *Start* to *Stop* in order to decide if these (a) match the requirements exactly and so must be included in the *Matched map,* (b) need adaptation before their inclusion in the *Matched map*, or (c) are irrelevant.

These three strategies have the same initial and target intentions showing that the target intention can be achieved in a non-deterministic way. This reflects the possibility that different organizations may interleave these strategies in different combinations thereby following different processes to *Construct Matched Map*. Findings from our experience are summed up as follows:

1. If the context is that of a well-defined business requirements to which the system should fit, and in-house development is not a problem, then the As-Wished driven matching strategy can be used.
2. If on the contrary, the system is less likely to change than the business (e.g. because customizing the system has become too expensive [72], or if the system customization is an opportunity to change the business (e.g. because it allows to generalize its associated best practice in the business) then the matching process should be driven by the system. This is what the Might-Be driven strategy proposes.
3. If it is particularly important to preserve the functionality provided by the existing system in the To-Be system functionality model, then an As-Is driven matching is required. We encountered such functional non regression requirements when we studied the introduction of software components for selling electricity in the PPC company at the occasion of European electricity market deregulation [71].

Construct As-Is, Might-Be, As-Wished maps is also multi-facetted. It can be achieved in two ways, by the *Abstraction strategy* or the *Feedback strategy*. The latter has *Construct Matched Map* as its source intention and allows an incremental achievement of *Construct As-Is, Might-Be, As-Wished maps*. This extends to *As-Is* and *ERP maps* the view of Anthony Finkelstein and colleagues [25] that starting with complete requirements specification is not always needed in software package requirements engineering. Finally, the *Stop* intention achieves completion of *Elicit ERP Installation Requirements* through the *To-Be driven verification strategy* that verifies the accuracy of the *Matched Map*.

9.4 Conclusion

Goal-driven requirements engineering are intended to provide the rationale of the system to be. Beyond this objective, we have seen that there are some other advantages:

- Goals bridge the gap between organizational strategies and system requirements thus providing a conceptual link between the system and its organizational context
- Goal decomposition graphs provide the pre-traceability between high level strategic concerns and low level technical constraints; therefore facilitating the propagation of business changes onto system features
- ORed goals introduce explicitly design choices that can be discussed, negotiated and decided upon
- AND links among goals support the refinement of high level goals onto lower level goals till operationalizable goals are found and associated to system requirements
- Powerful goal elicitation techniques facilitate the discovery of goal and requirements;
- Relationships between goals and concepts such as objects, events, operations etc. traditionally used in conceptual design facilitates the mapping of goal graphs onto design specification

We have also discussed the fact that goal driven RE approaches suffer from a number of weaknesses partly due to the nature of the concept of a goal and partly to the lack of modeling and support of the goal driven RE process. The belief of the authors is that goal-driven approaches are now facing the challenge of forthcoming multi-purpose systems, i.e. systems that incorporate variability in the functionality they provide and will be able to self adapt to the situation at hand. The goal/strategy maps have been introduced and discussed as an example of goal model that has been conceived to meet the aforementioned challenge.

References

1. Alexander I (2002) Initial industrial experience of misuse cases in trade-off analysis. In: Proceedings of IEEE Joint International Requirements Engineering Conference, 9-13 September, Essen, pp.61−68
2. Antòn AI, Potts C (1998) The use of goals to surface requirements for evolving systems. In: Proceedings of International Conference on Software Engineering (ICSE'98), Kyoto, Japan, pp.157−166
3. Antòn AI, Earp JB, Potts C, Alspaugh TA (2001) The role of policy and stakeholder privacy values in requirements engineering. In: Proceedings of IEEE 5th International Symposium on Requirements Engineering (RE'01), Toronto, Canada, pp.138−145
4. Antòn AI, Potts C, Takahanshi K (1994) Inquiry based requirements analysis. IEEE Software 11(2): 21−32

5. Antòn AI, Carter R, Dagnino A, Dempster J, Siege DF (2001) Deriving goals from a use-case based requirements specification. Requirements Engineering Journal, 6: 63−73
6. Antòn AI, McCracken WM, Potts C (1994) Goal decomposition and scenario analysis in business process reengineering. CAISE'94, LNCS 811, Springer-Verlag, pp.94−104
7. Antòn AI (1996) Goal based requirements analysis. In: Proceedings of 2nd International Conference on Requirements Engineering ICRE'96, pp.136−144
8. Bachmann F, Bass L (2001) Managing variability in software architecture. ACM SIGSOFT Symposium on Software Reusability (SSR'01), pp.126−132
9. Boehm B, Bose P, Horowitz E, Ming-June L (1994) Software requirements as negotiated win conditions. In: Proceedings of 1st International Conference on Requirements Engineering, USA, pp.74−83
10. Bosch J, Florijn G, Greefhorst D, Kuusela J, Obbink JH, Pohl K (2001) Variability issues in software product lines. In: Proceedings of 4th International Workshop on Product Family Engineering (PEE-4), Bilbao, Spain, pp.22−37
11. Bubenko J, Rolland C, Loucopoulos P, de Antònellis V (1994) Facilitating 'fuzzy to formal' requirements modelling. In: Proceedings of IEEE 1st Conference on Requirements Engineering, ICRE'94 pp.154−158
12. Chung KL, Nixon BA, Yu E, Mylopoulos J (1999) Non- functional requirements in software engineering. The Kluwer international series in software engineering. 1st edition, Kluwer Academic Publishers
13. Cockburn A (1995) Structuring use cases with goals. Technical report. Human and Technology, HaT.Technical Report.1995.01, Accessed on 3rd December 2004. http://alistair.cockburn.us/crystal/articles/sucwg/structuringucswithgoals.htm.
14. Dano B, Briand H, Barbier F (1997) A use case driven requirements engineering process. Journal of Requirements Engineering, Springer-Verlag, 2(2): 79−91
15. Dardenne A, Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Science of Computer Programming, Elsevier.20: 3−50
16. Darimont R, Lamsweerde A. (1996) Formal refinement patterns for goal-driven requirements elaboration. In: Proceedings of 4th ACM SIGSOFT Symposium on the Foundations of Software Engineering, San Francisco, pp.179−190
17. Darimont R, Lamsweerde A, Letier E (1998) Managing conflicts in goal-driven requirements engineering. IEEE Transactions on Software Engineering, 24(11): 908−926
18. Davis AM (1993) Software requirements objects, functions and states. Prentice Hall, UK
19. Dik SC (1989) The theory of functional grammar. Part 1: The structure of the clause. Functional Grammar Series, Fories Publications Dordrecht, Holland
20. Dubois E, Yu E, Pettot M (1998) From early to late formal requirements: a process-control case study. In: Proceedings of 9th International Workshop on software Specification and design. IEEE CS Press, pp. 34−42
21. Easterbrook SM, Finkelstein ACW, Kramer J, Nuseibeh BA (1994) Coordinating conflicting view points by managing inconsistency. Workshop on Conflict Management in Design, International Conference on Artificial Intelligence in Design, Lausanne, Switzerland, pp. 15−18.
22. ELEKTRA consortium (1997) Electrical enterprise knowledge for transforming applications. ELEKTRA Project Reports No 22927
23. European Software Institute (1996) European user survey analysis. Report USV_EUR 2.1, ESPITI Project

24. Fillmore C (1968) The case for case. In: Universals in linguistic theory, Bach E, Harms RT (Eds.), Holt, Rinehart & Winston New York, pp.1−90
25. Finkelstein A, Spanoudakis G, Ryan M (1996) Software package requirements and procurement. In: Proceedings of 8th International workshop on Software Specification and Design, IEEE Computer Society Press, Washington, DC, pp.141−145
26. Gotel OCZ, Finkelstein ACW (1994) Modelling the contribution structure underlying requirements. In: Proceedings of 1st International Workshop on Requirements Engineering: Foundations of Software Quality, Utrecht, The Netherlands, pp. 71−81
27. Halmans J, Pohl K, (2003) Communicating the variability of a software product family to customers. Journal of Software and System Modeling, 2(1): 15−36
28. Haumer P, Pohl K, Weidenhaupt K (1998) Requirements elicitation and validation with real world scenes. IEEE Transactions on Software Engineering, Special Issue on Scenario Management, Jarke M, Kurki-Suonio R (Eds.), 24(12): 11036−1054
29. Hoh P (2002) Multi-criteria preference analysis for systematic requirements negotiation. In: Proceedings of 26th Annual International Computer Software and Applications Conference, Oxford, England pp.887
30. Hui B, Liaskos S, Mylopoulos J (2003) Requirements analysis for customizable software: A goals-skills-preferences framework. In: Proceedings of IEEE Conference on Requirements Engineering, Monterey Bay, USA, pp.117−126
31. Ivankina E, Salinesi C (2004) An approach to guide requirement elicitation by analyzing the causes and consequences of Threats. In: Proceedings of14th European - Japanese Conference on Information Modelling and Knowledge Bases, Skövde, Sweden
32. Jackson M (1995) Software requirements & specifications – a lexicon of practice. Principles and Prejudices, ACM Press, Addison-Wesley
33. Jacobson I (1995) The use case construct in object-oriented software engineering. In Scenario-Based_Design: Envisioning Work and Technology in System Development, J.M. Carroll (Ed.), pp.309−336.
34. Jarke M, Pohl K (1993) Establishing visions in context: Towards a model of requirements processes. In: Proceedings of 12th International Conference on Information Systems, Orlando (ICIS), Orlando, pp.23−34
35. Kaindl H (2000) A design process based on a model combining scenarios with goals and functions. IEEE Transactions on Systems, Man and Cybernetic, 30(5): 537−551
36. Karlsson J, Olsson S, Ryan K (1997) Improved practical support for large-scale requirements prioritizing. Journal of Requirements Engineering, 2(1): 51−60
37. Lamsweerde A. (2004) Goal-oriented requirements engineering: A roundtrip from research to practice. In: Proceedings of 12th IEEE International Symposium on Requirements Engineering, Kyoto, Japan
38. Lamsweerde A, Willemet L (1998) Inferring declarative requirements specifications from operational scenarios. IEEE Transactions on Software Engineering, Special Issue on Scenario Management 24(12): 1089−1114
39. Lamsweerde A, Darimont R, Massonet P (1995) Goal-directed elaboration of requirements for meeting schedulers: Problems and lessons learnt. In: Proceedings of the 2nd IEEE International Symposium on Requirements Engineering (RE'95), pp.194−203
40. Lamsweerde A (2000) Requirements engineering in the year 2000: A research perspective. In: Proceedings of 22nd International Conference on Software Engineering, (ICSE'2000): Limerick, Ireland, Invited Paper, ACM Press, pp. 5–19

41. Lamsweerde A, Letier E (2000) Handling obstacles in goal-oriented requirements engineering. IEEE Transactions on Software Engineering, Special Issue on Exception Handling, 26(10): 978−1005
42. Lamsweerde A, Dardenne B, Delcourt F (1991) The KAOS project: Knowledge acquisition in automated specification of software. In: Proceedings of AAAI Spring Symposium Series, Stanford University, pp.59−62
43. Lamsweerde A (2001) Goal-oriented requirements engineering: A guided tour. In Proceedings of International Joint Conference on Requirements Engineering, Toronto, IEEE, pp.249−263
44. Le TL (1999) Guidage des processus d'ingénierie des besoins par un approche de réutilisation de cas, Master Thesis, CRI, Université Paris-1, Panthéon Sorbonne
45. Lee SP (1997) Issues in requirements engineering of object-oriented information system: A review. Malaysian Journal of computer Science, 10(2)
46. Leite JCS, Rossi G, Balaguer F, Maiorana A, Kaplan G, Hadad G, Oliveros A (1997) Enhancing a requirements baseline with scenarios. In: Proceedings of 3rd IEEE International Symposium on Requirements Engineering, Antapolis, Maryland, pp.44−53.
47. Letier E (2001) Reasoning about agents in goal-oriented requirements engineering. Ph. D. Thesis, University of Louvain
48. Leveson NG (2000) Intent specifications: an approach to building human-centred specifications. IEEE Transactions on Software Engineering 26: 15−35
49. Loucopoulos P (1994) The f^3 (from fuzzy to formal) view on requirements engineering. Ingénierie des systèmes d'information, 2(6): 639−655
50. Loucopoulos P, Kavakli V, Prekas N (1997) Using the EKD approach, the modelling component. ELEKTRA project internal report, UMIST, Manchester, UK
51. Maiden N, Ncube C (1998) Acquiring COTS software selection requirements. IEEE Software, 15(2): 46−56
52. Maiden N, Kuim H, Ncube C (2002) Rethinking process guidance for software component selection. In: Proceedings of 1st Int. Conf. of Component Based Eng, pp.151−164
53. Massonet P, Lamsweerde A, (1997) Analogical reuse of requirements frameworks. In: Proceedings of 3rd International Symposium on Requirements Engineering, Annapolis, pp.26−37.
54. META Group (2003) Research on requirements realization and relevance, Meta Group report
55. Moisiadis F (2002) The fundamentals of prioritising requirements systems engineering. Systems Engineering, Test & Evaluation Conference, Sydney, Australia, October
56. Mostow J (1985) Towards better models of the design process. AI Magazine, 6: 44−57
57. Mylopoulos J, Chung KL, Nixon BA (1992) Representing and using non- functional requirements: a process-oriented approach. IEEE Transactions on Software Engineering, Special Issue on Knowledge Representation and Reasoning in Software Development, 18(6): 483−497
58. Mylopoulos J, Chung KL, Yu E (1999) From object-oriented to goal-oriented requirements analysis. Communications of the ACM, 42(1): 31−37
59. Nuseibeh B, Kramer J, Finkelstein A (1994) A framework for expressing the relationships between multiple views in requirements specification. IEEE Transactions on Software Engineering, 20: 760−773
60. Pohl K (1996) Process centred requirements engineering, J. Wiley and Sons.

61. Pohl K, Haumer P (1997) Modelling contextual information about scenarios. In: Proceedings of the 3rd International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona, Spain, pp.187−204
62. Potts C (1995) Using schematic scenarios to understand user needs. In: Proceedings of ACM Symposium on Designing interactive Systems: Processes, Practices and Techniques, University of Michigan, USA, pp.247−256
63. Potts C (1997) Fitness for use: The system quality that matters most. In: Proceedings of International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona, pp.15−28
64. Potts C, Takahashi K, Antòn AI (1994) Inquiry-based requirements analysis. IEEE Software 11(2): 21−32
65. Prat N (1997) Goal formalisation and classification for requirements engineering. In: Proceedings of 3rd International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona, Spain, pp.145−156
66. Ramesh B, Powers T, Stubbs C, Edwards M (1995) Implementing requirements traceability: a case study. In: Proceedings of the 2nd Symposium on Requirements Engineering (RE'95), UK, pp.89−95
67. Rasmussen J (1990) Mental models and the control of action in complex environments. In: Mental Models and Human--Computer Interaction, Ackermann D, Tauber MJ (Eds.) North-Holland: Elsevier, pp.41−69
68. Robinson WN, Volcov S (1996) Conflict oriented requirements restructuring, Georgia State University, Atlanta, GA, Technical Paper CIS-96-15, October 8,
69. Robinson WN, (1989) Integrating multiple specifications using domain goals. In: Proceedings of 5th International Workshop on Software Specification and Design, IEEE, pp.219−225
70. Robinson WN, Volkov S (1998) Supporting the negotiation life-cycle. Communications of the ACM, 41(5): 95−102
71. Rolland C (2000) Intention driven component reuse. In: Information Systems Engineering, Brinkkemper S, Lindencrona, E, Solvberg A (Eds.) Springer, pp.197−208
72. Rolland C, Prakash N (2000) Bridging the gap between organizational needs and ERP functionality. Requirements Engineering Journal, 4(1): 180−193
73. Rolland C, Salinesi C, Etien A (2004): Eliciting gaps in requirements change. Requirements Engineering Journal, 9(1): 1−15
74. Rolland C, Grosz G, Kla R (1999) Experience with goal-scenario coupling. In Proceedings of 4th IEEE International Symposium on Requirements Engineering, Limerik, Ireland, pp.74−81
75. Rolland C, Nurcan S, Grosz G (1997) Guiding the participative design process. In Proceedings of Association for Information Systems Americas Conference, Indianapolis, Indiana, pp.922−924
76. Rolland C, Souveyet C, Salinesi C (1998) Guiding goal modelling using scenarios. IEEE Transactions on Software Engineering, Special Issue on Scenario Management, 24(12): 98−27
77. Ross DT, Schoman KE (1977) Structured analysis for requirements definition. IEEE Transactions on Software Engineering, 3(1): 6−15
78. Easterbrook SM (1994) Resolving requirements conflicts with computer-supported negotiation. In Requirements Engineering: Social and Technical Issues, Jirotka M, Goguen J (Eds.) London: Academic Press, pp.41−65

79. Sindre G, Opdahl L (2001) Templates for misuse case description. In Proceedings of 7th International Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ'2001): Interlaken, Switzerland
80. Sommerville I, Sawyer P (1997) Requirements engineering. Worldwide Series in Computer Science, Wiley
81. Standish Group (1995) Chaos, Standish Group Internal Report, www.standishgroup.com/chaos.html
82. Sutcliffe A, Maiden N (1993) Bridging the requirements gap: Policies, goals and domains. In: Proceedings of 7th International Workshop on Software Specification and Design, IEEE Computer Society Press, pp.52−55
83. Sutcliffe A, Minocha S (1999) Analyzing socio-technical system requirements. CREWS project Report 98-37, Accessed on 5th December 2004, http://sunsite.informatik.rwth-aachen.de/CREWS/reports.htm
84. Sutcliffe AG, Galliers J, Minocha S (1999) Human errors and system requirements. In: Proceedings of 4th IEEE International Symposium on Requirements Engineering, Limerick, Ireland, pp.23-30
85. Sutcliffe AG, Maiden N, Minocha S, Darrel M (1998) Supporting scenario-based requirements engineering. IEEE Transactions Software Engineering, 24(12): 1072−1088
86. Svahnberg M, Gurp J, Bosch J (2001) On the notion of variability in software product lines. In: Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA 2001) pp.45−55
87. Van Gurp J (2000) Variability in software systems: The key to software reuse. Licentiate Thesis, University of Groningen, Sweden
88. Vicente KJ, Rasmussen J (1992) Ecological interface design: Theoretical foundations. IEEE Transactions on Systems, Man and Cybernetics, 22(4): 589−606
89. Yu E (1997) Towards modeling and reasoning support for early-phase requirements engineering. In: Proceedings of 3rd IEEE International Symposium on Requirements Engineering -RE97, pp.226−235
90. Yu E (1994) Modeling strategic relationships for process reengineering. Ph.D. Thesis, Department Computer Science, University of Toronto, Canada
91. Yue K (1987) What does it mean to say that a specification is complete? In: Proceedings of 4th International Workshop on Software Specification and Design, Monterey, CA, USA, pp.34−41
92. Zave P, Jackson M (1997) Four dark corners of requirements engineering. ACM Transactions on Software Engineering and Methodology, 6(1): 1−30

Author Biography

Colette Rolland is Professor of Computer Science and head of CRI at University of Paris 1 - Panthéon Sorbonne. Her research interests lie in the areas of information modeling, object-oriented analysis and design, requirements engineering, CASE and CAME tools, change management and enterprise knowledge development. She has supervised 62 PhD theses and was author/co-author of 5 textbooks and of over 170 invited and refereed papers. She also has extensive experience in leading research projects funded by French institutions (CNRS, INRIA, Ministry of Research and Technology) as well as by ESPRIT programs, including

NATURE (N° 6353) - *ESPRIT 3*, ELEKTRA (N° 22927), and CREWS (N° 21903) - *FRAMEWORK 4*.

Dr. Camille Salinesi is a senior lecturer of Computer Science at the University of Paris 1 Panthéon - Sorbonne. His research works deal with Requirements Engineering, Systems Engineering, and Process Engineering. He has published more than 40 refereed papers and organized several conferences (OOIS'98, REP'99, REFSQ'01'02'03, RE'05) in these domains. His recent works showed significant results on the topics of the use of Use Cases, Goals, and Scenarios in Requirements Engineering and about Information System evolution and ERP implementation. Dr Salinesi was involved in several fundamental research projects such as ESPRIT NATURE and CREWS, and was consultant for several national and international companies. Camille Salinesi is a member of the INCOSE and belongs to the RE group of the French association of Systems Engineering.