Chapter 2
Reasoning About Dynamic Aspectual Requirements

Yijun Yu, Xin Peng, and Julio Cesar Sampaio do Prado Leite

Abstract Aspect-oriented requirements modelling separates the early crosscutting concerns as quality requirements such that one can reason about such requirements without cluttering with another. In this chapter, we propose a step further to reason about the dynamic goal models while the separated aspectual requirements are also dynamic. The key to this step is a list of change propagation rules for the goal models such that it is possible to reuse as much previous reasoning results as possible. To demonstrate, we use the Crisis Management System case study to indicate the application of these rules.

2.1 Introduction

Given a highly dynamic environment it is desirable that a software system be able to manage changes *continuously,* without suspension of the execution, whilst maintaining its essential requirements. In order to decide what is needed to change for satisfying the requirements, however, a system shall be aware of its current situation and the context in which it is situated. In support of requirements-driven self-managing systems, requirements reasoning needs to be made as dynamic as possible to take into account the new knowledge learnt during runtime.

We explore the issues of requirements-driven self-managing systems using a representation scheme, the goal-oriented non-functional requirements (NFR) framework [1, 2], that clearly differentiates functional and quality requirements.

Y. Yu (✉)
Centre for Research in Computing, The Open University, Buckinghamshire, UK
e-mail: y.yu@open.ac.uk

X. Peng
School of Computer Science, Fudan University, Shanghai, China

J.C.S. do Prado Leite
Departamento de Informática, PUC-Rio, Rio de Janeiro, Brazil

A. Moreira et al. (eds.), *Aspect-Oriented Requirements Engineering*,
DOI 10.1007/978-3-642-38640-4_2,

The NFR framework separates functions, which are modelled as hard goals (i.e. with crisp/binary satisfaction criteria), from quality requirements, which are modelled as soft goals (i.e. with non-crisp/non-binary satisfaction criteria, and using the term *satisficed* to acknowledge the distinction).

In relation to these classifications, functions of an implementation are composed by AND–OR rules to satisfy those high-level hard goals; and they are weaved through MAKE-BREAK-HELP-HURT contribution rules to satisfy those high-level soft goals. Goals are labelled by both their type (the function or the quality) and their topics (the domains) to which the type is being applied.

Our earlier work [3] introduced the notion of aspectual requirements to modularise the functions for the operationalisation of quality requirements (i.e. *advices* tasks) and to separate them from those functional requirements (i.e. *join-points* tasks) crosscut by the subject domains (i.e. *point-cuts* topics). The weaving algorithms for introducing the advising tasks into the join-points aim to maintain the equivalence of answers to the same deductive question, with or without the aspects: "Given a set of subgoals, are the high-level hard goals satisfied and high-level soft goals satisficed well enough?"

Given a model consisting of static rules and an a priori set of hard/soft goals, one could perform goal-based reasoning algorithms to get the answer. However, as we have just motivated earlier, these rules may no longer be static in real situations: changes to the hard/soft goals and their operationalised tasks, or even the rules, themselves, may change given that the context of a system is highly changeable, and the knowledge of the system requirements (awareness) is also allowed to be changed at runtime.

In this chapter, we aim to provide a theoretical answer to the following research question: "Is it possible to reason about the equivalence between the crosscutting requirements problem and the separated aspectual requirements problem while taking into account the changes to the requirements model as well as to the situations in the context?" The main contributions of this chapter are as follows:

1. Deal with change in the context of aspect-oriented requirements models, i.e. aspectual requirements models.
2. Propose a list of fundamental rules that characterises the equivalence of requirement aspect weaving in dynamic goal models.
3. Demonstrate the application of these rules in the common case study, the Crisis Management System [4].

The remainder of the chapter is organised as follows. Section 2.2 gives more background of goal-based reasoning and goal aspects using an illustrative example. Section 2.3 presents the list of equivalence rules that guarantees the dynamic aspectual requirements do not introduce problems to the dynamic goal reasoning. Section 2.4 presents an example application of the dynamic aspectual requirements. Section 2.5 summarises the limitations in this work, which points to a few future research directions. Section 2.6 concludes and points out a few interesting direction we hope to explore in future.

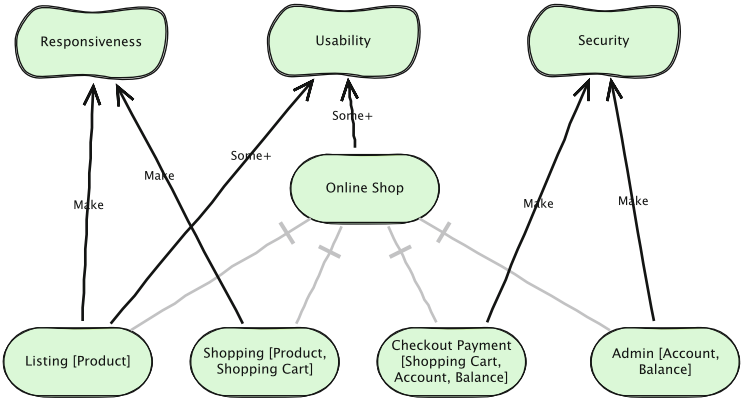

Fig. 2.1 The top-level hard and soft goals of an online shop

2.2 Background

In this section, we use the requirements model of an online shop as the running example to illustrate the problem and the existing solutions.

2.2.1 A Running Example

Figure 2.1 shows a simple goal model with the top-level hard goal "Online shop" decomposed by function into four sub-goals, "Listing [product]", "Shopping [Product, ShoppingCart]", "Checkout Payment [ShoppingCart, Account, Balance]" *and* "Admin [Account, Balance]". The canonical convention we adopted for these goals or subgoals is "type [topics]", named after the type of the primary function of these hard goals, along with the subject domains listed as the topics enclosed by square brackets, separated by commas. Meanwhile, they are contributing to three top-level soft goals indicated by the curly nodes, including "Responsiveness", "Usability" and "Security". As quality requirements, these soft goals do not have an absolute satisfaction, but they do have criteria to judge whether they are sufficiently implemented. Specifically, the "Listing" and "Shopping" hard goals *must* achieve the "responsiveness" because they are frequent operations; the "Checkout payment" and "Admin" hard goals *must* have "Security", because they both may access the customers' accounts and may update their balances; the hard goals "Listing" and "Shopping" may *help* "Usability" through memorising the potential customers' transactions as a shopping cart.

2.2.2 Reasoning with Goals and Contexts

In order to reason about the dynamic goals within changing contexts, we first transform, using the formulae introduced by Zave et al. [5], the dynamic goals' problem into the problem of judging whether (2.1) is equivalent to (2.2) in the following form:

$$\mathrm{E}, \mathrm{S} \Big| = \mathrm{R} \tag{2.1}$$

$$\mathrm{E}, \mathrm{S_F}, \mathrm{S_Q} \Big| = \mathrm{R_F} \wedge \Phi\left(\mathrm{R_Q}\right)$$

$$\mathrm{R} = \mathrm{R_F} \wedge \Phi\left(\mathrm{R_Q}\right)$$

$$\mathrm{S} = \mathrm{S_F}, \mathrm{S_Q} \tag{2.2}$$

$$\mathrm{E}, \mathrm{S_F} \Big| = \mathrm{R_F}$$

Here *E* stands for the environment, *R* stands for the requirements and *S* stands for the system specifications. Zave et al. [5] define the formula (2.1) as the requirements problem, whilst a refinement according to [1] can be formulated into (2.2) whereby S is separated into the functional tasks $\mathrm{S_F}$, and the advising tasks $\mathrm{S_Q}$, and R consists of the conjunction of both functional requirements $\mathrm{R_F}$ and the quality requirements $\mathrm{R_Q}$ and the evaluation of its fully/partially satisficed or not by $\Phi(\mathrm{R_Q})$.[1] Note that the composition of $\mathrm{S_F}$ and $\mathrm{S_Q}$, denoted by the "," separator, can be achieved by aspect weaving [7].

With an extension to the temporal dimension *t*, now we would like to establish the equivalence of (2.1t) and (2.2t):

$$\mathrm{E}\,(\mathrm{t})\,,\,\mathrm{S}\,(\mathrm{t}) \Big| = \mathrm{R}\,(\mathrm{t}) \tag{2.1t}$$

$$\mathrm{E}\,(\mathrm{t})\,,\,\mathrm{S_F}\,(\mathrm{t})\,,\,\mathrm{S_Q}\,(\mathrm{t}) \Big| = \mathrm{R_F}\,(\mathrm{t}) \wedge \Phi(t)\ \left(\mathrm{R_Q}\,(\mathrm{t})\right)$$

$$\mathrm{R}\,(\mathrm{t}) = \mathrm{R_F}\,(\mathrm{t}) \wedge \Phi\left(\mathrm{R_Q}\,(\mathrm{t})\right)$$

$$\mathrm{S}\,(\mathrm{t}) = \mathrm{S_F}\,(\mathrm{t})\,,\,\mathrm{S_Q}\,(\mathrm{t}) \tag{2.2t}$$

$$\mathrm{E}\,(\mathrm{t})\,,\,\mathrm{S_F}\,(\mathrm{t}) \Big| = \mathrm{R_F}\,(\mathrm{t})$$

[1] An extension to Zave et al. has been proposed in Jureta et al. [6] to include the requirements other than the functional/quality, such as attitude, which are treated as part of the context here.

In other words, when at any given time t, the basic requirements problem shall hold. One may interpret the continuous satisfaction of requirements as a desirable self-management quality requirement at the process (meta-) level. With the help of satisfying additional monitoring [8], diagnosis [9] and feedback loop control [10] requirements derived from this process requirement, the systems may guarantee the satisfaction of the core functional requirements [11].

When the formula is focused on the runtime requirements of the software system product, the reasoning of the goal model is rather straightforward. Suppose that the top-level requirements are $R = {R_F}^{\text{Online shop}} \hat{} \text{FS}({R_Q}^{\text{Security}}) \hat{} \text{PS}({R_Q}^{\text{Usability}}) \hat{} \text{FS}({R_Q}^{\text{Responsiveness}})$. Here FS and PS stand for "fully satisficed" and "partially satisficed," respectively, indicating the degree of the perceived quality criteria relating to the soft goals in parenthesis.

Then it is possible to specify $S_F = {S_F}^{Listing}, {S_F}^{Shopping}, {S_F}^{Checkout\ Payment}, {S_F}^{Admin}$ that given the context domains $E = E^{Product}, E^{Shopping\ Cart}, E^{Account}, E^{Balance}$ shall satisfy $R = {R_F}^{Online\ Shop} \hat{} FS({R_Q}^{Responsiveness}) \hat{} PS({R_Q}^{Usability}) \hat{} FS({R_Q}^{Security})$. In other words, the S_Q is required to be specified for each function as ${S_Q}^{Listing}, {S_Q}^{Shopping}$, ${S_Q}^{Checkout\ Payment}, {S_Q}^{Admin}$. Here we use "," to connect these domains, instead of "^" because the composition of them usually involves structural and behaviour semantics beyond simple logic conjunction. Nonetheless, the requirements problem $E, S| = R$ in (2.1) or $E, S_F, S_Q| = R_F \hat{} \Phi(R_Q)$ in (2.2) can thus be refined into conjunctive sub-problems in the same form, with E, S and R substituted by the counterparts of the sub-problems. For examples, each sub-goal can be regarded as a sub-requirement for one of the sub-problems:

$$E^{Product}, S^{Listing} \Big| = {R_F}^{Listing} \wedge FS\left({R_Q}^{Responsiveness}\right) \wedge PS\left({R_Q}^{Usability}\right)$$

$$E^{Product}, E^{ShoppingCart}, S^{Shopping} \Big| = {R_F}^{Shopping} \wedge FS\left({R_Q}^{Responsiveness}\right) \wedge PS\left({R_Q}^{Usability}\right)$$

$$E^{ShoppingCart}, E^{Account}, E^{Balance}, S^{Payment} \Big| = {R_F}^{Payment} \wedge FS\left({R_Q}^{Security}\right)$$

$$E^{Account}, E^{Balance}, S^{Admin} \Big| = {R_F}^{Admin} \wedge FS\left({R_Q}^{Security}\right)$$

Note that here although the functional R_F and quality requirements R_Q are separated, their operationalisations are still to be refined into functional tasks S_F and advice tasks S_Q. In the next subsection, we explain how we represent the functional tasks as functions and advice tasks as aspects.

Now a simple version of the goal reasoning is to establish that ${R_F}^{Listing} \hat{} {R_F}^{Shopping} \hat{} {R_F}^{Payment} \hat{} {R_F}^{Admin} \hat{} FS({R_Q}^{Responsiveness}) \hat{} PS({R_Q}^{Usability}) \hat{} FS({R_Q}^{Security}) = R$. From the known satisfaction result of the requirements on the left-hand side (i.e. ${R_F}^{Listing}, {R_F}^{Shopping}, {R_F}^{Payment}, {R_F}^{Admin}, FS({R_Q}^{Responsiveness}), PS({R_Q}^{Usability})$, $FS({R_Q}^{Security})$) to the satisfaction of the requirements on the right-hand side (i.e. R), the reasoning is called *bottom-up* label propagation [12], whilst the opposite,

reasoning from the known high-level goals in order to find the minimal plan or conjunction of sub-goals, is called *top-down* satisfiability (SAT) solution [13]. In either case, only goals are participating in the computation. Recently, Ali et al. [14] expand the semantics of goal reasoning to those of the context (set of topics) , such that it is also possible to reason about the contextual relationships in logic rules. The formula we presented, although looks simple, reflects more expressive reasoning rules because both the context and the goals can participate in the computation. By turning the missing goals or the missing context into wildcards, the same rule can be used to model all the rules [12–14]. More complete rules for requirements problems can be found in the canonical form [15] that also includes the *justification*.

$$\frac{E(t)\,,\, S(t) \;\Big|\!=\; R_F(t) \;\wedge\; \Phi\left(R_Q(t)\right)}{E(t),\, S(t) \;\Big|\!=\; R(t)}\langle\langle justification\rangle\rangle \tag{2.3}$$

However, to reason about the justifications, a high-order logic or reification is required. For simplicity, in this chapter we only consider reasoning in the first-order predicate logic where the time t is treated as a special term in the predicates.

2.2.3 Reasoning with Aspectual Requirements

According to the systematic process in Yu et al. [3] using the conceptual construct of the V-graph model (see Fig. 2.2), the (hard) goals and soft goals are respectively operationalised into tasks. After the process, the tasks that operationalise the soft goals are separated from those tasks that operationalise the hard goals, as the advice tasks for the aspectual requirements. Moreover, the high-level contributions from the hard goals to the soft goals must be maintained by the low-level tasks when they are weaved together. Once they are separated, it is possible to specify the functional tasks S_F as functions and the advice tasks S_Q as aspects.

For example, the three soft goals are operationalised to three aspectual requirements. They weave their advising tasks (operationalisations) as modifications to the basic functions to guarantee the expected contribution relationship at a higher level. To illustrate that, we expand the high-level goal model Fig. 2.1 with one more level as a more concrete goal model in Fig. 2.3.

The "Listing [Product]" goal can be refined by either of the two alternatives: "Listing [Product, Persistence=Databases]" or "Listing [Product, Persistence=LDAP]", depending on the different mechanisms for persistence. Specifically, these two tasks can be separated into three tasks: one common functional task which does not specify the exact persistence mechanism "Listing [Product, Persistence=?]" and two alternative advice tasks "Database" and "LDAP". According to the documentation of the case study (i.e. http://oscommerce.com), the designer prefers to implement the product lists using LDAP service because

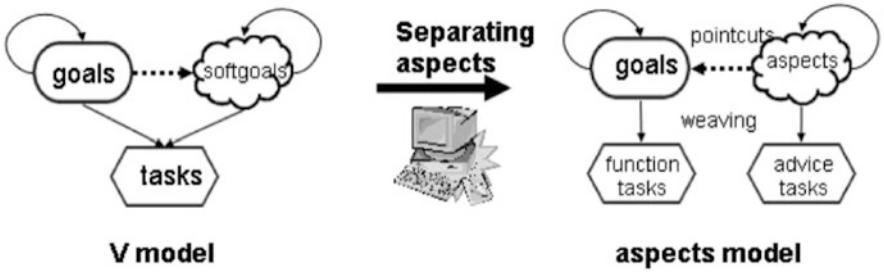

Fig. 2.2 The V-shaped graphs show conceptually how a tasks [1, Fig. 15]

it is more efficient in online shopping scenarios as customers frequently browse and select the products. Although a database solution is more scalable, it is not as responsive as the LDAP-based solutions. Therefore if the goal reasoning (top-down) is applied and $FS(R_Q^{Responsiveness})$ is required (on the right-hand side), one would find "$S^{Listing\ [Product,\ Persistence = LDAP]} = S_F^{Listing[Product,\ Persistence = ?]}, S_Q^{Persistence\ [LDAP]}$" in the plan because the contribution link from "Listing[Persistence=LDAP]" to "Responsiveness" is MAKE, whilst the contribution link from "Listing[Persistence=Database]" to "Responsiveness" is only HELP.

Similarly, the same trade-off needs to be made for the other hard goal "$S_F^{Shopping\ [Produce,\ ShoppingCart,\ Persistence = ?]}$". Although the connections from both the "Database" and "LDAP" tasks to the parent goals are "OR decomposition", once the trade-off decision is made in the reasoning, they can be simplified into "AND decomposition" because no matter which OR sub-goal of "Listing" or "Shopping" is chosen, they all need the "LDAP"-based implementations for the full satisficing of the Responsiveness soft goal.

In addition, "$S_F^{Listing\ [Product,\ Persistence = ?]}$" is refined into a task making use of the "$S_F^{Listing\ [Product,\ Persistence = ?,\ Display = ?]}$" and "$S_Q^{Display\ [StyleBox]}$", which presents user with the consistent look and feel in the whole website. In fact, this *Style Box* design should be applied to every page to be displayed, including both the listing and shopping pages. Therefore it is one of the candidate aspects.

Likewise, for the "Security" soft goal and all the sub-goals that require to *make* it satisfied (i.e. "Payment" and "Admin"), a common advice task "$S_Q^{Autentication\ [Login,\ User,\ Credentials]}$" is to be composed with the functional tasks "$S_F^{Checkout\ Payment\ [Shopping\ Cart,\ Account,\ Balance,\ Authentication = ?]}$" and "$S_F^{Admin[Account,\ Balance,\ Authentication = ?]}$". This makes the "Authentication [Login, User, Credentials]" another requirement aspect.

The third requirement aspect "Display [StyleBox]" has to do with "Usability". Although it is required to be partially satisfied by the "Listing" and "Shopping" sub-goals, having the *Style Box* is common to both and therefore crosscut every place in the program wherever the information is to be displayed as a web page.

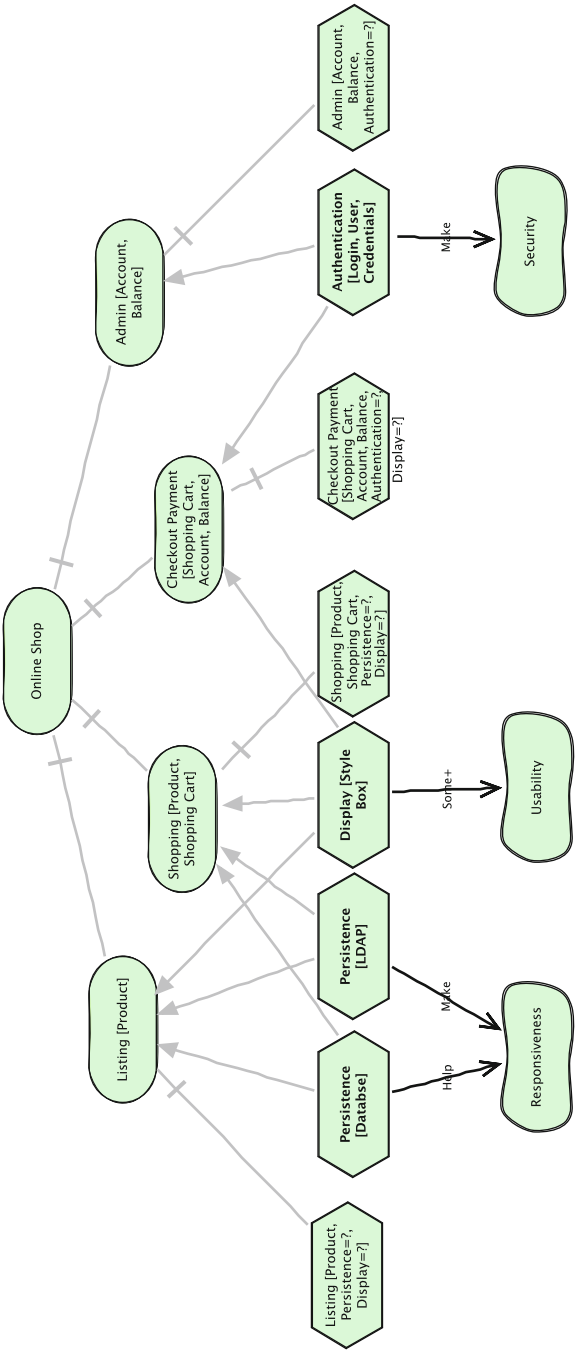

Fig. 2.3 The refined tasks of an online shop (cf. Fig. 2.1). Candidate aspectual requirements and their contribution to the hard and soft goals are highlighted

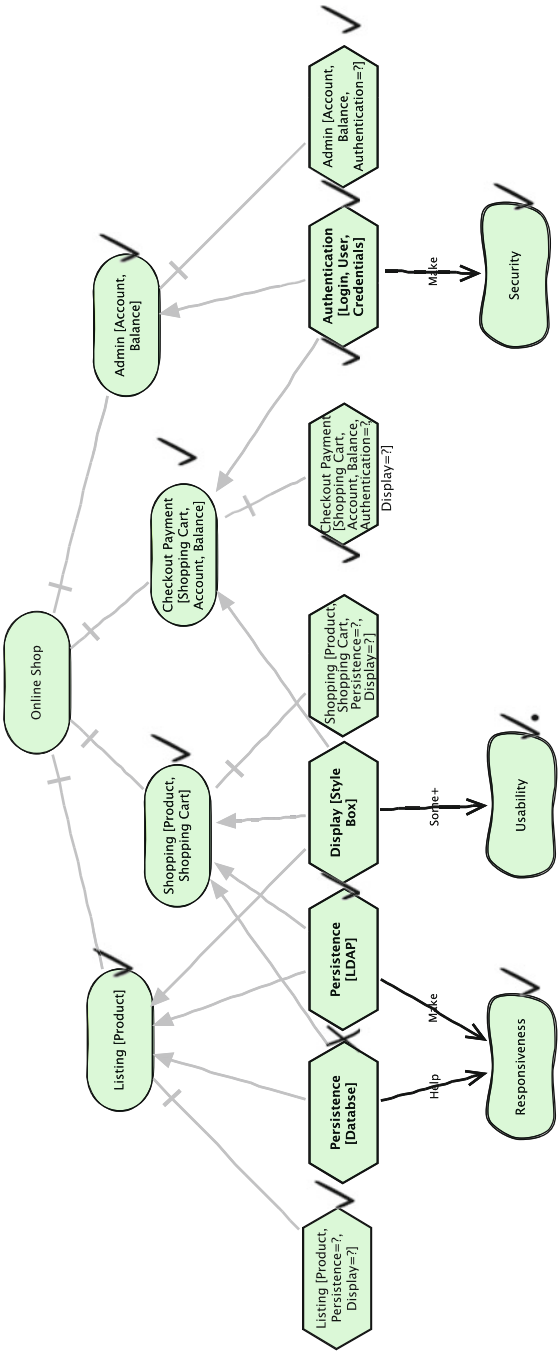

Fig. 2.4 Some reasoning results (cf. Fig. 2.3). In the NFR framework [2], the FS (satisfied/fully satisficed), PS (partially satisficed), CF(conflict), PD (partially denied), FD (denied/fully denied), and UN (unknown) labels are conventionally shown using the following *check marks*, respectively: ✓ ✓̇ ✗̇ ✗ ✗̇ ?

In Yu et al. [3] there are three other aspectual requirements; however, for the sake of discussions and illustration, these three aspectual requirements in the chapter are adequate.

Given the aspectual requirements diagram in Fig. 2.3, how do we perform the reasoning on then? In fact, it is to be done by encoding the rules (2.1) and (2.2) in the same way as any existing goal reasoning algorithm. The only difference is that, when presented to the requirements engineer, much fewer number of contribution links will be needed [3]. Figure 2.4 shows the results of how the labels are propagated once the aspectual requirements model is given, either top-down or bottom-up.

2.3 Dynamic Goals Aspects

In this section we define the notion of dynamic aspectual requirements and present a list of rules that can govern the changes with respect to them. As shown in the introduction section, our aim is to be able to find crosscutting concerns in dynamic requirement problems (2.1t) and (2.2t) such that it is still feasible to reason about the equivalence of (2.1t) and (2.2t). To recall these rules, we copy them here so that it is possible to revisit them in this section:

$$\mathrm{E}(\mathrm{t}), \mathrm{S}(\mathrm{t}) \Big| = \mathrm{R}(\mathrm{t}) \tag{2.1t}$$

$$\mathrm{E}(\mathrm{t}), \mathrm{S_F}(\mathrm{t}), \mathrm{S_Q}(\mathrm{t}) \Big| = \mathrm{R_F}(\mathrm{t}) \wedge \Phi(t) \ \left(\mathrm{R_Q}(\mathrm{t})\right)$$

$$\mathrm{R}(\mathrm{t}) = \mathrm{R_F}(\mathrm{t}) \wedge \Phi(t) \ \left(\mathrm{R_Q}(\mathrm{t})\right)$$

$$\mathrm{S}(\mathrm{t}) = \mathrm{S_F}(\mathrm{t}), \mathrm{S_Q}(\mathrm{t}) \tag{2.2t}$$

$$\mathrm{E}(\mathrm{t}), \mathrm{S_F}(\mathrm{t}) \Big| = \mathrm{R_F}(\mathrm{t})$$

After the discussion in Sect. 2.2, it should be clear now that the aspectual requirements indeed contain both S_Q and R_Q in the representation after the composition of S_F, S_Q. However, at runtime *t*, all of them can be dynamic (i.e. changeable). Even for the satisficing interpretation $\Phi(t)$ is not always constant, for example, when the survivability of the system is concerned [16]. To assist the maintenance of the reasoning relationships between the aspectual requirements, we need to establish a list of basic rules.

Since the weaving of a requirement aspect can be defined by **a tuple <soft goal Φ (R_Q), advice function S_Q, point-cuts S_F, join-operation ",">**, the basic reasoning rules concerning the changes to any one of them are as below:

- SG1—**soft goal change Φ (t) (R_Q (t))**: If a quality requirement R_Q changes its expected satisficing level Φ (t) due to the change of the satisficing criteria, then there is no need to change the tuple except that one must re-evaluate every rule involving $R_Q(t)$ either bottom-up, by label propagation or top-down, using a SAT solver reasoning. An example of such changes can be illustrated, for example, by modifying the "Usability" expectation from PS to FS. In that case, unless an operationalisation through the MAKE contribution link is found, it is not possible to satisfy the requirements model. In this case, one needs to consider the existing advising function inadequate or there could be an obstacle to fulfil the new soft goal. Introducing new advising function, such as "Common Look and Feel", "Multi Touch interface" may solve or at least alleviate this problem.
- SG2—**soft goal change $S_Q(t) \rightarrow R_Q$ (t)**: If the label of a contribution link from S_Q to R_Q changes, e.g. from HELP to MAKE or vice versa, then there is no need to change all tuples except that one must re-evaluate every rule involving $S_Q \rightarrow R_Q$ (t). For example, if one modifies the label of the "Authentication" → "Security" contribution link from MAKE to HELP, to reflect the new situation that Denial of Service attack may hinder the system to serve all customers at all times. In this case, additional security measure needs to be introduced to mitigate the risk [17]. To do this at runtime means that the requirement aspect or its implementation needs to be extensible at runtime.
- AF1 **(advice function S_Q(t) changes):** This type of changes could happen when $E(t), S_F(t), S_Q(t)| = R(t)$ no longer holds even when R_Q is the same. For example, whether or not the contribution link between "Authentication" and "Security" is a MAKE or HELP relation depends on how strong the Login function is implemented. If it was encrypted using an algorithm that is no longer safe, for example, then the relationship $S_Q \rightarrow R_Q$ needs to be revisited.
- PC1 **(point-cuts $S_F(t)$, S_Q (t) changes):** It is perhaps the most frequent changes with respect to the aspectual requirements because the interface between S_F and S_Q affects the scope of the requirement aspect as well. Just as in AOP [18] whereby the signature of a method call used in the point-cut expression is changed would significantly change the scope of the aspect, it is also the case in the aspectual requirements. According to the definition of the point-cut of aspectual requirements, any changes to the contribution links between S_Q and R_Q, or S and R_Q, would lead to redefinition of the point-cuts. For example, modifying the topics of "Shopping" goal to include "Account" or "Balance", it would require the "Authentication" aspect to be weaved because they are sensitive to the mechanisms of protections. However, if there was no contribution link between "Shopping" and "Security", the high-level requirements model will be inconsistent with the lower one. A resolution is either recommending the removal of the "Account/Balance" access in the "Shopping" goal, or making it explicit that the same level of "Security" for "Admin" is required for the "Shopping" as well.
- JO1 **(join-operation "," semantics change)**: Although AND decomposition is often the case for weaving aspectual requirements (as shown in all the three examples), sometimes other form of operation is allowed too. For example,

especially when even the $PS(R_Q^{\text{usability}})$ is not required, then one could even remove the application of the "StyleBox" aspect at the runtime. Therefore it is more appropriate to model the weaving operation as optional or OR decompositions in such cases. Otherwise, contextual conditions need to be added to the interface to enrich its semantics, e.g. by insisting on monitoring the context variable for the need of usability. In certain cases, even existing AND decomposition weaving operations can be sacrificed to guarantee the survivability of the system [16]. Therefore it is important to manage the join-operation dynamically.

2.4 Common Case Study Discussion

A common case study in the crisis management domain [4] is used in the book. Specifically, we use the car crash crisis management system (CCCMS) in our case study based on given requirements. In this case study, we show how goal aspects can be identified and represented and how the rules identified in Sect. 2.3 do apply for the dynamic goal aspects.

2.4.1 Goal Model

Figure 2.5 shows the top-level goal model of CCCMS. The top-level hard goal "Resolve Car Crash Crisis" is decomposed by function into four sub-goals, "Capture Witness Report [Witness, Crisis]", "Create Mission [Mission]", "Allocate Resource [resource]" and "Execute Mission [Mission]". Besides these functional requirements, CCCMS is expected to meet quality requirements like "Facilitate Future Analysis", "Security", "Reliability" and "Mobility", which are represented by soft goals. Specifically, all the sub-goals of "Resolve Car Crash Crisis" should help "Facilitate Future Analysis", since historical analysis involves records about the whole crisis resolution process; the "Capture Witness Report", "Allocate Resource" and "Execute Mission" must ensure "security", because they are relevant to interactions with external uses and systems; "Allocate Resource" and "Execute Mission" must achieve "Reliability" and "Mobility" to ensure reliable communication with mobile rescue resources (e.g. firemen, doctors, policemen); all the sub-goals of "Resolve Car Crash Crisis" must ensure "Real Time".

2.4.2 Goal Aspect Analysis

After goal refinement and operationalisation, we can get the refined goal model in Fig. 2.6. In the model, soft goals are separately operationalised into tasks,

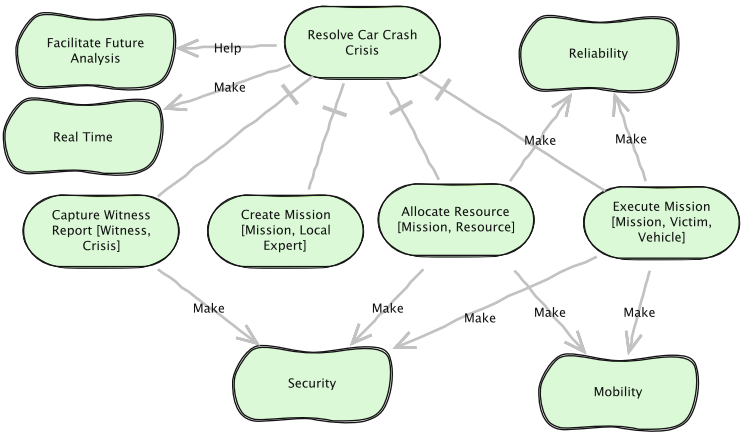

Fig. 2.5 The top-level hard and soft goals of CCCMS

which are weaved into relevant hard goals as advices. For example, as the operationlisations of "Facilitate Future Analysis", both the "Logging [Text]" and "Logging [Multimedia]" are weaved into the four sub-goals of the root goal as alternative implementation. They have different contributions to "Facilitate Future Analysis" and other soft goals. The "Logging [Multimedia]" means to record the crisis resolution process by audios and videos; thus it can achieve "Facilitate Future Analysis" but hurts "Real Time" at the same time.

By contrast, "Logging [Text]" records relevant events in text and thus can help "Facilitate Future Analysis" and has no obvious influence on "Real Time".

Based on the refined CCCMS goal model, we can identify candidate goal aspects as listed in Table 2.1. Besides "Facilitate Future Analysis", "Security", "Reliability" and "Mobility" are also identified as candidate goal aspects. All of them are operationalised into a set of tasks and linked with relevant hard goals by some point-cuts. The soft goal "Real Time" is not identified as a goal aspect, and although it actually constrains the implementation of basic functions, there are no obvious operationalisations for it at this level of abstraction in our modelling exercise.

2.4.3 Dynamic Goal Aspects

Given the goal aspects, we can determine the satisfaction levels of high-level goals in a *bottom-up* way or plan a minimum task set for specific top-level requirements by goal reasoning. For example, given the top-level requirements $R = R_F{}^{\text{Resolve Car Crash Crisis}} \wedge FS\left(R_Q{}^{\text{Facilitate Future Analysis}}\right) \wedge PS\left(R_Q{}^{\text{Real Time}}\right) \wedge FS\left(R_Q{}^{\text{Reliability}}\right) \wedge FS\left(R_Q{}^{\text{Security}}\right) \wedge FS\left(R_Q{}^{\text{Mobility}}\right)$, one would find "Logging [Multimedia]" in the plan (see Fig. 2.7), since it achieves "Facilitate Future Analysis" and hurts "Real Time".

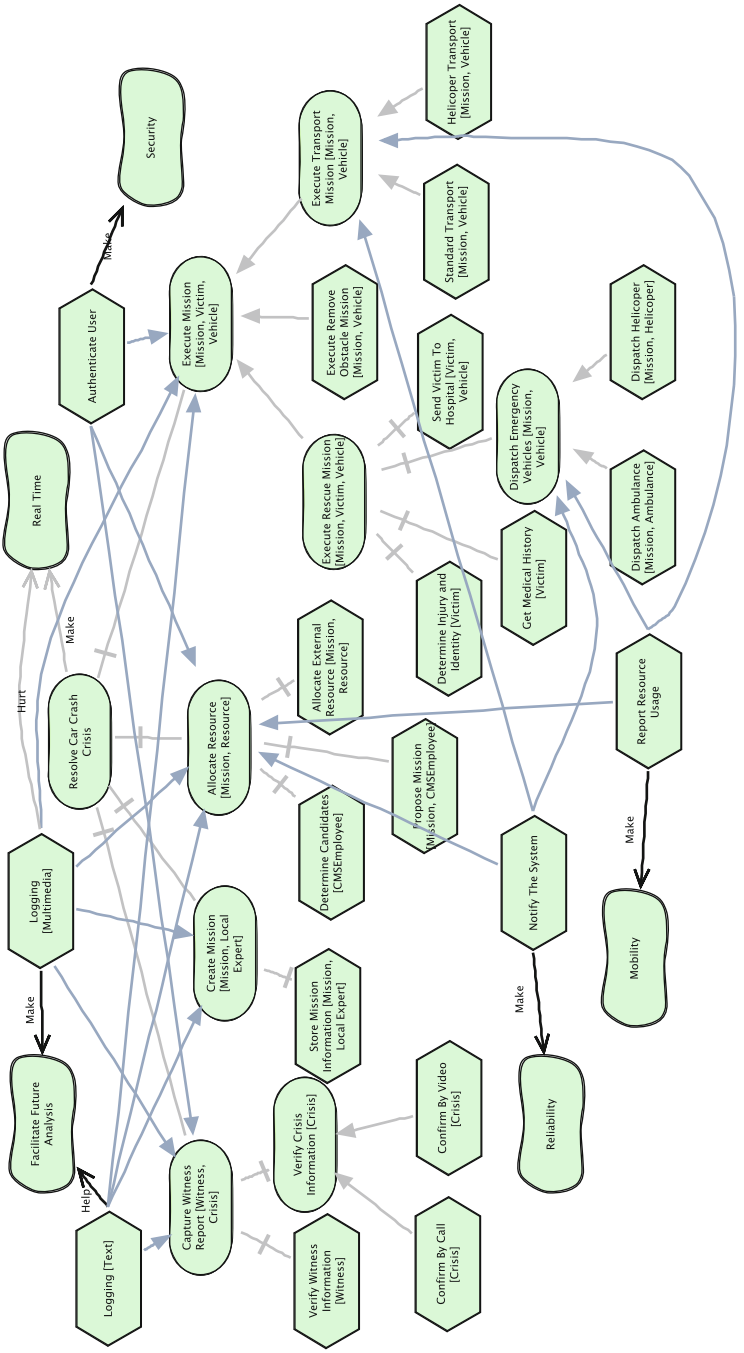

Fig. 2.6 Refined CCCMS goal model with candidate goal aspects

Table 2.1 Part of the identified goal aspects from CCCMS

Aspect (R_Q)	Advice Function (S_Q)	Point-cuts (S_F)
Facilitate Future Analysis	Logging	Capture Witness Report, Create Mission, Allocate Resource, Execute Mission
Security	Authenticate User	Capture Witness Report, Allocate Resource, Execute Mission
Reliability	Notify The System	Allocate Resource, Dispatch Emergency Vehicles, Execute Transport Mission
Mobility	Report Resource Usage	Allocate Resource, Dispatch Emergency Vehicles, Execute Transport Mission

Considering the changing environment and the runtime adaptation of CCCMS, the reasoning about goal aspects may change at runtime. Figure 2.7 shows an example of software goal change in CCCMS (Rule SG1), which involves a dynamic tradeoff between "Facilitate Future Analysis" and "Real Time" at runtime. In this scenario, the initial requirements include $\mathrm{FS}(R_Q^{\text{Facilitate Future Analysis}}) \wedge \mathrm{PS}(R_Q^{\text{Real Time}})$. Corresponding plan for this requirement includes "Logging [Multimedia]" for recording crisis processing logs. With the rapidly increasing user requests, real-time processing of requests becomes a more urgent requirement. And due to the conflict between "Facilitate Future Analysis" and "Real Time", the expected requirements change to $\mathrm{PS}(R_Q^{\text{Facilitate Future Analysis}}) \wedge \mathrm{FS}(R_Q^{\text{Real Time}})$. Accordingly, "Logging [Multimedia]" is replaced by "Logging [Text]" to achieve better satisfaction for "Real Time" with lowered satisfaction level of "Facilitate Future Analysis".

Another example of dynamic goal aspects in CCCMS is shown in Fig. 2.8, which illustrates the case of point-cuts change (Rule PC1). Initially, the hard goal "Create Mission" involves local expert in the creation of mission plans. At that time, it is irrelevant to the goal aspect "Security" and its advising task "Authenticate User", since it only involves local interactions within the crisis management centre. In some cases, remote experts are involved to evaluate the situation and define necessary missions, for example, when local experts are not available. Thus, "Create Mission" becomes a basic function that is relevant to security assurance, since it involves interactions with remote users. As a result, the point-cuts (i.e. the scope) of the goal aspect "Security" change to include "Create Mission" as shown in Fig. 2.8.

2.5 Limitations and Discussions

The proposed change propagation rules do cover the general types of changes in our modelling framework; however, there are several limitations known. As one can see from the CCCMS case study, all the change propagation rules are applicable. As they stand currently, however, the coverage of all possible changes

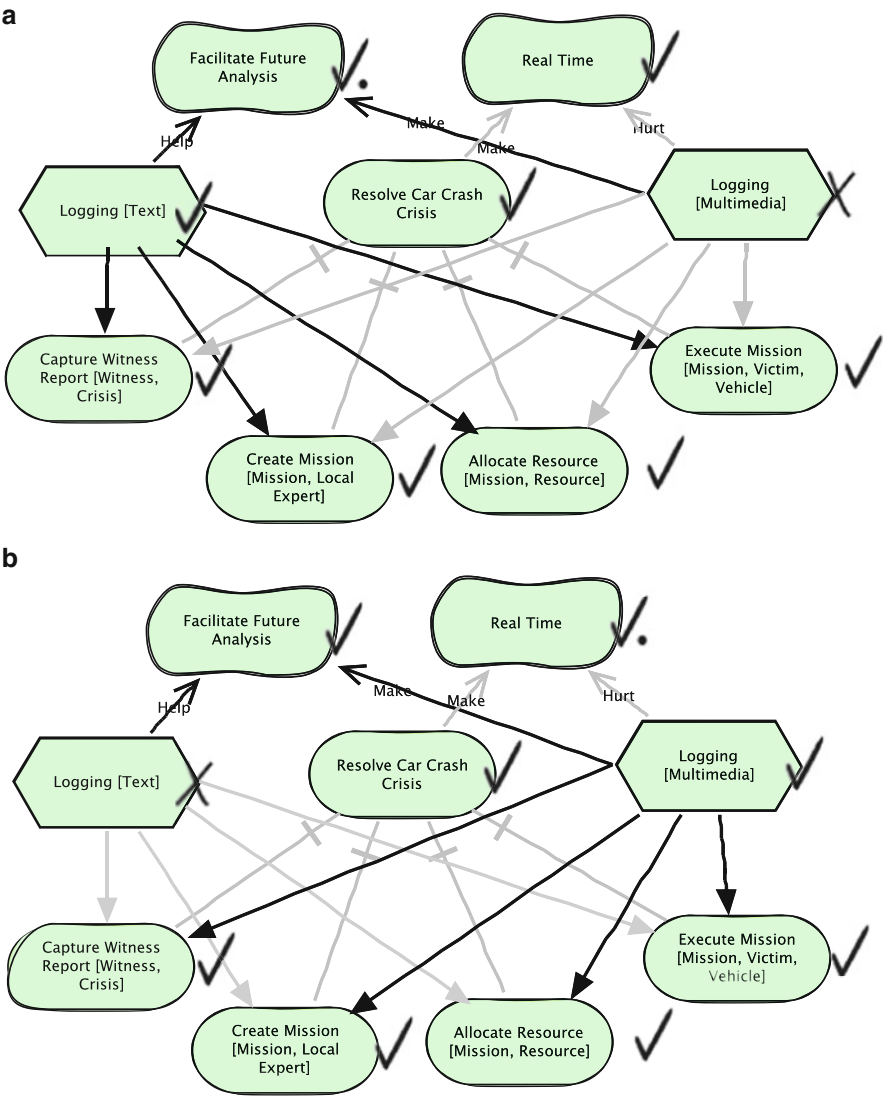

Fig. 2.7 Example of software goal change in CCCMS (Rule SG1) (**a**) before change (**b**) after change

to the crosscutting requirements is not complete. For instance, adding a quality soft goal would require additional aspectual requirements to be inserted into the model. However, without explicit analysis of such a new aspectual requirement, the crosscuts to existing functional requirements are likely missing when the topic names do not match. Therefore, a similarity-based computation is required to help requirements analysts to find more recalls.

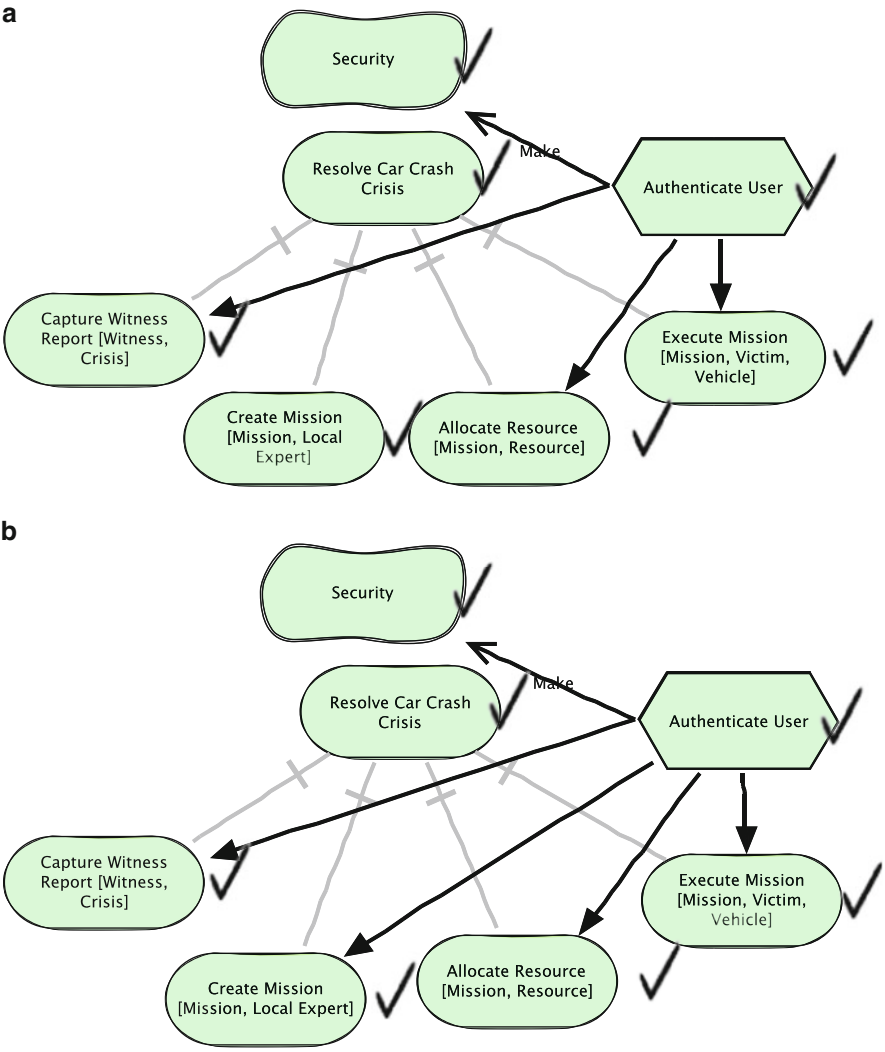

Fig. 2.8 Example of point-cuts change in CCCMS (Rule PC1) (**a**) before change (**b**) after change

It is worth to note that, currently, there is no knowledge about how invariant is the structure of the high-level goal structures. A possibility is that they tend to change more often for the application software for customer market, such as mobile apps than for those software systems of a mature domain, such as compilers and relational database management systems.

As we stated earlier, the continuous satisfaction of the "invariant" core requirements can be regarded as a quality requirement for the software evolution process. Our assumption is that such process-oriented quality requirements are to be

addressed by additional monitoring, diagnosis and feedback requirements. Even so, the scope of product-oriented requirements invariants may still be rather relative. If the software developers decided to substantially/radically change the functional requirements from one domain to a completely different one, one would not be able to maintain the traceability to these invariant requirements.

In model-driven software development, on the other hand, such invariant traceability between software models and implementation code can be largely maintained through bidirectional transformations [19]. It is a future direction to consider whether aspectual requirements, given the sophisticated many-to-many change propagations, can be maintained through bidirectional transformations.

Another limitation is that our rules are bound by our representation, and as such they may be incomplete for complex evolution scenarios, for example, when multiple types of changes happen at the same time. In the worst case, such incremental adjustment of the reasoning process will have to degenerate into complete recalculations. Ernst et al. [20] point out that when the logic is as simple as propositional and the soft goals are excluded in the modelling, there is an efficient incremental reasoning algorithm to preserve the existing solutions. It remains a future work to include the soft goals when such automation is needed.

2.6 Conclusion and Future Work

In this chapter, we present a list of adaptation rules for the aspectual requirements to be managed at the runtime. Using Zave et al.'s formula for basic requirements problems, we explained how different concepts in aspectual requirements are formulated, and reasoned about. Furthermore, the basic adaptation rules are classified according to their roles played in the runtime changes. It is our hope that these rules can be at least useful in dynamic reasoning of the aspectual requirements, even if they are incomplete for all dynamic requirements.

The formula we presented, although looks simple, reflects more expressive reasoning rules because both the context and the goals can participate in the computation. By turning the missing goals or the missing context into wildcards, the same rule can be used to model all the rules [12–14].

The work has some limitations to be overcome in future. For example, we do not handle early aspects of other forms, such as natural language documents [21], or trust assumptions about the threat descriptions [22]. Also it is clear that having some form of tool support would greatly enhance the applicability of the methodology to larger and more interesting case studies. Currently we have implemented a simple form of aspect-monitoring framework based on our existing tool support of OpenOME [23] and OpenArgue [24]. Although they are able to perform the required reasoning tasks, it is still worthy to explore how well the work of incremental reasoning [20, 25] could be incorporated in the near future.

References

1. L. Chung, J.C.S. do Prado Leite, On non-functional requirements in software engineering, in *Conceptual Modeling: Foundations and Applications*, ed. by A.T. Borgida, V.K. Chaudhri, P. Giorgini, E.S. Yu (Springer, Berlin, 2009), pp. 363–379
2. J. Mylopoulos, L. Chung, B. Nixon, Representing and using nonfunctional requirements: a process-oriented approach. IEEE Trans. Softw. Eng. **18**(6), 483–497 (1992)
3. Y. Yu, J.C.S. do Prado Leite, J. Mylopoulos, From goals to aspects: discovering aspects from requirements goal models, in *Presented at the RE*, 2004, pp. 38–47
4. J. Kienzle, N. Guelfi, S. Mustafiz, Crisis management systems: a case study for aspect-oriented modeling, in *Transactions on Aspect-Oriented Software Development*, ed. by S. Katz, M. Mezini (Springer, Berlin, 2010), pp. 1–22
5. P. Zave, M. Jackson, Four dark corners of requirements engineering. ACM Trans. Softw. Eng. Methodol. **6**(1), 1–30 (1997)
6. I.J. Jureta, J. Mylopoulos, S. Faulkner, A core ontology for requirements. Appl. Ontol. **4**(3), 169–244 (2009)
7. N. Niu, Y. Yu, B. González-Baixauli, N.A. Ernst, J.C.S. do Prado Leite, J. Mylopoulos, Aspects across software life cycle: a goal-driven approach. Trans. Aspect Oriented Softw. Dev. **6**, 83–110 (2009)
8. M. Salifu, Y. Yu, B. Nuseibeh, Specifying monitoring and switching problems in context, in *15th IEEE International Requirements Engineering Conference*, 2007, pp. 211–220
9. Y. Wang, S.A. McIlraith, Y. Yu, J. Mylopoulos, Monitoring and diagnosing software requirements. Autom. Softw. Eng. **16**(1), 3–35 (2009)
10. X. Peng, B. Chen, Y. Yu, W. Zhao, Self-tuning of software systems through dynamic quality tradeoff and value-based feedback control loop. J. Syst. Softw. **85**(12), 2707–2719 (2012)
11. M. Salifu, Y. Yu, A.K. Bandara, B. Nuseibeh, Analysing monitoring and switching problems for adaptive systems. J. Syst. Softw. **85**(12), 2829–2839 (2012)
12. P. Giorgini, J. Mylopoulos, E. Nicchiarelli, R. Sebastiani, Reasoning with goal models, in *Conceptual Modeling—ER 2002*, ed. by S. Spaccapietra, S.T. March, Y. Kambayashi (Springer, Berlin, 2003), pp. 167–181
13. R. Sebastiani, P. Giorgini, J. Mylopoulos, Simple and minimum-cost satisfiability for goal models, in *Advanced Information Systems Engineering*, ed. by A. Persson, J. Stirna (Springer, Berlin, 2004), pp. 20–35
14. R. Ali, F. Dalpiaz, P. Giorgini, Reasoning with contextual requirements: detecting inconsistency and conflicts. Inf. Softw. Technol. **55**(1), 35–57 (2013)
15. X. Peng, Y. Yu, W. Zhao, Analyzing evolution of variability in a software product line: from contexts and requirements to features. Inf. Softw. Technol. **53**(7), 707–721 (2011)
16. B. Chen, X. Peng, Y. Yu, W. Zhao, Are your sites down? Requirements-driven self-tuning for the survivability of Web systems, in *Requirements Engineering Conference (RE), 2011 19th IEEE International*, 2011, pp. 219–228
17. V.N.L. Franqueira, T.T. Tun, Y.Yu, R.Wieringa, B. Nuseibeh, Risk and argument: a risk-based argumentation method for practical security, in *RE*, 2011, pp. 239–248
18. G. Kiczales, Aspect-oriented programming. ACM Comput. Surv. **28**(4es), 154 (1996)
19. Y. Yu, Y. Lin, Z. Hu, S. Hidaka, H. Kato, L. Montrieux, Maintaining invariant traceability through bidirectional transformations, in *ICSE*, 2012, pp. 540–550
20. N.A. Ernst, A. Borgida, I. Jureta, Finding incremental solutions for evolving requirements, in *Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference*, Washington, DC, 2011, pp. 15–24
21. A. Rashid, A. Moreira, J. Araújo, Modularisation and composition of aspectual requirements, in *Proceedings of the 2nd International Conference on Aspect-Oriented Software Development*, New York, NY, 2003, pp. 11–20

22. C.B. Haley, R.C. Laney, B. Nuseibeh, Deriving security requirements from crosscutting threat descriptions, in *Proceedings of the 3rd International Conference on Aspect-Oriented Software Development*, New York, NY, 2004, pp. 112–121
23. J. Horkoff, Y. Yu, E.S.K. Yu, OpenOME: an open-source goal and agent-oriented model drawing and analysis tool, in *Presented at the iStar*, 2011, pp. 154–156
24. Y. Yu, T.T. Tun, A. Tedeschi, V.N.L. Franqueira, B. Nuseibeh, OpenArgue: supporting argumentation to evolve secure software systems, in *19th IEEE International Requirements Engineering Conference*, 2011
25. N.A. Ernst, J. Mylopoulos, Y. Yu, T. Nguyen, Supporting requirements model evolution throughout the system life-cycle, in *16th IEEE International Requirements Engineering, 2008. RE '08*, 2008, pp. 321–322

http://www.springer.com/978-3-642-38639-8

Aspect-Oriented Requirements Engineering
Moreira, A.; Chitchyan, R.; Araújo, J.; Rashid, A. (Eds.)
2013, XIX, 383 p. 195 illus., Hardcover
ISBN: 978-3-642-38639-8