

A taxonomy for identifying requirement interactions in software systems

Mohamed Shehata ^{a,c,*}, Armin Eberlein ^b, Abraham O. Fapojuwo ^a

^a Department of Electrical and Computer Engineering, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada

^b Department of Computer Engineering, American University of Sharjah, P.O. Box 26666, Sharjah, United Arab Emirates

^c Department of Electrical and Computer Engineering, Shoubra Faculty of Engineering, Benha University, 108 Shoubra Street, Cairo, Egypt

Available online 27 September 2006

Responsible Editor: H. Rudin

Abstract

This paper presents an interaction taxonomy for classifying and identifying requirement interactions in software systems. The proposed taxonomy is in the form of a four-layered pyramid that defines 6 Main Interaction Categories in the first layer, 17 Interaction Subcategories in the second layer, 29 Interaction Types in the third layer, and 29 Interaction Scenarios in the fourth layer. Each interaction scenario has a corresponding interaction detection guideline that describes how the interaction can be detected. The proposed interaction taxonomy was compared to other existing taxonomies in the literature and was not only able to address all the issues in those taxonomies, but also contained many other interaction types. The proposed interaction taxonomy serves as the first domain-independent requirement interactions taxonomy. It provides a detailed description of when two requirements interact.

© 2006 Published by Elsevier B.V.

Keywords: Requirement engineering; Requirement interaction taxonomy; Interaction scenarios

1. Introduction

A key issue in obtaining a set of clear requirements is to introduce ways to manage negative relationships between requirements [1]. Robinson et al. defines requirement interaction management as “the set of activities directed towards the discovery, management, and disposition of critical relationships among a set of requirements” [2]. Requirements often interact because of the heterogeneity and diversity of stakeholders [3]. Hence, there is a need to have a requirement interactions taxonomy that would answer questions such as: When and why do two requirements interact? How to detect this interaction? And how do we resolve it?

* Corresponding author. Address: Department of Electrical and Computer Engineering, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada. Tel.: +1 403 2108196; fax: +1 403 282 6855.

E-mail address: Msshahat@ucalgary.ca (M. Shehata).

To the best of our knowledge, not much work has been done in the area of general requirement interactions taxonomies. Even though Robinson et al. defined in detail the concept of requirement interactions in [2], his work does not include in-depth information on when two requirements are considered interacting and how to detect such interactions. This area has traditionally been addressed by the feature interaction community. In 1994, Cameron et al. published a paper [4] describing a benchmark for classifying the different categories of feature interactions. However, this paper is very specific to the telecommunications domain and all examples are related to interactions between telephony features and therefore it is very hard to generalize to other software domains. In 2000, Gibson et al. presented a taxonomy for triggered interactions using fair objects semantics [5]. This work builds on the assumption of having a set of triggered features and using a semantic point of view for classifying interactions between those telecommunications features. Hence, the work by Gibson et al. [5] cannot be used beyond its assumption especially in cases where there can be triggered and non-triggered requirements. In 2004, Reiff–Marganiec and Turner presented a taxonomy for identifying policy conflicts [6]. However, this work focuses on the social nature of policies interactions and the reasons why they occur. Also, the taxonomy in [6] is geared towards the policy domain and is therefore not generally applicable. There are also research efforts to present partial taxonomies as sections of papers or thesis where no claim of completeness has been made [7–10].

This paper investigates the problem of requirement interactions in software systems and presents a general interaction taxonomy for classifying and identifying requirement interactions. The proposed taxonomy is in the shape of a four-layered pyramid where the first layer describes 6 main interaction categories, the second layer describes 17 interaction subcategories, the third layer describes 29 interaction types, and finally the fourth layer describes 29 interactions scenarios that also contain 29 interaction detection guidelines. This in-depth structure addresses the lack of details that exist in other interaction taxonomies. Moreover, the proposed interaction taxonomy was compared to other existing taxonomies in the literature and was not only able to address all the issues in those taxonomies, but also contained other interaction types.

The remainder of this paper is structured as follows: Section 2 presents the concept of system decomposition. Section 3 presents the proposed interactions taxonomy. Section 4 compares the proposed interaction taxonomy with already existing taxonomies. Then Section 5 contains the paper's conclusions.

2. System decomposition

In this paper we address the problem of requirement interactions and hence we focus our effort on detecting interactions during the requirements engineering phase of system development. The output of the requirements engineering phase is a requirements specification document that contains a set of requirements that describe stakeholders' needs. This set of requirements can either describe certain properties that have to be preserved (static view) or dynamic behaviour which the system exhibits when certain triggers occur (dynamic view). Usually there is also a description of resources the system uses (environmental view). Therefore, we consider a system to consist of the following components:

1. **System axioms:** Each system axiom describes a certain property of the system that must be preserved. For example, in the lift system [11], a system axiom states that “At any time the user can press a call button to call the lift”. Any system axiom consists of at least the following basic attributes [12]: ID, which is a unique identifier for that system axiom; Description, which is an informal description of the system axiom as specified in the requirements document; and Rule, which is a description of the property that must be preserved.
2. **Dynamic behaviour requirements:** Each dynamic behaviour requirement describes how the system should behave when it is in a certain state and a specific trigger event occurs. For example, a requirement from the lift system might state the following: “When the lift stops at floor K, it will open its doors”. Any dynamic behaviour requirement will consist of ID and Description attributes, similar to those listed in system axioms, plus at least the following basic attributes: Prestate, which is a description of the required system state prior to the execution of this dynamic requirement; Trigger event, which is a description of the trigger event required for this dynamic requirement to execute; Action, which is a description of the action carried out by this dynamic requirement once triggered; and Next state, which describes the next state that the system reaches after executing this requirement [12].

3. Resources: Each resource describes the physical elements that the system uses to fulfill its requirements. For example, Infra Red sensors (IR) in security systems are considered as resources used to detect motion. Any resource will consist of ID and Description attributes, similar to the ones listed in system axioms plus at least the following basic attributes: Availability, which describes this resource’s availability; Performance, which describes this resource’s performance; and Interface, which describes this resource’s interface [12].

Any software system can be described during the early development stage with a detailed textual description of requirements as mentioned above. However, the software system can also be described using features such as in the case of telecommunication systems, or policies as in the case of smart homes systems. Regardless of the way a system is described (using requirements, features, or policies), the system will still consist of the three main components: a static view represented by system axioms, a dynamic view represented by dynamic behaviour, and an environmental view represented by resources. Since any of these elements can be represented using the attributes discussed above (e.g., Prestate, Trigger Event, Action, and Next State in the case of dynamic behaviour), the proposed interaction taxonomy is suitable for detecting interactions between requirements, between features, or between policies. This has been demonstrated by the different examples presented in the fourth layer in Section 3 and also in Appendix A.

3. The proposed interaction taxonomy

3.1. General architecture

The architecture of the proposed interaction taxonomy is shown in Fig. 1. The proposed taxonomy starts in the first layer by addressing the question of WHERE in the system the interactions can occur. Whenever two elements (either from two different components, e.g., a system axiom and a resource, or from the same component, e.g., two system axioms) interact, they are said to form a main interaction category that describes where the interaction manifests itself.

The second layer of the taxonomy contains interaction subcategories and addresses the question of WHAT attributes of the two interacting system elements, from the first layer, cause the interaction. The third layer of the proposed taxonomy contains interaction types and addresses the question of WHY the attributes identified in the second layer cause the interaction to occur. The fourth layer contains interaction scenarios and addresses the question HOW to detect interaction types identified in the third layer.

The question of how to resolve interactions was left out of the taxonomy as resolution strategies heavily depend on stakeholders preferences and on the software domain.

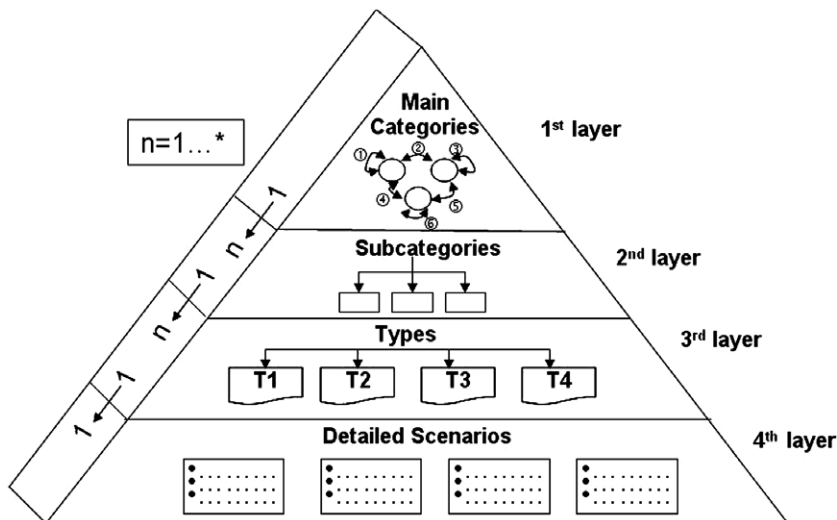


Fig. 1. General architecture of the proposed taxonomy.

3.2. First layer: main interaction categories

There are nine main interaction categories based on all possible pairwise combinations among system axioms, dynamic behaviour requirements and resources. However, for reasons of simplicity, the main interaction category “A dynamic behaviour requirement interacting with a system axiom” has been merged with the main interaction category “A system axiom interacting with a dynamic behaviour requirement” and hence the latter will include situations where the negative effect is from the system axiom on the dynamic behaviour requirement and situations when the negative effect is from the dynamic behaviour requirement on the system axiom. This is distinguished later in the fourth layer of the taxonomy through the interaction scenario that describes which of the two negatively affects the other. Similarly, “A resource interacting with a system axiom” has been merged with “A system axiom interacting with a resource”, and “A resource interacting with a dynamic behaviour requirement” has been merged with “A dynamic behaviour requirement interacting with a resource”. Thus, the number of main interaction categories has been reduced from nine to six and are listed as follows (see also Fig. 2):

- ① Two interacting system axioms.
- ② A system axiom interacting with a dynamic behaviour requirement.
- ③ Two interacting dynamic behaviour requirements.
- ④ A system axiom interacting with a resource.
- ⑤ A dynamic behaviour requirement interacting with a resource.
- ⑥ Two interacting resources.

In the remainder of the main body of this paper we explain and describe only the main interaction category “Two interacting dynamic behaviour requirements” as an ongoing example to fully explain the four layers of the proposed taxonomy. The other five main interaction categories ①, ②, ④, ⑤ and ⑥ are included in [Appendix A](#) at the end of this paper to provide complete details of the entire taxonomy.

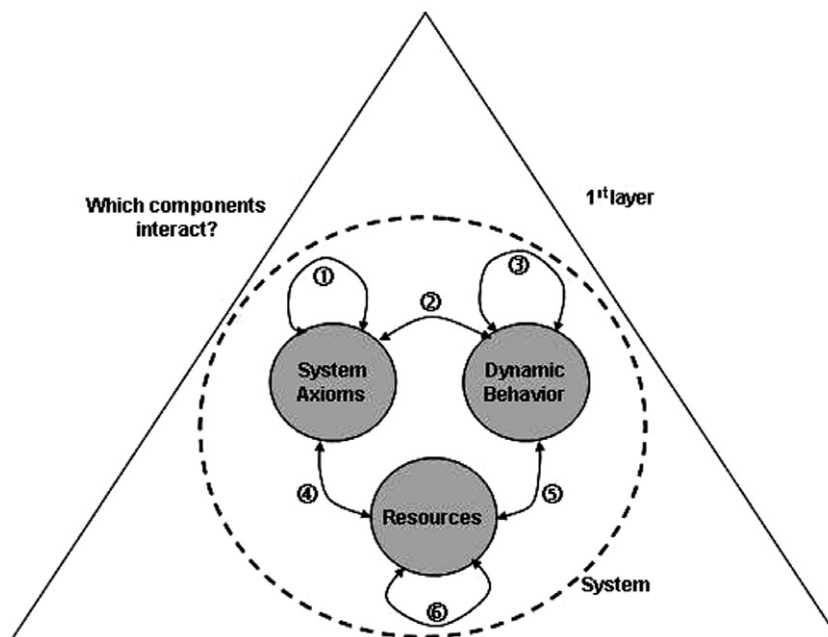


Fig. 2. First layer of the proposed taxonomy.

3.3. Second layer: interaction subcategories

3.3.1. General description

The second layer of the proposed interaction taxonomy contains interaction subcategories that are linked to the first layer through a decomposition relationship which is done using attributes as the basis for the decomposition. An interaction subcategory describes what attributes of the two interacting system elements, identified in the first layer, are really causing the interaction to occur. Therefore, to generate the second layer interaction subcategories, each possible pair of attributes between the two interacting elements in which the first attribute is from the first requirement and the second attribute is from the second requirement, is first listed. Then the obtained pairs of attributes are analyzed to determine which ones can cause interactions. Any pair of attributes that could cause an interaction is then listed and considered to be an interaction subcategory. Therefore, the main interaction categories from the first layer are decomposed into different numbers of interaction subcategories in the second layer depending on the outcome of the analysis of attribute pairs (e.g., the third main interaction category has four subcategories – see Fig. 3b – whereas the fourth main interaction category has three subcategories – see Fig. 8). The general structure of decomposing a first layer main interaction category into interaction subcategories in the second layer is shown in Fig. 3a. The first layer’s six main interaction categories resulted in the following 17 interaction subcategories in the second layer: one subcategory (S1) from the main category ①, three subcategories (S2, S3, and S4) from the main category ②, four subcategories (S5, S6, S7, and S8) from the main category ③, three subcategories (S9, S10, and S11) from the main category ④, three subcategories (S12, S13, and S14) from the main category ⑤, and three subcategories (S15, S16, and S17) from the main category ⑥. Note that each subcategory is denoted by S_n where n is the subcategory number.

In the next subsection we continue describing subcategories derived from the main interaction category ③ “Two interacting dynamic behaviour requirements” as our ongoing example. However, Appendix A gives full details about all remaining subcategories.

3.3.2. Interaction subcategories derived from the main interaction category ③

Fig. 3b shows how the main interaction category ③ from the first layer is decomposed into four interaction subcategories in the second layer. The decomposition is based on the attributes of dynamic behaviour requirements, namely: Prestate, Trigger event, Action, and Next state (see Section 2).

After analyzing the possible pairs of attributes that can form interaction subcategories, only the following four pairs were found to really present interactions subcategories:

- S5: Next State–Next State interactions: This subcategory contains all the interactions that arise between two dynamic behaviour requirements because the next state attribute of the first requirement interacts with the next state attribute of the second requirement.
- S6: Action–Action interactions: This subcategory contains all the interactions that arise between two dynamic behaviour requirements because the action attribute of the first requirement interacts with the action attribute of the second requirement.

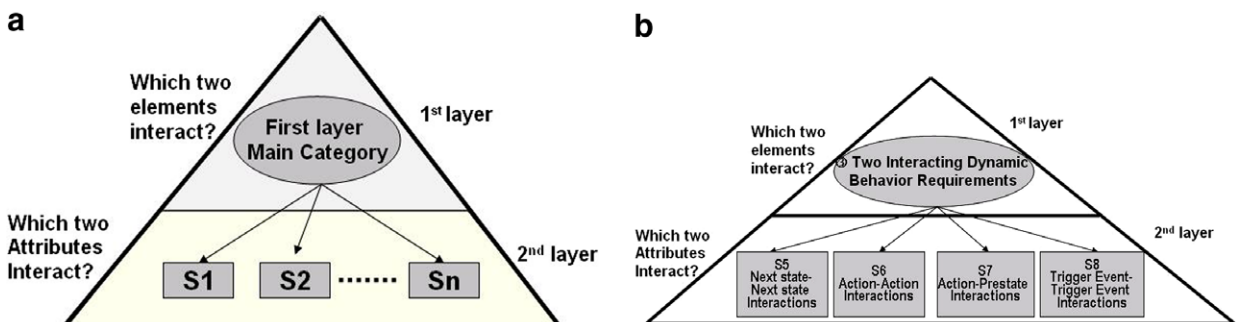


Fig. 3. (a) General structure of decomposing the first layer into the second layer, (b) example: interaction subcategories under the main category ③.

S7: Action–Prestate interactions: This is a subcategory that contains all the interactions that arise between two dynamic behaviour requirements because the action attribute of the first requirement interacts with the prestate attribute of the second requirement.

S8: Trigger Event–Trigger Event interactions: This is a subcategory that contains all the interactions that arise between two dynamic behaviour requirements because the Trigger Event attribute of the first requirement interacts with the Trigger Event attribute of the second requirement.

Note that the numbering of the interaction subcategories started from 5 because there are other subcategories derived from the first two main interaction categories ① and ② which can be found in Appendix A.

3.4. Third layer: interaction types

3.4.1. General description

The third layer of the interaction taxonomy describes the reasons why the attributes, identified in interaction subcategories in the second layer, cause interactions. Each of these reasons are said to form an interaction type. The number of interaction types for an interaction subcategory depends on the number of reasons that cause the two attributes, described in this interaction subcategory, to interact.

Sometimes, certain constraints have to be met for an interaction type to occur. For example, consider the interaction subcategory “Next State–Next State interactions” derived from the main interaction category “Two interacting dynamic behaviour requirements” (see Fig. 4b). This interaction subcategory has only one interaction type t_8 called “Non-Determinism” in the third layer that describes that the two attributes Next State (of the first requirement) and Next State (of the second requirement) interact because they have different values and therefore cause a non-determinism situation in the system. However, for this interaction type to occur, the two dynamic behaviour requirements must have (the same prestates) AND (the same trigger events).

This is considered to be a constraint on the interaction type “Non-Determinism” and therefore the subcategory “Next State–Next State interactions” is connected to the “Non-Determinism” interaction type through the constraint C1 “Same prestates AND same trigger events”. It must be noted that some of the interaction types can be repeated more than once under the same subcategory because this interaction type occurs under two different constraints (e.g., t_{10} and t_{12} under S6 in Fig. 4b). The general structure of deriving interaction types in the third layer from interaction subcategories in the second layer is shown in Fig. 4a.

The 17 interaction subcategories from the second layer resulted in 29 interaction types and 5 constraints in the third layer of the proposed interaction taxonomy. We continue with our ongoing example shown in Fig. 4b, however the rest of the interaction types are described in their respective places in Appendix A.

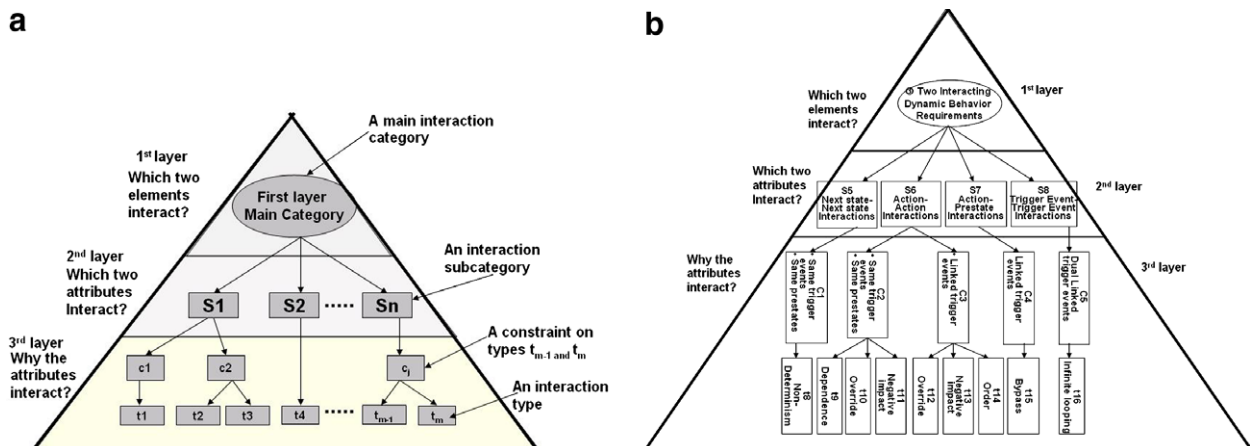


Fig. 4. (a) General structure of deriving interaction types in the third layer, (b) example: interaction types under the main interaction category ③.

3.4.2. Interaction types derived from the main interaction category ③

The 17 interaction subcategories from the second layer resulted in 29 interaction types and five constraints in the third layer of the proposed interaction taxonomy. In the following, we present a description of interaction types under the main interaction category ③ but all the other interaction types are listed in [Appendix A](#). In all the interaction types t8 to t16 described below, consider R1 and R2 to be dynamic behaviour requirements:

- t8: Non-determinism (under constraint C1):** Consider R1 and R2 to have different values for their Next State attributes. If these two requirements are executed at the same time then the system will face an ambiguous situation in which the system is unable to determine which state to go to (the next state specified in R1 or the next state specified in R2). In order for two dynamic behaviour requirements to execute at the same time, R1 and R2 must have: the same prestates AND the same trigger events. This is enforced by C1 that preceded the interaction type t8.
- t9: Dependence (under constraint C2):** Consider R1 and R2 to have the same trigger events and the same prestates and hence will be executed together. Now, suppose that the action of R1 requires that the action of R2 be successfully executed. This means that the action of R1 depends on the action of R2, i.e., an interaction occurs if the action of R2 is not completed successfully for any reason. This is considered as an interaction.
- t10: Override (under constraint C2):** Consider R1 and R2 to have the same trigger events and the same prestates and that they have been triggered and are executing simultaneously. Suppose that the action of R1 interrupts and cancels the action of R2 before its completion which means that the action of R1 has overridden the action of R2. Hence there is a negative relationship between the two requirements and, by definition, R1 and R2 interact.
- t11: Negative impact (under constraint C2):** Consider R1 and R2 to have the same trigger events and the same prestates and that they have both been triggered and are executing simultaneously. Now suppose that the action of R1 negatively impacts the action of R2. Hence, R1 and R2 interact according to the interaction type t11.
- t12: Override (under constraint C3):** Consider R1 and R2 to have different trigger events and thus should be, theoretically, unrelated and hence are not prone to interact. However, if R1 and R2 have their trigger events linked (i.e., the occurrence of the first trigger event is followed after some time by the occurrence of the second trigger event), hence R1 and R2 are still sequentially related and are still prone to interactions. Now suppose that R1 is triggered and starts executing. R2 will also be triggered and starts executing after some time because the trigger event of R2 is linked to the trigger event of R1. Now, assume that the action of R1 is not yet completed while R2 is triggered. If the action of R2 cancels and overrides the action of R1 before its completion then the two requirements interact. The interaction type t12 is also possible when the action of R1 overrides and cancels the action of R2. In both cases, R1 and R2 interact.
- t13: Negative impact (under constraint C3):** Assume R1 and R2 to have linked trigger events and hence if R1 is triggered and starts executing then R2 will also be triggered and starts executing after some time. Now, if the action of R2 negatively affects the action of R1 before its completion then the two requirements interact. This interaction type can also occur when the action of R1 negatively impacts the action of R2 and in both cases R1 and R2 interact.
- t14: Order (under constraint C3):** Consider the two dynamic behaviour requirements R1 and R2 to have linked trigger events. Assume that the trigger of the first requirement leads to the trigger of the second requirement, i.e., $T1 \rightsquigarrow T2$, and in this case the system will exhibit a specific behaviour B1 after the two requirements have executed their actions. Now if this specific behaviour B1 is different from the behaviour that the system would exhibit if R2 had started first then followed by R1, i.e., $T2 \rightsquigarrow T1$, then there is an interaction between the two requirements. This is because in this case the actions of the two requirements are not independent but have an effect on each other. If they were independent then the same behaviour would have been obtained no matter which action started first.

t15: Bypass (under constraint C4): Consider R1 and R2 with linked trigger events. Assume that R1 is triggered and starts executing and that the action of R1 prevents the system from being in a specific state. Suppose that this specific state is the same state specified in the prestate attribute of R2. Hence when the trigger event of R2 occurs, R2 will never execute because the system is in a state different from R2’s prestate. Thus R1 prevented the system from executing R2.

t16: Infinite looping (under constraint C5): If R1 is triggered and starts executing such that its action will create the trigger event for R2 then R2 starts executing its action. Now if the action of R2 causes the creation of the trigger event of the R1, then R1 is triggered again and starts executing its action which will again create trigger event of R2 and so on. Hence, R1 and R2 are forced into infinite looping and interact.

3.5. Fourth layer: interaction scenarios

3.5.1. General description

The fourth layer of the proposed interaction taxonomy contains interaction scenarios that provide detailed explanations of interaction types from the third layer and how to detect these interaction types (Fig. 5a). The template used to present the interaction scenarios contains the following information:

- Scenario ID: This is a unique ID that distinguishes interaction scenarios from each other.
- Interaction type: This is a description of the interaction type this scenario is associated with. The description does not only include the interaction type as a single leaf (e.g., t10) but it includes the whole tree branch starting from the first layer main interaction category until the interaction type this scenario is associated with.
- Detection guideline: The detection guideline describes how to detect the interaction type described in the scenario by a non-expert developer. The guideline includes a textual description and, where appropriate, a graphical description.
- Example: An example from a real life system is given to improve the understanding of how this interaction occurs. Note that in this paper, examples are taken from the smart homes domain [13], the lift system domain [11], e-commerce web domain, [14], and telecommunications domain [15]. However, these scenarios are general enough to be used in other domains.

The next subsection presents interaction scenarios related to the Main interaction category ③ “Two interacting dynamic behaviour requirements” as the ongoing example (Fig. 5b). However, Appendix A gives full details about all remaining scenarios.

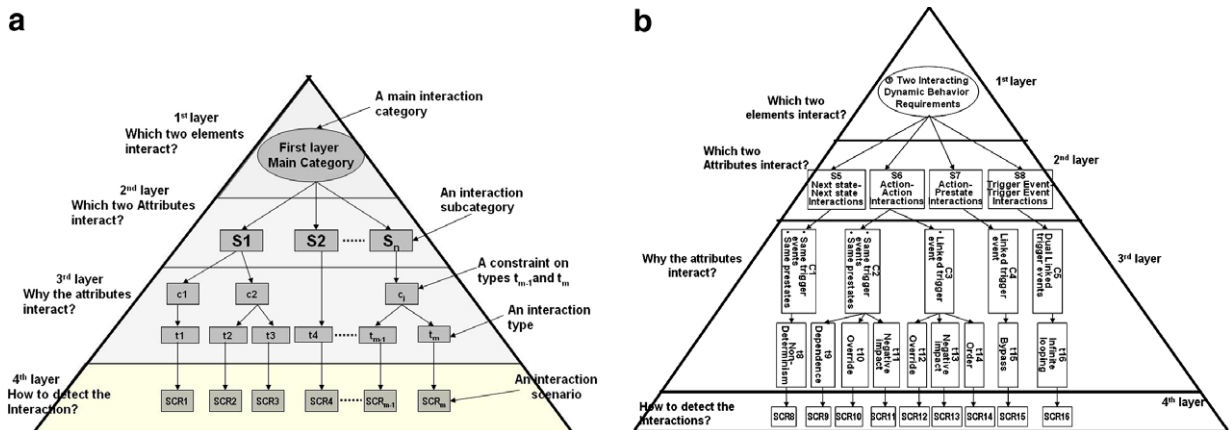
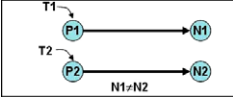
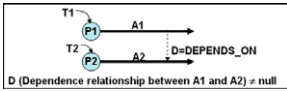
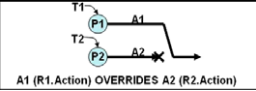


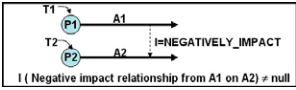
Fig. 5. (a) General structure of deriving interaction scenarios in the fourth layer and (b) example: interaction scenarios under the main interaction category ③.

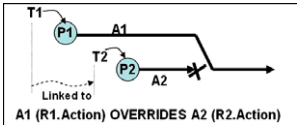
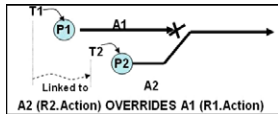
3.5.2. Interaction scenarios under main interaction category ③ “Two interacting dynamic requirements”¹

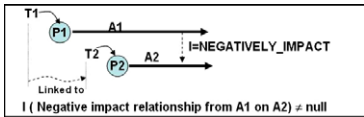
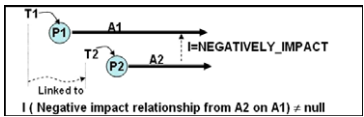
Scenario ID	SCR8
Type of interaction	TwoInteractingDynamicBehaviourRequirements → NextState–NextStateInteractions → Non-Determinism _{C1} = SameTriggerEvents & SamePreStates
Detection guideline	IF {(R1.TriggerEvent = R2.TriggerEvent) AND (R1.PreState = R2.PreState) AND (R1.NextState ≠ R2.NextState)} THEN {R1 interacts with R2 under the interaction type t8}
Example	<ul style="list-style-type: none"> • R1: “Adjust the audio level of the TV device to 35% of its maximum volume when device is first turned on” • R2: “Adjust the audio level of the TV when it is turned on to the last used audio level setting before the last shutdown” • Interaction: Assume that someone previously watched TV and manually adjusted the TV audio level to 20% of its max volume before he shuts it down. Later on, when the TV is first turned on then both R1 and R2 are triggered at the same time. Since the audio level specified in R2 (20% of max. audio level) is different from the audio level specified in R1 (35% of the max. volume), the system will face a non-determinism situation. Should it transit to the state where the TV volume is 20% as specified by R2 or should it transit to the state where the TV volume is 35% as specified by R1?
	
Scenario ID	SCR9
Type of interaction	TwoInteractingDynamicBehaviourRequirements → Action–ActionInteractions → Dependence _{C2} = SameTriggerEvents & SamePreStates
Detection guideline	IF {(R1.TriggerEvent = R2.TriggerEvent) AND (R1.PreState = R2.PreState) AND (R1.Action DEPENDS_ON R2.Action)} THEN {R1 interacts with R2 under interaction type t9}
Example	<ul style="list-style-type: none"> • R3: “Increase the temperature inside the house to the preset temperature (22 °C) when temperature reading from thermostat is ≤ (20 °C)” • R4: “Open the ventilation grills in locations (LivingRoom, BedRoom1) to allow air flow when the temperature reading from the thermostat is ≤ (20 °C)” • Interaction: When the temperature drops below 20 °C, then both requirements R3 and R4 trigger at the same time. However the action of R3 depends on the action of R4 as the temperature is increased by pumping hot air through the ventilation grills. If R4 fails to execute for any reason, then R3 will not be able to perform its action. Further, if the action of R4 opens only one or two ventilation grills, then the action of R3 is affected by the few opened ventilation grills and it will not be effective enough
	

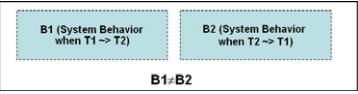
¹ The examples of interaction scenarios in this category are taken from the Smart Homes system which is a representative of the policy domain.

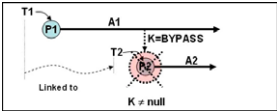
Scenario ID	SCR10	
Type of interaction	TwoInteractingDynamicBehaviourRequirements → Action–ActionInteractions → Override _{C2 = SameTriggerEvents & SamePreStates}	
Detection guideline	IF {(R1.TriggerEvent = R2.TriggerEvent) AND (R1.PreState = R2.PreState) AND (R1.Action OVERRIDES R2.Action)} THEN {R1 interacts with R2 under the interaction type t10}	
Example	<ul style="list-style-type: none"> • R5: “As a security measure, secure the doors and windows of a house by closing them starting at time (11:00 pm) for (6 h)” • R6: “automatically opens the windows in (LivingRoom) at time (11:00 pm)” • Interaction: When the time is 11:00 pm the two requirements, R5 and R6, are triggered and start executing. However, R6 tries to open the windows but R5, which is a security requirement with higher priority, will override the action of R6 and will not allow it to open the windows. If the two requirements had the same priority, then this interaction would be described as non-determinism using SCR8 	

Scenario ID	SCR11	
Type of interaction	TwoInteractingDynamicBehaviourRequirements → Action–ActionInteractions → NegativeImpact _{C2 = SameTriggerEvents & SamePreStates}	
Detection guideline	IF {(R1.TriggerEvent = R2.TriggerEvent) AND (R1.PreState = R2.PreState) AND (R1.Action NEGATIVELY_IMPACTS R2.Action)} THEN {R1 interacts with R2 under the interaction type t11}	
Example	<ul style="list-style-type: none"> • R6 (revisited from SCR10): “Automatically open the windows in (LivingRoom) at time (11:00 pm)” • R7: “Increase/Decrease the temperature of the house to the temperature (22 °C) at time (11:00 pm)” • Interaction: When the time is 11:00 pm the two requirements are triggered and both of them start executing. Now, if the temperature outside the house is too cold or too hot then the action of R6 will negatively affect the action of R7 as R7 tries to increase/decrease the temperature of the house when the windows are opened 	

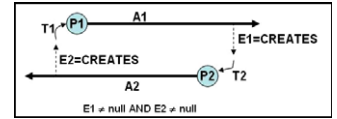
Scenario ID	SCR12
Type of interaction	TwoInteractingDynamicBehaviour Requirements → Action–ActionInteractions → Override _{C3 = LinkedTriggerEvents}
Detection guideline	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>② IF {(R1.TriggerEvent ~> R2.TriggerEvent) AND (R1.Action OVERRIDES R2.Action)} Then { R1 interacts with R2 under the interaction type t12}</p> </div> <div style="width: 45%;"> <p>① IF {(R1.TriggerEvent ~> R2.TriggerEvent) AND (R2.Action OVERRIDES R1.Action)} Then {R1 interacts with R2 under the interaction type t12}</p> </div> </div>
	<div style="display: flex; justify-content: space-around;">   </div>
Example	<ul style="list-style-type: none"> • R2 (revisited from SCR8): “Use the last stored audio level settings of the TV to adjust its volume when the TV is first turned on” • R8: “Completely shutdown power supply to all audio/video devices starting at (midnight) for (5 h)” • Interaction: Suppose that the TV was turned on just a few seconds before midnight. According to R2 the system will obtain the last stored audio level settings of the TV and starts adjusting its volume. But at midnight R8 starts executing and hence all audio/video devices including the TV are shutdown. Hence the action of R8 overrides the action of R2 before its completion

Scenario ID	SCR13
Type of interaction	TwoInteractingDynamicBehaviourRequirements → Action–ActionInteractions → NegativeImpact _{C3 = LinkedTriggerEvents}
Detection guideline	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>① IF {(R1.TriggerEvent ~> R2.TriggerEvent) AND (R1.Action NEGATIVELY_IMPACT R2.Action)} Then {R1 interacts with R2 under the interaction type t13}</p> </div> <div style="width: 45%;"> <p>② IF {(R1.TriggerEvent ~> R2.TriggerEvent) AND (R2.Action NEGATIVELY_IMPACT R1.Action)} Then { R1 interacts with R2 under the interaction type t13}</p> </div> </div>
	<div style="display: flex; justify-content: space-around;">   </div>
Example	<ul style="list-style-type: none"> • R6 (revisited from SCR10): “Automatically open the windows in (LivingRoom) at time (11:10 pm)” • R7 (revisited from SCR11): “Increase/decrease the temperature of the house to (22 °C) starting at (11:00 pm)” • Interaction: When the time is 11:00 pm, R7 is triggered and starts executing. Now, R7 tries to increase/decrease the house temperature to the value 22 °C. However this needs some time and meanwhile the time gets to 11:10 pm which triggers R6. Now R6 opens the windows and consequently negatively affecting the action of R7

Scenario ID	SCR14	
Type of interaction	TwoInteractingDynamicBehaviourRequirements → Action–ActionInteractions → Order _{C3 = LinkedTriggerEvents}	
Detection guideline	<p>IF {(SYSTEM_BEHAVIOUR _{R1.TriggerEvent ~> R2.TriggerEvent}) ≠ (SYSTEM_BEHAVIOUR _{R2.TriggerEvent ~> R1.TriggerEvent})}</p> <p>Then { R1 interacts with R2 under the interaction type t14}</p>	
Example	<ul style="list-style-type: none"> • R9: “The system shall support a one-click remote control 911 emergency service that calls the emergency centre and provides the home address and a pre-recorded message once a connection is established” • R10: “The system shall provide a regular telephone line with a set of telephony features (Three Way Calling) • Interaction: Suppose that an elderly resident A faces an emergency health condition (e.g., heart attack). A calls his son to take him to hospital but meanwhile, A’s condition gets worse so he uses R9 to call 911. Now R9 finds line is busy and it cannot execute 911 directly, so the system uses the Three Way Calling feature in R10 to put the son on hold and then connects to the emergency centre using 911. Consider this as the system behaviour B1 when Three Way Calling is activated first then 911 is followed later. Now, consider the same situation but at this time A uses R9 first to call 911 then tries to use the Three Way Calling feature in R10 to put 911 on hold and inform his son of the situation. In this case the system will not execute the Three Way Calling as the 911 service prevents anyone from putting it on hold. Consider this as system behaviour B2 when 911 executes first and then the Three Way Calling. Obviously B1 ≠ B2 because in B1 both Three Way Calling and 911 are executed successfully but in B2 only 911 is executed successfully 	

Scenario ID	SCR15	
Type of interaction	TwoInteractingDynamicBehaviourRequirements → Action–PreStateInteractions → Bypass _{C4 = LinkedTriggerEvents}	
Detection guideline	<p>IF {(R1.TriggerEvent ~> R2.TriggerEvent) AND (R1.Action Bypass R2.PreState)}</p> <p>Then {R1 interacts with R2 under the interaction type t15}</p>	
Example	<ul style="list-style-type: none"> • R5 (revisited from SCR10): “As a security measure, secure the doors and windows of a house by having them closed starting at time (11:00 pm) for (6 h)” • R13: When the intruder alarm is triggered, the security control unit is frozen and it can be unfrozen only by entering a PIN • Interaction: Suppose that R13 is triggered and starts executing. One part of R13’s action is to freeze the security control unit to prevent an intruder from disabling the alarm and prevent the opening of doors and windows to escape. Now this would bypass the pre-state of R5 and it will not allow the trigger event of R5 to trigger R5 simply because the whole security control unit including doors and windows is completely frozen (in another abnormal state). Therefore it can be said that R13 action bypasses the prestate of R5 	

Scenario ID	SCR16
Type of interaction	TwoInteractingDynamicBehaviourRequirements → TriggerEvent–TriggerEventInteractions → InfiniteLooping C5 = DualLinkedTriggerEvents
Detection guideline	IF {(R1.TriggerEvent <~~> R2.TriggerEvent) AND (R1.Action CREATES R2.TriggerEvent) AND (R2.Action CREATES R1.TriggerEvent)} Then {R1 interacts with R2 under the interaction type t16}
Example	<ul style="list-style-type: none"> • R4 (revisited from SCR9): “Increase the temperature inside the house to the preset temperature (22 °C) when temperature reading from thermostat is ≤ (20 °C)” • R12: “Open the windows in locations (X16 = LivingRoom and BedRoom) to decrease the temperature when the thermostat reading is ≥ (22) °C. Then close them again when the thermostat reading is ≤ (20) °C” • Interaction: Suppose that the house temperature is now at 24 °C then R12 is triggered and the windows are opened to decrease the temperature inside the house to 20 °C. Once the temperature is at 20 °C then the windows close but also R4 is triggered (i.e., the action of R12 dropped the temperature to 20 which means that it created the trigger event of R4). Now R4 starts executing and pumps hot air to increase the temperature back to 22. Once the temperature reaches 22 then R12 is triggered and starts executing again (i.e., the action of R4 created the trigger of R12 which is to have a temperature ≥ 22 °C). The preceding process repeats indefinitely. It is noted that the first requirement is created by a person who wants to keep the house temperature at 22 °C while the second requirement is created by someone who wants to keep the house temperature at 20 °C. This is understandable in a smart home with multi-occupants



4. Comparison of the proposed taxonomy to already existing taxonomies

4.1. Comparison results

In this section, we compare the proposed interaction taxonomy to the following already existing interaction taxonomies from the literature:

1. Feature interaction benchmark for Intelligent Networks—proposed by Cameron et al. in 1994 [4]. Note that Cameron et al. present in [4] two approaches for categorizing interactions which will be denoted in this paper by C-1 and C-2, respectively.
2. Interaction taxonomy for services of networked appliances—proposed by Kolberg et al. in 2003 [8] and denoted in this paper by K.
3. Interaction taxonomy for policies—proposed by Reiff Marganiec and Turner in 2004 [6] and denoted in this paper by R.

The three taxonomies mentioned above were chosen because they are cited frequently (first taxonomy) or there is close similarity with our proposed interaction taxonomy (second taxonomy) or they are very recent (third taxonomy). The comparison will be based on:

1. The method used for categorizing interactions (e.g., nature of interactions).
2. The main focus of the taxonomy (e.g., telecommunication telephony features).
3. The number of interaction categories proposed in each interaction taxonomy.
4. The number of examples presented to illustrate each interaction category.

Table 1
Comparing the proposed taxonomy to other existing taxonomies

	C		K	R	S
	C-1	C-2			
Method of categorization	Nature of interactions	Cause of interactions	Cause of interactions	Nature of interactions	Cause of interactions
Main focus	Telecommunications Intelligent Networks	Telecommunications Intelligent Networks	Smart homes networked devices	Policies	General with restriction on implementation interactions
Number of interaction categories	Five main categories	Three main Categories 12 subcategories	Four main categories	Five main categories 19 subcategories	Nine main categories 24 subcategories 37 types
Number of presented examples	22	22 (same ones used in C-1)	5	10	37
Number of examples addressed by our taxonomy	Addressed: 18 Missed: 4 (implementation interactions)	Addressed: 18 Missed: 4 (implementation interactions)	Addressed: 5 Missed: 0	Addressed: 10 Missed: 0	N/A

C: Cameron et al. taxonomy[17]; C-1: Cameron et al. taxonomy—first approach; C-2: Cameron et al. taxonomy—second approach; K: Kolberg et al. taxonomy [8]; R: Reiff-Marganic and Turner taxonomy [6]; S: Shehata et al. taxonomy (proposed taxonomy).

5. The number of presented examples that are addressed by our proposed interaction taxonomy and whether there are any examples that are missed and not addressed by our proposed interaction taxonomy.

Using the criteria mentioned above, the results of the comparison are summarized in Table 1.

4.2. Discussion

- Our proposed interaction taxonomy categorizes interactions according to the cause of interactions. This satisfies the objective of our proposed taxonomy that presents where, how, and why interactions occur. This is most beneficial in understanding the technical aspects rather than the social aspects of interactions and also facilitates the definition of detection guidelines for detecting any interactions between two requirements.
- Our proposed interaction taxonomy starts by categorizing interactions into high-level main interaction categories similar to other taxonomies. This helps provide a general high level understanding of the possible interactions.
- Our proposed interaction taxonomy is able to address all examples presented in other taxonomies except for four missed interactions found by the taxonomy of Cameron et al. Those four missed interactions are caused by the way the system was implemented and not by the requirements of the system and therefore they are not addressed by the proposed interaction taxonomy.
- The proposed interaction taxonomy has the limitation of not being able to address design or implementation interactions. The objective of the taxonomy is to address interactions at the requirements and early design stages of software systems. Hence, all implementation interactions are missed by the proposed interaction taxonomy. However, in the majority of cases most of the critical interactions manifest themselves during the requirements engineering phase of the software lifecycle. These interactions are captured by our proposed interaction taxonomy.
- The proposed interaction taxonomy has the advantage of being able to work in different domains with almost the same quality as domain-specific taxonomies. The taxonomy of Cameron et al. works very well to detect interactions between telecommunication features in the application layer of the OSI model. However, it does not work when investigating interactions between services of other layers of the OSI model or in other domains. Our taxonomy performs not only well in telecommunications but also in very different

domains. Case studies reported in [10,11,14–16] show that our taxonomy was successfully applied to detect interactions in the telecommunications domain, the control domain, web domain, and policy domain. Even if a new software domain needs to be inspected for interactions, our taxonomy can be applied.

- The proposed interaction taxonomy focuses mainly on detecting functional/behavioural interactions. It currently does not detect non-functional requirements (NFR) interactions. However, part of our future work is to expand the proposed taxonomy to be able to detect NFR interactions.

5. Conclusions

This paper addresses the problem of requirement interactions in software systems and presents a general taxonomy for identifying requirement interactions. In total, the proposed interaction taxonomy has 6 main interaction categories, 17 interaction subcategories, 29 interaction types, and 29 interaction scenarios. Each interaction scenario also has a detection guideline that can be used to detect the corresponding interaction type.

The proposed interaction taxonomy is novel in the following sense: It is a general taxonomy that can be applied in any domain rather than being oriented towards a specific domain. Hence it can be considered as the first domain-independent requirements interaction taxonomy. Also, the taxonomy provides 29 interaction scenarios that give a detailed description of when two requirements interact. The understanding of each interaction scenario is enhanced using a real life example that describes such an interaction. The 29 interaction scenarios also provide 29 detection guidelines that can be used by any developer to detect the different interaction types.

The proposed interaction taxonomy was compared to other existing taxonomies in the literature and not only was it able to address all the issues in those taxonomies but it also contained many other interaction types that have not been captured by other taxonomies.

Appendix A

A.1. Details of the main interaction category \odot “Two interacting system axioms”²

A.1.1. Interaction subcategories

The decomposition of the main interaction category \odot into subcategories was done based on the Rule attribute of system axioms (see Section 2). Therefore, the only possible pair of attributes that can form interaction subcategories is

- S1: Rule–Rule interactions:** This subcategory contains all the interactions that arise between two system axioms because the Rule attribute of the first system axiom interacts with the Rule attribute of the second system axiom. This subcategory is shown in the second layer of the proposed taxonomy shown in Fig. 6.

A.1.2. Interaction types

- t1: Override:** Consider the two system axioms R1 and R2. Suppose that the rule of R1 overrides and cancels the rule of R2. Hence R1 and R2 interact.
- t2: Negative impact:** Consider R1 and R2 to be system axioms. Suppose that the rule of R1 negatively impacts the rule of R2. Hence, R1 and R2 interact according to the interaction type t2. This interaction type is similar to t1; however, the difference is that in t1 the rule of R1 will completely cancel and override the rule of R2 while in t2 the rule of R1 will only negatively impact, but not completely cancel, the rule of R2.

² The examples of interaction scenarios in this category are taken from a web e-commerce system which is a representative of the web domain.

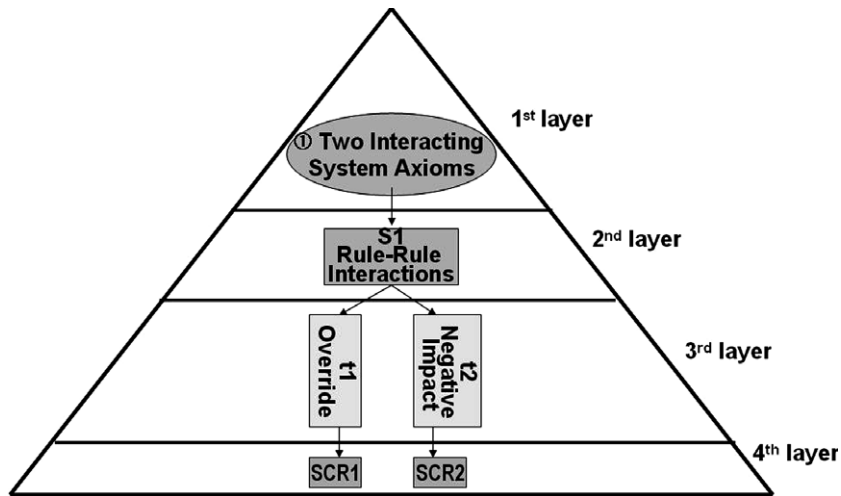


Fig. 6. Details of the main interaction category ①.

A.1.3. Interaction scenarios

Scenario ID	SCR1
Type of interaction	TwoInteractingSystemAxioms → Rule–RuleInteractions → Override
Detection guideline	IF {(R1.Rule OVERRIDES R2.Rule)} THEN {R1 interacts with R2 under the t1 interaction type}
Example	<ul style="list-style-type: none"> • R1: “The library page on the website shall use secure logon for members only using user-name and password” • R2: “All webpages on the website are accessible with no more than two clicks from the menu bar” • Interaction: What happens if a user who is not logged in, wants to go to the library page? In this case the security requirement R1 overrides R2 and redirects him to a sign in page. This means that the user, assuming he is a member, needs more than two clicks to go to the library page

Scenario ID	SCR2
Type of interaction	TwoInteractingSystemAxioms → Rule–RuleInteractions → NegativeImpact
Detection guideline	IF {(R1.Rule NEGATIVELY_IMPACTS R2.Rule)} THEN {R1 interacts with R2 under the t2 interaction type}
Example	<ul style="list-style-type: none"> • R1: “There shall be an input acceptability checking mechanism X to validate the input data before the system exhibits any response” • R2: “The response time of the system should not exceed 3.0 s” • Interaction: What happens if the input acceptability mechanism X is set to a very complex mechanism? This will cause the system response time to increase which negatively impacts R2

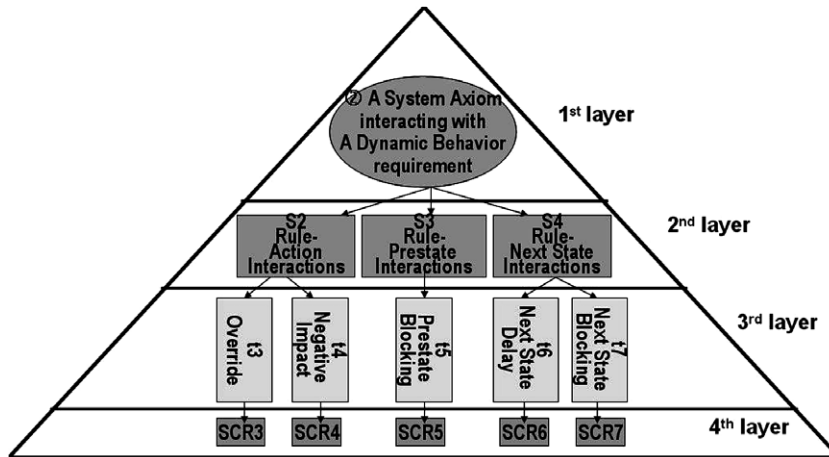


Fig. 7. Details of the main interaction category ②.

A.2. Details of main interaction category ② “A system axiom interacting with a dynamic requirement”³

A.2.1. Interaction subcategories

The main interaction category ② describes interactions between a system axiom and a dynamic behaviour requirement. Therefore, to derive all the interaction subcategories from it, all possible pairs of attributes of a system axiom and a dynamic behaviour requirement are examined to determine which ones can form interaction subcategories. The examination resulted in the following subcategories (which are shown in the second layer of the taxonomy in Fig. 7):

- S2: Rule–Action interactions:** This subcategory contains all interactions between a system axiom and a dynamic requirement because Rule attribute of the system axiom interacts with Action attribute of the dynamic requirement.
- S3: Rule–Prestate interactions:** This subcategory contains all interactions between a system axiom and a dynamic requirement because Rule attribute of the system axiom interacts with Prestate attribute of the dynamic requirement.
- S4: Rule–Next state interactions:** This subcategory contains all interactions between a system axiom and a dynamic requirement because Rule attribute of system axiom interacts with Next state attribute of the dynamic requirement.

A.2.2. Interaction types

- t3: Override:** Consider R1 to be a system axiom and R2 to be a dynamic behaviour requirement. Suppose that the rule of R1 overrides and cancels the Action of R2 before its completion. Hence R2 was not successfully executed. This is a negative relationship between R1 and R2 and hence they interact.
- t4: Negative impact:** Suppose that the rule of R1 negatively impacts the Action of R2 which reduces the efficiency of the execution of the action of R2. Hence, R1 and R2 interact.
- t5: Prestate blocking:** Consider R1 to be a system axiom and R2 to be a dynamic behaviour requirement. Suppose that the rule of R1 prevents the system from being able to be in a state which is the same as the prestate of R2. This means that R2 will never be executed since the system cannot be in its prestate. This is a negative relationship between R1 and R2 and hence they interact.
- t6: Next state delay:** Suppose that the rule of R1 delays the system to reach the next state of R2. This means that R2 has been delayed in finishing and making the system reach the next state specified in R2. This is a negative relationship between R1 and R2 and hence they interact.

³ The examples of interaction scenarios in this category are taken from the lift system and the smart homes system.

t7: Next state blocking: Suppose that the rule of R1 completely prevents the system from reaching the next state of R2. This means that the next state of R2 has been blocked. Hence R1 and R2 interact.

A.2.3. Interaction scenarios

Scenario ID	SCR3
Type of interaction	SystemAxiomInteractingWithDynamicBehaviourRequirement → Rule–ActionInteractions → Override
Detection guideline	IF {(R1.Rule OVERRIDES R2.Action)} THEN {R1 interacts with R2 under the t3 interaction type}
Example	<ul style="list-style-type: none"> • R1: “The max temperature of hot water from the boiler is 45 °C in order to keep the boiler in safe operation” • R2: “Increase the temperature of the hot water to 55 °C in the washing machine when the machine starts operating” • Interaction: The Rule of R1 overrides the action of R2 and does not allow the increase of temperature to 55 °C
Scenario ID	SCR4
Type of interaction	SystemAxiomInteractingWithDynamicBehaviourRequirement → Rule–ActionInteractions → NegativeImpact
Detection guideline	IF {(R1.Rule NEGATIVELY_IMPACTS R2.Action)} THEN {R1 interacts with R2 under the t4 interaction type}
Example	<ul style="list-style-type: none"> • R1 “Executive floor calls are of highest priority” • R2 “The lift is called by pressing the call button and it should arrive within 2 min otherwise an alternative car is assigned to that floor” • Interaction: If there are calls from the executive floor then the arrival of the lift is delayed until all calls from the executive floor are served. Hence the rule of R1 has negatively affected the action of R2 by delaying the arrival of the lift
Scenario ID	SCR5
Type of interaction	SystemAxiomInteractingWithDynamicBehaviourRequirement → Rule–PreStateInteractions → PreStateBlocking
Detection guideline	IF {(R1.Rule BLOCKS R2.PreState)} THEN {R1 interacts with R2 under the t5 interaction type}
Example	<ul style="list-style-type: none"> • R1(maintenance) “To avoid system problems, the lift has to be maintained on a monthly basis” • R2(operation) “When the lift is on standby at floor k with doors closed and it receives a call from floor K, it opens its doors” • Interaction: What happens when the lift is maintained while being at floor K and someone calls the lift from floor K? It will not open its doors because the power is disconnected during maintenance to prevent accidents. Hence the rule of R1 has prevented and blocked the lift from being in standby, which is the prestate of R2

Scenario ID	SCR6
Type of interaction	SystemAxiomInteractingWithDynamicBehaviourRequirement → Rule–NextStateInteractions → NextStateDelay
Detection guideline	IF {(R1.Rule DELAYS R2.NextState)} THEN {R1 interacts with R2 under the t6 interaction type}
Example	<ul style="list-style-type: none"> • R1 (Operation) “Executive floor calls always has highest priority” • R2 (Operation) “When the lift passes by floor K and there is a call from this floor, the lift will stop at floor K” • Interaction: What happens when the lift is passing by floor K and there is a call from floor K but there are also five calls from the executive floors? In this case, the next state of R2 will not be reached which is to stop at floor K until all executive calls are served. Hence the rule of R1 has delayed the next state of R2
Scenario ID	SCR7
Type of interaction	SystemAxiomInteractingWithDynamicBehaviourRequirement → Rule–NextStateInteractions → NextStateBlocking
Detection guideline	IF {(R1.Rule BLOCKS R2.NextState)} THEN {R1 interacts with R2 under the t7 interaction type}
Example	<ul style="list-style-type: none"> • R1 “For a lift at floor K, the lift doors must close after a maximum of 1 min” • R2 “When something blocks lift doors, the lift interrupts the process of closing the doors and reopens them” • Interaction: If a user keeps blocking the lift doors then after a 1 min the rule of R1 is enforced and prevents R2 from being able to reach its next state which is “Doors opened”

A.3. Details of main interaction category ④ “A system axiom interacting with a resource”⁴

A.3.1. Interaction subcategories

The subcategories of interactions derived from main interaction category ④ are shown in Fig. 8 and described in the following:

S9: Rule–Availability interactions: This subcategory contains all the interactions between a system axiom and a resource because the Rule attribute of the system axiom interacts with the Availability attribute of the resource.

S10: Rule–Performance interactions: This subcategory contains all the interactions between a system axiom and a resource because the Rule attribute of the system axiom interacts with the Performance attribute of the resource.

S11: Rule–Interface interactions: This subcategory contains all the interactions between a system axiom and a resource because the Rule attribute of the system axiom interacts with the Interface attribute of the resource.

A.3.2. Interaction types

t17: Failure of resource: Consider R1 to be a system axiom and R2 to be a resource. If the Rule attribute of the system axiom causes the resource to fail such that it violates the Availability attribute of R2, then this is considered an interaction.

⁴ The examples of interaction scenarios in this category are taken from the web e-commerce system which is a representative of the web domain.

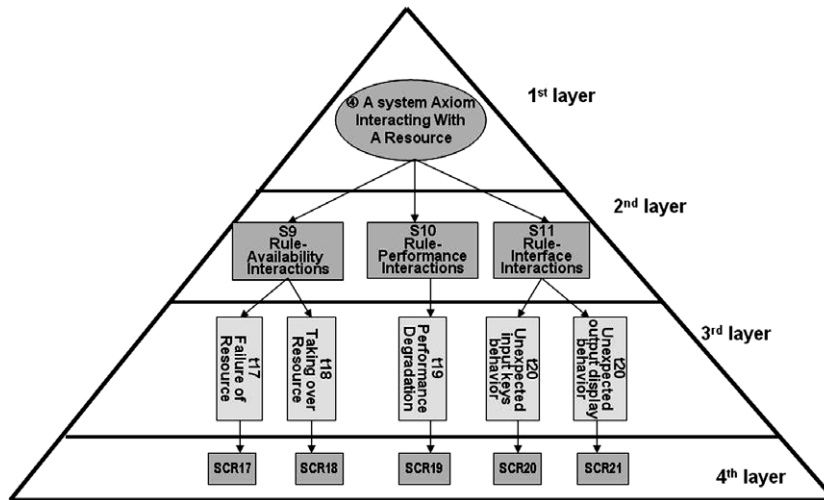


Fig. 8. Details of the main interaction category ④.

- t18: Taking over resource:** If the Rule attribute of the system axiom R1 takes over the resource and does not allow any other usage of it by other requirements in a manner that violates the Availability attribute of R2, then this is considered to be an interaction between R1 and R2.
- t19: Performance degradation:** In this interaction type, the Rule attribute of the system axiom causes degradation of the performance of the resource such that it violates the Performance attribute of R2. This is considered as interaction between R1 and R2.
- t20: Unexpected input key behaviour:** This interaction type addresses situations in which the Rule attribute of R1 causes the user to use the input interface in a manner that violates the Interface attribute of R2.
- t21: Unexpected output key behaviour:** This interaction type addresses situations in which the Rule attribute of R1 causes the display of output data to the user in a manner that would violate the Interface attribute of R2.

A.3.3. Interaction scenarios

Scenario ID	SCR17
Type of interaction	SystemAxiomInteractingWithResource → Rule–AvailabilityInteractions → FailureOfResource
Detection guideline	IF {(R1.Rule violates Resource.Availability) AND (R1.Rule LEADS_TO_FAILURE Resource.Availability)} THEN {R1 interacts with Resource under t17 interaction type}
Example	<ul style="list-style-type: none"> • R1 “The website shall be able to handle a maximum of (50 hits/s)” • Resource.Availability: “The Application server must be able to process at least 99.9% of all requests made during a week” • Interaction: If the website receives a heavy load, say 100 hits/s, this might cause the application server to fail. This is because the website was not designed to handle such an amount of requests. If this occurred frequently then the rule of R1 caused the failure rate of the application server to exceed the constraint stated in resource availability

Scenario ID	SCR18
Type of interaction	SystemAxiomInteractingWithResource → Rule–AvailabilityInteractions → TakingOverResource
Detection guideline	IF {(R1.Rule violates Resource.Availability) AND (R1.Rule LEADS_TO_TAKING_OVER Resource.Availability)} THEN {R1 interacts with Resource under t18 interaction type}
Example	<ul style="list-style-type: none"> • R1: “The system shall use X database server to store and retrieve data” • Resource.Availability “The database server must be able to process at least 99.9% of all requests made during a week” • Interaction: If the database server used is an old database server, then it might only be able to handle few requests simultaneously. Every time an application server sends a few requests to the database server, the database server gets busy and cannot handle new requests. If there are many application servers accessing this database server then it is unavailable for other application servers
Scenario ID	SCR19
Type of interaction	SystemAxiomInteractingWithResource → Rule–PerformanceInteractions → PerformanceDegradation
Detection guideline	IF {(R1.Rule violates Resource.Performance) AND (R1.Rule LEADS_TO_PERFORMANCE_DEGRADATION Resource.Performance)} THEN {R1 interacts with Resource under t19 interaction type}
Example	<ul style="list-style-type: none"> • R1: “The system shall use X techniques for the encryption of transmitted financial data” • Resource.Performance “The response time of the application server is less than 3 s” • Interaction: Assume that X is a very complex encryption algorithm. Every time a user tries to submit financial data, the server must encrypt the data using X and since X is very complex its performance is degraded for other requests. This might cause the response time of the application sever to exceed the 3 seconds limit required
Scenario ID	SCR20
Type of interaction	SystemAxiomInteractingWithResource → Rule–InterfaceInteractions → UnexpectedInputKeysBehaviour
Detection guideline	IF {(R1.Rule violates Resource.Interface) AND (R1.Rule LEADS_TO_UNEXPECTED_INPUT_BEHAVIOUR Resource.Interface)} THEN {R1 interacts with Resource under t20 interaction type}
Example	<ul style="list-style-type: none"> • R1: “The user can accept incoming calls on the net phone using (pressing number nine key) technique” • Resource.Interface “A standard input interface is provided as the net phone interface” • Interaction: If there is an incoming call and the user is not familiar with this technique, the user might press the regular keys to accept new calls but it will not work so he might try different keys which might result in unexpected termination of the incoming call

Scenario ID	SCR21
Type of interaction	SystemAxiomInteractingWithResource → Rule–InterfaceInteractions → UnexpectedOutputDisplayBehavior
Detection guideline	IF {(R1.Rule violates Resource.Interface) AND (R1.Rule LEADS_TO_UNEXPECTED_OUTPUT_DISPLAY Resource.Interface)} THEN {R1 interacts with Resource under t21 interaction type}
Example	<ul style="list-style-type: none"> • R1: “The user is notified by incoming calls on his computer screen using (switch the focus to the net phone incoming message interface and keeps the user there until he provides a response) technique” • Resource.Interface “A standard output interface is provided as the net phone interface” • Interaction: If the user is playing a game on the screen and there is an incoming call then the focus is switched automatically to the net phone and the user loses the game he is playing (which sometimes might be more important for the user than the incoming call) which results in unexpected display behaviour

A.4. Details of main interaction category © “A dynamic behaviour requirement interacting with a resource”⁵

A.4.1. Interaction subcategories

The subcategories of interactions that are derived from the main interaction category © are shown in Fig. 9 and are described in the following:

S12: Action–Availability interactions: This subcategory contains all the interactions between a dynamic requirement and a resource because the Action attribute of the dynamic requirement interacts with the Availability attribute of the resource.

S13: Action–Performance interactions: This subcategory contains all interactions between a dynamic requirement and a resource because the Action attribute of the dynamic requirement interacts with Performance attribute of the resource.

S14: Action–Interface interactions: This subcategory contains all the interactions between a dynamic requirement and a resource because the Action attribute of the dynamic requirement interacts with the Interface attribute of the resource.

A.4.2. Interaction types

t22: Failure of resource: Consider R1 to be a Dynamic Behaviour Requirement and R2 to be a resource. If the Action attribute of the dynamic requirement causes failure to the resource such that it violates the Availability attribute of R2, then this is considered to be an interaction according to t22.

t23: Taking over resource: If the Action attribute of the dynamic requirement R1 takes over the resource and does not allow any other usage of it by other requirements in a manner that violates the Availability attribute of R2, then this is considered to be an interaction between R1 and R2.

t24: Performance degradation: In this interaction type, the Action attribute of the dynamic requirement causes degradation to the performance of the resource such that it violates the Performance attribute of R2. This is considered an interaction between R1 and R2.

t25: Unexpected input key behaviour: This interaction type addresses the situations when the Action attribute of R1 can cause the user to use the input interface in a manner that would violate the Interface attribute of R2.

t26: Unexpected output key behaviour: This interaction type addresses the situations when the Action attribute of R1 causes the display of output data to the user through a manner that would violate the Interface attribute of R2.

⁵ The examples of interaction scenarios in this category are taken from smart home system which is a representative of the policy domain.

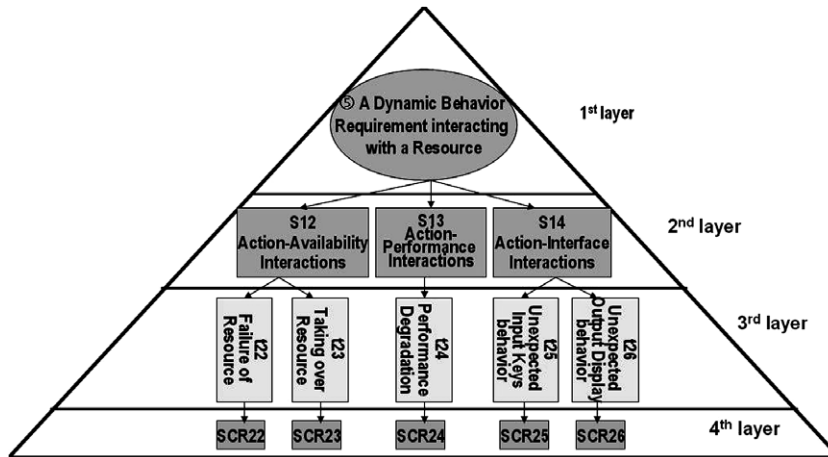


Fig. 9. Details of the main interaction category 5.

A.4.3. Interaction scenarios

Scenario ID	SCR22
Type of interaction	DynamicBehaviourRequirementInteractingWithResource → Action-AvailabilityInteractions → FailureOfResource
Detection guideline	IF {(R1.Action violates Resource.Availability) AND (R1.Action LEADS_TO_FAILURE Resource.Availability)} THEN {R1 interacts with Resource under t22 interaction type}
Example	<ul style="list-style-type: none"> • R1 “When the electricity consumption exceeds X kW/h, start shutting down devices A, then B, then C, then D in this order until consumption reaches Y kW/h” • Resource.Availability “The boiler shall be available more than 99.9% during each week” • Interaction: Assume that the boiler is device C in R1. If X is set to a small number then the system will likely shutdown the boiler several times in order to maintain the consumption rate. This will violate the resource availability constraint

Scenario ID	SCR23
Type of interaction	DynamicBehaviourRequirementInteractingWithResource → Action-AvailabilityInteractions → TakingOverResource
Detection guideline	IF {(R1.Action violates Resource.Availability) AND (R1.Action LEADS_TO_TAKING_OVER Resource.Availability)} THEN {R1 interacts with Resource under t23 interaction type}
Example	<ul style="list-style-type: none"> • R1: “During vacation, the vacation control system shall imitate the sound of occupants between times A and B using the TV” • Resource.Availability “The A/V devices are available during daytime for personal use” • Interaction: R1 causes the unavailability of TV between A and B and this violates the resource availability constraint on having the TV available during daytime for personal use such as recording a show while away. In this case the TV is unavailable as it cannot do the two things at the same time

Scenario ID	SCR24
Type of interaction	DynamicBehaviourRequirementInteractingWithResource → Action–PerformanceInteractions → PerformanceDegradation
Detection guideline	IF {(R1.Action violates Resource.Interface) AND (R1.Action LEADS_TO_DEGRADATION Resource.Performance)} THEN {R1 interacts with Resource under t24 interaction type}
Example	<ul style="list-style-type: none"> • R1 “The user can set the CD player to play stream audio tracks from the internet between times A and B using (Dialup) connection” • Resource.Performance “Audio/Video devices perform using high definition quality standards” • Interaction: The dialup connection does not provide a reliable, high-speed connection. Hence, in this case the action of R1 shall affect the performance of the CD player and violates the resource performance constraints
Scenario ID	SCR25
Type of interaction	DynamicBehaviourRequirementInteractingWithResource → Action–InterfaceInteractions → UnexpectedInputKeysBehavior
Detection guideline	IF {(R1.Action violates Resource.Interface) AND (R1.Action LEADS_TO_UNEXPECTED_INPUT_BEHAVIOUR Resource.Interface)} THEN {R1 interacts with Resource under t25 interaction type}
Example	<ul style="list-style-type: none"> • R1 “To dial a voice activated number, user must pick the handset, press key (number key, which is an unusual input key in this case) and say the voice sample of the desired number” • Resource.Interface “Standard input interface is provided for the smart home phone interface” • Interaction: “Assume a user picks up the handset and presses this key number, the telephone will not know if this number is part of a dialled number or if it should activate the voice dialing system and hence this input might result in unexpected behaviour”
Scenario ID	SCR26
Type of interaction	DynamicBehaviourRequirementInteractingWithResource → Action–InterfaceInteractions → UnexpectedOutputDisplayBehavior
Detection guideline	IF {(R1.Action violates Resource.Interface) AND (R1.Action LEADS_TO_UNEXPECTED_OUTPUT_DISPLAY Resource.Interface)} THEN {R1 interacts with Resource under t26 interaction type}
Example	<ul style="list-style-type: none"> • R1 “When the user is talking on the phone, Alert him 5 s before the end of every minute using (displaying a warning on the screen of the telephone set) technique” • Resource.Interface “A standard output interface is provided for the smart home phone interface” • Interaction: Consider a user who is storing a phone number while talking with someone on the phone. When the minute is about to finish (55 s), the system alerts the user and causes him to lose all his data because of the unexpected display behaviour which switches the normal screen to display the call time

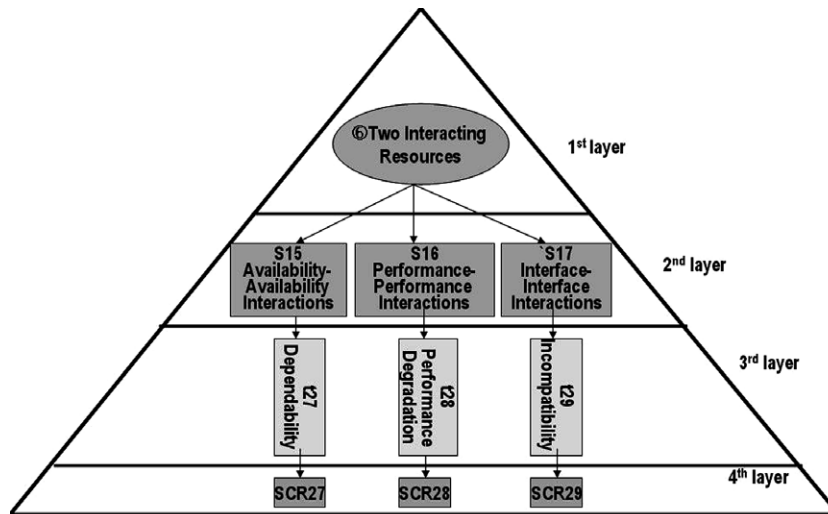


Fig. 10. Details of the main interaction category ⑥.

A.5. Details of the main interaction category ⑥ “Two interacting resources”⁶

A.5.1. Interaction subcategories

Fig. 10 shows how the sixth main interaction category ⑥ from the first layer of the taxonomy is decomposed into interaction subcategories in the second layer. After analyzing the possible pairs of attributes that can form interaction categories, only the following pairs were found to really represent interactions subcategories:

- S15: Availability–Availability interactions:** This subcategory contains all the interactions between two resources because the Availability attribute of the first resource interacts with the Availability attribute of the second resource.
- S16: Performance–Performance interactions:** This subcategory contains all the interactions between two resources because the Performance attribute of the first resource interacts with the Performance attribute of the second resource.
- S17: Interface–Interface interactions:** This subcategory contains all the interactions between two resources because the Interface attribute of the first resource interacts with the Interface attribute of the second resource.

A.5.2. Interaction types

- t27: Dependability:** Suppose R1 and R2 to be resources. If the Availability of R2 depends on the availability of R1 in such a manner that the failure of the resource R1 will cause the failure of the second resource R2, then the two system resources R1 and R2 interact.
- t28: Performance degradation:** If the performance attribute of resource R1 is linked with the performance attribute of R2 such that the performance of R1 can affect and degrade the performance of R2, then the two resources R1 and R2 interact.
- t29: Incompatibility:** If the interface attribute of R1 is incompatible with the interface attribute of R2 and the two resource requirements are related to two resources that communicate between each other, then the two requirements R1 and R2 interact.

⁶ The examples of interaction scenarios in this category are taken from smart home system which is a representative of the policy domain.

A.5.3. Interaction scenarios

Scenario ID	SCR27
Type of interaction	TwoInteractingResources → Availability–AvailabilityInteractions → Dependability
Detection guideline	IF {(Resource1.Availability DEPENDS_ON Resource2.Availability)} THEN {Resource1 interacts with Resource2 under t27 interaction type}
Example	<ul style="list-style-type: none"> • Resource1:Availability “The natural gas boiler shall be available more then 99.9% every year” • R2: “The natural gas regulator shall be available 100% every year” • Interaction: If the natural gas regulator fails, i.e., becomes unavailable, for any reason and the natural gas is being blocked then the boiler is not working and hence becomes also unavailable

Scenario ID	SCR28
Type of interaction	TwoInteractingResources → Performance–PerformanceInteractions → PerformanceDegradation
Detection guideline	IF {(Resource1.Performance LEADS_TO_DEGRADATION Resource2.Performance)} THEN {Resource1 interacts with Resource2 under t28 interaction type}
Example	<ul style="list-style-type: none"> • Resource1:Performance “The (X = T1) Network Card used to connect to the internet provides best performance for connection speed” • Resource2:Performance “Audio/devices performs using high definition quality standards” • Interaction: The performance of a CD player, which plays stream audio from the internet, is related to the performance of the T1 card. If the T1 card performance is degraded for any reason (e.g., lost connection, paths congestion) then the CD performance is also degraded

Scenario ID	SCR29
Type of interaction	TwoInteractingResources → Interface–InterfaceInteractions → Incompatibility
Detection guideline	IF {(Resource1.Interface INCOMPATIBLE_WITH Resource2.Interface)} THEN {Resource1 interacts with Resource2 under t29 interaction type}
Example	<ul style="list-style-type: none"> • Resource1.Interface “The TV has a interface compatible with the smart home communication protocol” • R2: “The VCR has KONNEX, which is a smart home communication protocol, compatible interface” • Interaction: Obviously the two resources have incompatible interfaces and they cannot communicate directly with each other

References

- [1] M. Shehata, L. Jiang, A. Eberlein, A requirements interaction detection process guide, presented at Canadian Conference on Electrical and Computer Engineering 2004, 2–5 May 2004, Niagara Falls, Ont., Canada, 2004.

- [2] W. Robinson, S. Pawlowski, V. Volkov, Requirements Interaction Management, GSU CIS Working Paper 99–7, Georgia State University, Atlanta, GA, August 1999.
- [3] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, M. Goedicke, Viewpoints: a framework for integrating multiple perspectives in system development, *International Journal of Software Engineering and Knowledge Engineering* 2 (1992) 31–58.
- [4] E. Cameron, A feature interaction benchmark for IN and beyond, *A Feature Interaction Benchmark for IN and Beyond*, in: E.J. Cameron (Ed.), *Feature Interactions in Telecommunications Systems*, IOS Press, 1994, pp. 1–23.
- [5] P. Gibson, G. Hamilton, D. Mery, A Taxonomy for Triggered Interactions using Fair Object Semantics, in: M. Calder, E. Magill (Eds.), *Feature Interactions in Telecommunications and Software Systems*, IOS Press, 2000, pp. 193–209.
- [6] S. Reiff-Marganiec, K.J. Turner, Feature interaction in policies, *Computer Networks* 45 (2004) 569–584.
- [7] N. Gorse, The Feature Interaction Problem: Automated filtering of Incoherences and Generation of Validation Test suites at the Design Stage, University of Ottawa, Ottawa, Ont., Canada, 2001.
- [8] M. Kolberg, E.H. Magill, M. Wilson, Compatibility issues between services supporting networked appliances, *IEEE Communications Magazine* 41 (2003) 136–147.
- [9] M. Frappier, A. Mili, J. Desharnais, Defining and detecting feature interactions, in: *Algorithmic Languages and Calculi*, 1997.
- [10] M. Shehata, A. Eberlein, A.O. Fapojuwo, Feature Interactions between Networked Smart Home Appliances, in: *QSSE, 4th ASERC Workshop on Quantitative and Soft Computing Based Software Engineering*, Banff, Alberta, Canada, February 16–17, 2004.
- [11] M. Shehata, A. Eberlein, Requirements interaction detection using semi-formal methods, in: *Proceedings of the 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems. ECBS 2003*, Huntsville, AL, USA, April 7–10, 2003.
- [12] M. Shehata, A. Eberlein, A. Fapojuwo, IRIS: a semi-formal approach for detecting requirements interactions, in: *Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, Brno, Czech Republic, May 24–27, 2004.
- [13] D. Briere, P. Hurley, *Smart Homes for Dummies*, Hungry Minds, Inc., New York, 1999.
- [14] M. Shehata, A. Eberlein, A. Fapojuwo, C. Guerrero, C. Juiz, R. Puigjaner, Using IRIS to Detect Interactions in The Web Domain: A case study in the TPC-W Benchmark, Technical Report, RE Group, Department of Electrical and Computer Engineering, University of Calgary, Calgary, November, 2003.
- [15] M. Shehata, A. Eberlein, A. Fapojuwo, The use of semi-formal methods for detecting requirements interactions, in: *SE 2004, The IASTED International Conference on Software Engineering*, Innsbruck, Austria, February 17–19, 2004.
- [16] M. Shehata, A. Eberlein, A. Fapojuwo, Using Semi-Formal Methods For Detecting Interactions Among Smart Homes Policies, *International Journal of Science of Computer Programming*, in press.
- [17] E.J. Cameron, N.D. Griffith, Y.-J. Ling, M.E. Nilson, W.K. Schnure, H. Velthuisen, A feature interaction benchmark for IN and beyond, in: *Proceedings of 2nd International Workshop on Feature Interactions in Telecommunications Software Systems*, Amsterdam, Netherlands, 1994.



Mohamed Shehata received his Ph.D. from the Department of Electrical and Computer Engineering, University of Calgary, Canada. Currently, he is an assistant professor at the Department of Electrical and Computer Engineering, Shoubra Faculty of Engineering, Benha University, Egypt. He previously worked as a Post-Doctor Fellow at LIVS, the University of Calgary directing a project funded by the City of Calgary, Alberta Infrastructure and Transportation, and Transport Canada. He received B.Sc. and M.Sc. from Zagazig University, Egypt in 1996 and 2001, respectively. During the period from 1997 until 2001, he was an instructor in the Department of Electrical Engineering—Shoubra Faculty of Engineering—Zagazig University. Also he worked during that time as a part time software engineer at the Egyptian Company of Technology where he lead and participated in developing several software systems.



Armin Eberlein received the Dipl-Ing (FH) degree in telecommunications engineering at the Mannheim University of Applied Sciences in Germany, the M.Sc. degree in communications systems, and the Ph.D. degree in software engineering from the University of Wales, Swansea, UK. He is an Associate Professor and is currently on leave from the Department of Electrical and Computer Engineering at the University of Calgary, Canada, where he served as a codirector of the Alberta Software Engineering Research Consortium (ASERC) and as the director of the Software Engineering Program. He spends his leave in the Computer Engineering Department of the American University of Sharjah in the United Arab Emirates. His research interests focus on the improvement of requirements engineering practices and techniques. He has worked previously as a hardware and software developer at Siemens in Munich, Germany, and has consulted for various companies in Germany, the UK, and Canada.



Abraham O. Fapojuwo received the B.Eng. degree (first class honors) from the University of Nigeria, Nsukka, in 1980 and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Calgary, Calgary, AB, Canada, in 1986 and 1989, respectively. From 1990 to 1992, he was a Research Engineer with NovAtel Communications Ltd., where he performed numerous exploratory studies on the architectural definition and performance modeling of digital cellular systems and personal communications systems. From 1992 to 2001, he was with Nortel Networks, where he conducted, led and directed system-level performance modeling and analysis of wireless communication networks and systems. In January 2002, he joined the Department of Electrical and Computer Engineering, University of Calgary, as an Associate Professor. He is also an Adjunct Scientist at *TRLabs*, Calgary. His current research interests include protocol design and analysis for future generation wireless communication networks and systems, and best practices in software reliability engineering and requirements engineering. Dr. Fapojuwo is a registered Professional Engineer in the Province of Alberta, Canada.