

## Specifying Use Case Interaction: Clarifying Extension Points and Rejoin Points

**Pierre Metz and John O'Brien**, Dept. of Mathematics & Computing, Cork Institute of Technology, Ireland

**Wolfgang Weber**, Dept. of Computer Science, Darmstadt University of Applied Sciences, Germany

### Abstract

This article is the second work in a series of papers on advancing specification and modelling techniques for goal-based use cases. In this article we highlight major and remaining problems associated with the true semantics of extension points and rejoin points, which are used as branching and return locations for a use case's alternative interaction courses, and also for the application of UML's Extend-relationship. Based on a previous work that identified and defined three reconciling types of alternative courses of use case interaction, this article proposes clear and simple semantics for extension points and rejoin points that significantly reduce use case interaction ambiguity. These semantics allow the uniform treatment of any type of alternative course, so that a practitioner can concentrate on the coherence of the use case's goal, precondition and business results. Our proposal is independent of the way the use case interaction is documented, i.e. textually or graphically. Finally, our suggested semantics will dramatically contribute to an understanding of practical use case interaction specification.

### PAPER ROADMAP

We will highlight the ambiguity that surrounds the current semantics of extension points and rejoin points for specifying alternative courses of use case interaction. In order to address this issue, we propose that the interaction steps, within the branched interaction course that are referenced by the extension point, are always excluded from a use case scenario, while the interaction steps within the rejoined interaction course that are referenced by a rejoin point are always included into a use case scenario. Relevant use case fundamentals and terms are reviewed, briefly. A definition of the inconsistently-used

term “use case scenario” is also presented. Existing problems involving the use of extension points and rejoin points are highlighted. Finally, we present a solution to the raised issues, and outline the implications for UML’s use case foundation.

## 1 BRIEF REVIEW OF USE CASE BASICS

### Actors, Stakeholders, Goals, Use Cases and Use Case Interaction

A use case, as considered here, is the black-box requirements description of a single software-supported business goal of an actor or stakeholder. It must provide a corresponding business result of value to at least one actor or stakeholder. The actor instances need to participate in a use case execution. This they achieve by interacting with the system in order to make the system accomplish the use case goal and, thus, deliver the use case business result instances.

Example 1:

Use case goal: Withdraw Cash (ATM)

1. The customer inserts the card.
2. The system validates the card.
3. The Customer enters the PIN.
4. The system validates the PIN.
5. The customer enters the desired amount of money.

...

Use case business result:

The money has been dispensed and debited from the customer’s account. The card has been ejected. All failures have been logged. Transaction date and time have been logged.

Now consider the following alternative course:

- 4a. PIN has been incorrectly entered for the first or the second time:

- 4a1. The system logs the attempt.
  - 4a2. The system notifies the customer.
- Rejoin at 3.

The description of all possible use case interactions encompasses, generally, two parts: a text block describing the “main” sequence of interaction which always shows the use case goal succeeding; this is called the Basic Course and is also known as the “happy path”. Zero or more text blocks describe alternative courses of interaction to the basic course,

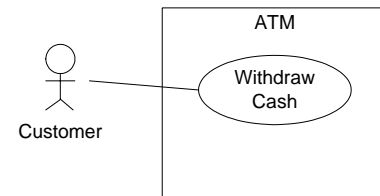
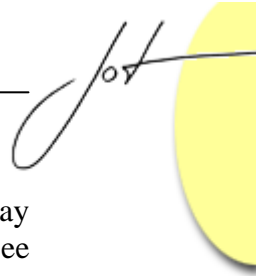


Figure 1. Use Case Diagram for Example 1



e.g. optional interactions and interactions handling business error recovery which may end in the use case goal succeeding or failing, are called Alternative Courses (see Example 1). These text blocks, being optional, must be guarded by a business condition [1], [5], [7], [9], [10], [15].

The reference to the label of an interaction step of a branched interaction course, where an alternative course branches off, is called an Extension Point. Correspondingly, a Rejoin point is a reference to the label of an interaction step in the base interaction course that represents the return location (see Example 1). It should be noted that, originally, UML introduced the term extension point to indicate branching locations for the Extend-relationship only [13]; an Extend-relationship is used to attach optional behaviour, that resides in an extension use case, to its base use case. However, we consider the term extension point to be also applicable to branching points of alternative courses that are text blocks residing within the same base use case document.

## Use Case Scenarios

In order to fully comprehend the issues raised and the examples presented in this paper, the term “use case scenario” needs to be clearly understood.

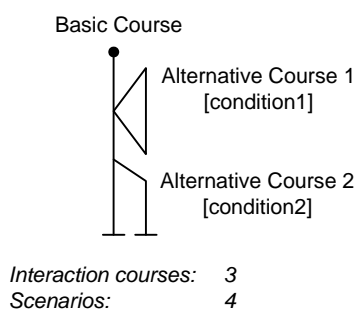


Figure 2. Illustration of the term “scenario”

A use case scenario is a single path through the use case’s interaction courses. Figure 1 illustrates three interaction courses, i.e. one basic and two alternative courses. In fact, there are four scenarios: the basic course only, the basic course plus alternative course 1, the first part of the basic course and alternative course 2, and basic course with both alternative courses 1 and 2. Cockburn [5] refers to the basic course as the “main success scenario” because, if never branched by an alternative course, the basic course directly shows the use case goal succeeding.

## 2 PROBLEMS IDENTIFIED WITH THE SEMANTICS OF EXTENSION POINTS AND REJOIN POINTS

The remaining and confusing problem with extension points and rejoin points in both literature and practice is that it remains unclear if the interaction steps referenced by extension points and rejoin points are actually included in or excluded from the scenario.

Consider our “Withdraw Cash” use case in Example 1. Clearly, as we know from our daily banking knowledge, the alternative course in Example 1 rejoins the basic course at label 3 because the next logical scenario step is to offer the possibility to re-enter the PIN. However, the scenario which steps through the alternative course must exclude the basic

course step at the extension point, i.e. label 4. The step at label 4 is clearly excluded from this scenario because if the failure condition of the alternative course holds then there actually is no system action which validates the PIN.

Now consider the following altered basic course of Example 1. This trivial basic course may also be correctly written with the validation statements omitted, i.e. the validation success is implicit:

Example 2: Alteration of Example 1

Use case goal: Withdraw Cash (ATM)

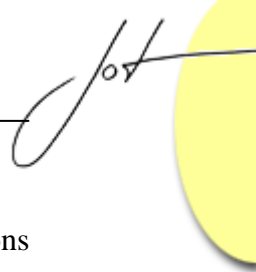
1. The customer inserts the card.
2. The customer enters the PIN.
3. The customer enters the desired amount of money.
4. ...

Now consider the altered alternative course:

- 2a. PIN has been incorrectly entered for the first or the second time:
  - 2a1. The system logs the attempt.
  - 2a2. The system notifies the customer.
 Rejoin at 2.

It is clear that the basic course in Example 2 is rejoined at label 2. However, in this example the validation success is implicit, i.e. it is not explicitly documented. Hence, the extension point references the basic course label 2 because this seems to be the suitable context for checking the failure condition. However, in such a situation, the interaction step of the extension point, i.e. entering the PIN at label 2, clearly cannot be excluded from the scenario, as was the case in Example 1, because entering the PIN is the prerequisite for the system to be able to check if the entered PIN is incorrect. As we see, in contrast to Example 1, in Example 2 the interaction step referenced by the extension point must be included into the scenario.

These examples highlight the following problem: in the use case world there is no common agreement on whether or not the interaction step at an extension point is included into the current scenario; the outcome may depend on the use case specification approach chosen, i.e. explicitly or implicitly specified validation action, both of which are commonly considered good practice. Moreover, it may even depend on the interpretation of the text. This presents further difficulties because individual interpretation varies and is highly dependent on the reader's experience and knowledge of the underlying business semantics. Consequently, without clear use case semantics a given use case specification may be interpreted differently. Business analysts and software developers find this situation unsatisfactory because the absence of clear semantics glaringly fosters misunderstandings and misinterpretations of the functional requirements expressed by use cases.



With respect to a solution, we can say that there can be found four possible combinations of extension point and rejoin points semantics, namely:

- Option (1) -  
extension point **inclusive**, and rejoin point **inclusive**;
- Option (2) -  
extension point **inclusive**, and rejoin point **exclusive**;
- Option (3) -  
extension point **exclusive**, and rejoin point **inclusive**;
- Option (4) -  
extension point **exclusive**, and rejoin point **exclusive**.

However, leading technical text book authors and development process methods do not commit on the issue; in fact, the literature remains totally vague and imprecise, thus leaving the decision and interpretation to the reader [1], [2], [3], [5], [6], [7], [8], [9], [10], [15], [16]. In particular, use case examples in literature do not consistently use one of these options; the options are mixed in [1], [2], [3], [5], [10]. This is mostly the case when two distinct and subsequent actions, i.e. sub-goals, are labelled as a single interaction step but only one of these is to be excluded from the active scenario. Furthermore, the four above-mentioned possibilities are not even highlighted in literature. As a result, there is no methodological guidance for use case practitioners.

Before proposing a solution, however, we wish to highlight further weaknesses arising from the absence of commitment on the true semantics of extension points and rejoin points. One such weakness is the way conditional insertion is documented.

Conditional insertion refers to an optional interaction part that is subject to pure guarded insertion into another interaction course. Conditional insertion is not replacement; rather, it is a conditional addition of use case interaction [1], [2], [5], [10], [15], [17], [18]. But what exactly does this mean? Does this mean that the extension point and the rejoin point are strictly consecutive? Does this mean that the extension point and the rejoin point must be identical? Careful examination of current literature reveals examples of both approaches. Consider the following example:

Example 3:

Consider a use case with the goal “Establish New Customer Bank Account“. Further consider the following fragment of the basic course:

1. The service clerk enters customer’s name and address.
2. The system registers the new customer.
3. The system assigns a bank account number.
4. ...

Now consider the following alternative course:

3a. The service clerk indicates that the customer is a VIP customer:

3a1. The service clerk determines an overdraft facility.

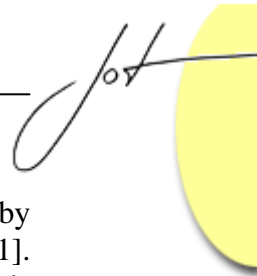
As we know from our knowledge about banking, the alternative course in Example 3 is obviously meant as conditional insertion. However, this is not made explicit because no rejoin point is specified; this could also mean the ending of the use case. Moreover, from our banking knowledge we know that an account number must be assigned first before an overdraft facility can be determined, i.e. the basic course action label 3 must not be excluded from the scenario flowing through the conditional insertion. However, this is not made explicit either. Nevertheless, we realise, because of available banking knowledge, that the active scenario must follow 1,2,3,3a1,4,... When taking a look at industrial requirements specifications and at the current literature, there are use case examples that, indeed, suffer from the problem of not explicitly stating rejoin points thus increasing the reading and understanding effort [1], [2], [5], [10], [16].

With respect to these problems, it is obvious that assumptions can be made based on our individual knowledge and experience of the underlying business domain. However, what happens when little, inconsistent or even no knowledge of the domain is available? This situation can be encountered by everyone who engages in software consulting in industrial projects where personnel changes are frequent. Even when equal knowledge of the true business domain is present, anyone experienced in industrial requirements engineering knows that there are always differing working philosophies among stakeholders competing with varying operating procedures among the prospective end users. Consequently, it is highly desirable to have general methodological guidance for use case analysis, specification and modelling clearly indicating how a baselined use case specification should be interpreted.

As noted previously, it is unclear whether the interaction steps referenced by an extension point or a rejoin point are inclusive or exclusive to a scenario. It is also unclear if the type of alternative course is relevant to answering this question since it must be clearly known where a rejoin point is required. Hence, all possible types and kinds of alternative courses must be considered, before a reasonable and uniform solution can be developed. The following section repeats all possible types of alternative courses as defined in [11], with each one indicating whether or not it needs a rejoin point. Our final solution proposal on extension and rejoin point semantics is presented subsequently.

### 3 TYPES OF ALTERNATIVE COURSES

In literature the term “alternative course“ is explained and treated inconsistently. Based on a rigorous examination and discussion of current literature, and practical observations,



we have clarified the “terms jungle“ that impacts on the term alternative course by suggesting the sub-types use case exception, alternative history, and alternative part [11]. Moreover, we have consequently related these definitions to the notion of use case goals and use case business results in the context of goal-driven requirements engineering. In particular, we have interconnected these types of alternative courses with Cockburn’s use case guarantees [5], which is a refinement of the term use case business results.

Note that the following examples all apply Option (3), as defined in Section 2, when needing to handle extension points and rejoin points.

1. Use Case Exception

A use case exception branches the base interaction course and never rejoins; furthermore, it never establishes the use case’s business results. However, minimum guarantees are met and therefore the stakeholders interests are protected. This is indicated by the “earthing” in Figure 3. The following example illustrates a use case exception for the use case in Example 1:



Figure 3. Illustration of a use case exception

Example 4: Use Case Exception

Consider the following additional alternative course, representing a use case exception, for the use case in Example 1:

- 3b. PIN has been entered incorrectly the third time:
  - 3b1. The system logs the attempt.
  - 3b2. The system withholds the card.
  - 3b3. The system notifies the customer.
  - Use case aborts.

2. Alternative History

An alternative history branches the base interaction sequence never to rejoin, i.e. an alternative history also represents a fully parallel interaction flow. An alternative history differs from a use case exception in that at the end of an alternative history the use case goal is achieved and its business results are delivered. This is indicated by the line going to the finishing line in Figure 4. The following example illustrates an alternative history:

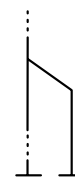


Figure 4. Illustration of an alternative history

Example 5: Alternative History

Consider a use case with the goal “Register New Employee“. Further consider the following basic course:

- 1. The system displays the vacancies and job category.
- 2. The personnel manager chooses the considered vacancy.

3. The personnel manager enters new employee's name, ...
4. The personnel manager assigns the company department.
5. The system assigns a unique employee number.
6. The system shows the scale of wages.
7. The personnel manager determines the hourly rate.

Now consider the following alternative history:

- 6a. The selected vacancy requires that the applicant be paid outside the pay scale and have flexible working hours:
    - 6a1. The personnel manager determines the new employee's monthly salary.
    - 6a2. The personnel manager determines fringe benefits.
- Use case completes.

### 3. Alternative Part

An alternative part is an alternative course that always rejoins the branched base sequence; this encompasses conditional insertion (see Figure 5 a), interaction cycles (see Figure 5 b) and alternative interaction fragments (see Figure 5 c). The following examples illustrate alternative parts:

#### Example 6: Conditional Insertion

Consider a use case with the goal "Register New Student". Further consider the following basic course:

1. Clerk enters new student's name, address, ...
2. The system assigns a student ID.
3. Clerk assigns the university department.
4. System prints out a confirmation of registration.
5. ...

Now consider the following alternative course:

- 3a. Student is a foreign student with pre-registered scholarship:
    - 3a1. The system shows the capacity of free campus apartments.
    - 3a2. The clerk assigns an apartment to student ID.
- Rejoin at 3.

#### Example 7: Alternative interaction fragment

Consider the following basic course of a use case with the goal "Change Customer's Contract Agreements": a customer can be identified either by entering the contract number or his name and birthday.





1. Service clerk enters the the customer's name.
2. The system shows all choices.
3. Service clerk chooses the considered customer.
4. The system shows the customer's contract profile.
5. ...

Now consider the following alternative course:

- 1a. Service clerk indicates that he wants to search by contract number:
  - 1a1. Service clerk enters the contract number.
 Rejoin at 4.

An example of an interaction cycle can be found in Example 1.

Now that there are three types of alternative courses defined that reveal when a rejoin point is actually needed, the following provides a solution for handling extension points and rejoin points uniformly for any type of alternative course.

## 4 SOLUTION PROPOSAL ON THE SEMANTICS OF EXTENSION POINTS AND REJOIN POINTS

### Promoting Option (3)

Our solution promotes Option (3) in Section 2 thereby suggesting the following uniform approach when dealing with alternative courses, extension points and rejoin points:

- Extension Point
 

The interaction steps within the branched interaction course that are referenced by the extension point are always excluded from the scenario; the first interaction step of the alternative course substitutes the one referenced by the extension point.
- Rejoin Point
 

The interaction steps within the rejoined interaction course that are referenced by the rejoin point are always included into the scenario.

Consequently, this solution implies that for conditional insertions the extension point and the rejoin point always reference the same label within the base interaction course. The reason is obvious: an insertion is not an alternative but an addition and, thus, nothing can be excluded from the scenario. The approach presented here guarantees that the excluded interaction step referenced by the extension point is re-included into the scenario after all. Note, however, that in our scenario semantics for conditional insertion the active scenario rejoining at the rejoin point only performs its function, i.e. it never checks for the same conditional behaviour again of other alternative courses branching off from this location; this implies that all guards at the same extension point must be formally disjoint.

For simplification in practice, we suggest the following default: for conditional insertion no textual rejoin point is specified in the use case narratives. If no rejoin point is given, this indicates conditional insertion, i.e. the rejoin point is implicitly identical with the extension point. Use case exceptions always state “Use case aborts”. Alternative histories always state “Use case completes”. In order to follow our approach in practice the extension points and rejoin points are correspondingly set as appropriate so that the intended scenario logic is maintained. This solution is consistent with any use case development approach that can be found in literature.

Correspondingly, the alternative courses in Example 2 and Example 3 are rewritten as shown below:

Adjustment of Example 2:

3a. PIN has been incorrectly entered for the first or the second time:

3a1. The system logs the attempt  
3a2. The system notifies the customer  
Rejoin at 2.

Possible scenario: 1, 2, 3a1, 3a2, 2, 3, 4, ...

Adjustment of Example 3:

4a. The service clerk indicates that the customer is a VIP customer:

4a1. The service clerk determines an overdraft facility.

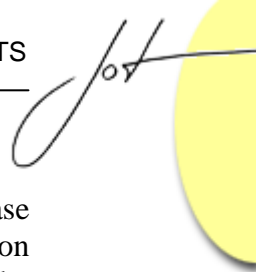
Possible scenario: 1, 2, 3, 4a1, 4, ...

The alternative courses in Example 5, Example 6 and

Example 7 are already compliant with our solution. Possible scenarios for them are:

- for Example 5: 1, 2, 3, 4, 5, 6a1, 6a2.
- for Example 6: 1, 2, 3a1, 3a2, 3, 4, 5, ...
- for
- Example 7: 1, 2, 3, 4a1, 4a2, 6, 7.

The careful reader will have noticed that in all examples the rejoin point statements, the unsuccessful termination statements and the successful completion statements of the use case are not labelled as an interaction step. This seems perfectly reasonable since “direction statements” are not ranking full use case business interaction steps. However, in some situations we may want to have an alternative course branch off from another



interaction course directly after the last interaction step but before the use case termination or the rejoining, respectively. We can do so by also labelling the termination and rejoin statements just like true use case interaction steps. Such labels can then be used for branching as presented. In this context, in symmetry with the way an alternative history (see Section 3) is specified, we may also append to the basic course the explicit system responsibility “Use case completes”.

### Why are Option (2), Option (4), and Option (1) inappropriate?

Option (2) and Option (4) suggest exclusive semantics for rejoin points. In our opinion exclusion does not intuitively accompany the intention of “rejoin at...” a certain interaction step. Hence, these options would hardly be accepted in practice.

Option (1) suggests inclusion semantics for both extension points and rejoin point. This approach is intuitive, and it is also independent of implicit or explicit validation statements (see Example 1 and Example 2). Moreover, Option (1) also supports the above mentioned device of branching off from any interaction course before the rejoining or use case termination, respectively. However, Option (1) is not consistent with the way alternative courses are specified: consider a situation in which we want to add an alternative course to the basic course branching off before the first basic course interaction step but after the use case has started<sup>1</sup>.

<sup>1</sup> In particular, having alternative histories branching off from the first step is common in practice. Such an alternative history is not an individual use case if the business results of this alternative history and the basic course are equivalent [5]; the alternative history then is an integral part of the use case’s behaviour because they deliver the same goal with respect to the goal’s business results (see the ScenarioPlusFragments and ExhaustiveAlternatives use case patterns in [1]).

Also consider the addition of an alternative course to an existing alternative course before the first alternative course interaction step but after the alternative course has been entered.

In order to illustrate this, consider Example 8 that reflects a simplified version of a real industrial use case description of the German Telekom general billing services software specification. When a customer enters a local Telekom service centre in order to request an upgrade from ISDN to DSL, for instance, the service clerk must find the customer’s profile. This is usually done by entering the billing number if the customer brings his paper contract or any billing invoice with him. If not, the customer’s profile can also be identified by his telephone number.

Example 8: Use case goal “Change customer’s Telekom service contract”

Basic course:

1. Service clerk enters the customer’s billing number.
2. The system shows the customer’s service contract and profile.

3. ...

Alternative course attempt: the extension point remains unclear using  
Option (1):

?a. Service clerk indicates that he wants to search by telephone  
number:

.1 Operations clerk enters the customer's telephone number.  
Rejoin at 2.

Example 9: Solution for Example 8

Basic course:

0. The use case starts.
1. Service clerk enters the customer's billing number.
2. The system shows the customer's profile with the customer's  
telecommunication services.
3. ...

Alternative course solution with a determined extension point:

0a. Service clerk indicates that he wants to search by telephone  
number:

0a1. Service clerk enters the customer's telephone number.  
Rejoin at 2.

Using Option (1), we would have to solve the problem in Example 8 by adding the auxiliary statement "The use case starts" as shown in Example 9, because Option (1) demands inclusion semantics for the interaction steps referenced by the extension point. This looks very clumsy and, further, differs significantly from the way alternative courses are specified: an alternative course must always either give a rejoin statement or a use case termination statement which, in any case, is located at the very end of the alternative course's text block (see Section 3). Now, since a basic course also is an interaction course, just like an alternative course, it is difficult to see why a basic course should carry a "starting" statement at the beginning while an alternative course carries a rejoin or termination statement at the end. Apart from this asymmetry, the textual complexity is unnecessarily increased for an alternative course having both starting and rejoin or termination statements, if we consider the branching from the first step of an alternative course. Introducing additional complexity truly appears counter-intuitive with respect to simplicity and methodological guidance.



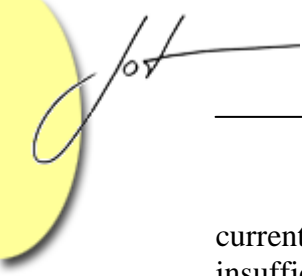
## 5 CONCLUSION

The true meaning of combining use case interaction descriptions with extension points and rejoin points in industrial functional requirements specifications dealing with complex domains has been shown to be currently unclear. The first clarifying step was to build on the alternative courses types as have been consolidated in the textual use case world in [11]. We have then developed an approach to extension points and rejoin points in order to finalise the definition of alternative courses. We call our solution approach the “*Exclusive-Inclusive Scenario Branching*” rule: the interaction steps within the branched interaction course that are referenced by the extension points are always excluded from a use case scenario, while the interaction steps within the rejoined interaction course that are referenced by rejoin points are always included into a use case scenario.

Following this rule the branching and rejoining of any type of alternative course can be treated and understood uniformly; no extra convention, UML stereotype or tagged value is needed. This facilitates a practitioner because he or she no longer needs to reflect on what type of alternative course he or she is documenting when dealing with extension points and rejoin points; the practitioner needs to focus on the goal and the business results only when documenting use case interaction. Furthermore, the understanding of extension points and rejoin points is now fully independent of the comprehension of the underlying business domain; it is also independent of business knowledge available in the team. Moreover, with our solution basic courses and alternative course specifications are treated symmetrically while having minimum textual complexity and still supporting all situations mentioned in this paper.

While most of the recently published use case patterns focus on the quality of the use case content and the writing *style*, the presented approach to extension points and rejoin points makes more precise the given semantics of the use case *syntax*. Hence, our solution is not in conflict with, but even adds to, the goals of the use case patterns *MultipleForms* and *PreciseAndReadable* in [1]. Our solution helps to reduce misunderstandings among stakeholders, requirements analysts, and software developers thus contributing to the correctness of use cases as functional requirements vehicles and to the stakeholder’s needs for requirements precision and content [1]. In particular, our approach contributes to use case rigor and precision which is essentially important when the implementation activity is outsourced and the stakeholder’s business “core competency” is no longer available [1].

UML neither defines use case documentation items, nor does it offer a tailorable use case template [4], [13], [14]. UML considers only the graphical part of what makes up a use case model, i.e. the use case diagram. However, in practice a use case model consists of written specifications that are supported by a use case diagram, not vice versa (“A bubble does not tell us the story!”). In fact, the “graphical world” and the “textual world”



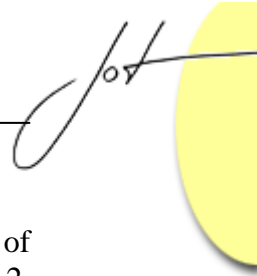
currently exist in isolation from each other. In this respect, UML's Extend-relationship is insufficiently explained as evidenced by numerous debates in the last decade; it is not even related to the textual world. However, in [12] it is suggested how to merge these two worlds by thoroughly refining and explicitly connecting UML's Extend-relationship with the notion of alternative courses following the original spirit of Jacobson et al. in OOSE [9], i.e. Extend can be applied in order to extract all types of alternative courses into an extension use case and to attach this extension use case to the base use case. Consequently, our definitions of the alternative course types [11] together with our suggested semantics of extension points and rejoin points presented here should also be applicable to the Extend-relationship which would assist practitioners in aligning both worlds and practical use case model refactoring.

## ACKNOWLEDGEMENTS

We wish to express our gratitude to Matthias Schork and Shane Sendall who have helped to improve this work. Thanks to Steve Adolph and Alistair Cockburn for the discussions on our alternative course types and our “*Exclusive-Inclusive Scenario Branching*” rule. Special thanks to Roderick Coleman.

## REFERENCES

- [1] Adolph S., Bramble P., Cockburn A., Pols A., *Patterns for Effective Use Cases*, Addison-Wesley, 2003
- [2] Armour F., Miller G., *Advanced Use Case Modeling*, Addison-Wesley, 2001
- [3] Bittner K., Spence I., *Use Case Modelling*, Addison-Wesley, 2003
- [4] Booch G., Jacobson I., Rumbaugh J., *The Unified Modeling Language User Guide*, Addison-Wesley, 1999
- [5] Cockburn A., *Writing Effective Use Cases*, Addison-Wesley, 2001
- [6] IBM Global Services, IBM Global Services Method Release 3.1, IBM Corporation, 1998, 1999, 2000, 2001, 2002
- [7] Jacobson I., “The Road to the Unified Software Development Process”, *Cambridge University Press, SIGS Reference Series, 2000* (compilation of formerly published articles of I. Jacobson, revised and updated by S. Bylund)
- [8] Jacobson I., Booch G., Rumbaugh J., *The Unified Software Development Process*, Addison-Wesley, 2000
- [9] Jacobson I. Christerson M., Jonsson P., Övergaard, G., *Object Oriented Software Engineering – A Use Case Driven Approach*, Addison-Wesley, 1992
- [10] Kulak D., Guiney E., *Use Cases – Requirements In Context*, Addison-Wesley, ACM Press, 2000



- [11] Metz P., O'Brien J., Weber W. "Specifying Use Case Interaction: Types of Alternative Courses", in *Journal of Object Technology (JOT)*, vol. 2, no. 2, March-April 2003, pp. 111-131. [http://www.jot.fm/issues/issue\\_2003\\_03/article1](http://www.jot.fm/issues/issue_2003_03/article1)
- [12] Metz P. "Merging the Use Case Textual and Graphical Use Case Worlds", *Ph.D. thesis, Dept. of Mathematics & Computing, Cork Institute of Technology*, 2003
- [13] OMG Unified Modeling Language Specification, Version 1.4, September, 2001
- [14] Rumbaugh J., Jacobson I., Booch G., *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999
- [15] Rosenberg D., Scott K., *Use Case Driven Object Modeling With UML: A Practical Approach*, Addison-Wesley, 1999
- [16] Schneider G., Winters J.P., *Applying Use Cases: A Practical Guide*, Addison-Wesley, 1999
- [17] Simons A. "Use Cases Considered Harmful", *Proceedings of TOOLS-29 Europe*, eds. R Mitchell, A C Wills, J Bosch and B Meyer (Los Alamitos, CA: IEEE Computer Society, 1999), p. 194-203, available on <http://www.dcs.shef.ac.uk/~ajhs/abstracts.html#harmful>
- [18] Simons A., Graham I. "30 Things That Go Wrong In Object Modelling With UML 1.3", *Behavioral Specifications of Businesses and Systems*, eds. H. Kilov, B. Rumpe, I. Simmonds, Kluwer Academic Publishers, 1999, p. 237-257, available on <http://www.dcs.shef.ac.uk/~ajhs/abstracts.html#uml30thg>

## About the authors



**Pierre Metz** is a postgraduate PhD student at the Dept. of Mathematics and Computing, Cork Institute of Technology, Ireland. His interested in the area of software process assessment and improvement following CMMI and ISO 15504/SPICE, requirements engineering, quality assurance, use case analysis and modelling, object-oriented foundations, and UML. Pierre can be reached at [pmetz@cit.ie](mailto:pmetz@cit.ie)

**John O'Brien** joined the lecturing staff of the Maths and Computing Department, Cork Institute of Technology, CIT in 1987. Here he has lectured on, and developed courses in, software engineering at undergraduate and post-graduate levels. His current research areas include quality software engineering and project management. John is a current member of the academic council at CIT. John can be contacted at [jobrien@cit.ie](mailto:jobrien@cit.ie)



**Wolfgang Weber** is a full professor at the Dept. of Computer Science at Darmstadt University of Applied Sciences, Germany. He works and teaches in the field of project management, software engineering, UML, object-oriented methods and programming. Wolfgang can be reached at [w.weber@fbi.fh-darmstadt.de](mailto:w.weber@fbi.fh-darmstadt.de)