IDENTIFYING CROSSCUTTING CONCERNS IN REQUIREMENT SPECIFICATIONS - A CASE STUDY

by

Gang Li

A thesis submitted to the School of Computing

In conformity with the requirements for

the degree of Master of Science

Queen's University

Kingston, Ontario, Canada

(September, 2009)

ABSTRACT

Aspect-Oriented Requirement Engineering (AORE) is an emerging software engineering paradigm with increasing attention from academic as well as industrial communities. AORE aims at the systematic identification, modularization, composition and analysis of crosscutting concerns that manifest in requirements. It is believed that systematically managing crosscutting concerns early on at the requirement engineering stage can provide valuable insight at the architecture design and implementation stages and can help identify and thus manage crosscutting concerns at these stages [6]. Moreover, identifying crosscutting concerns in requirements can help to reveal the scope of each concern in a software system, to detect potential conflicts between concerns and to facilitate trade-off negotiation early on. Hundreds of papers regarding AORE have been published in AORE communities. However, few of them address crosscutting concerns in real world requirements. Whether the proposed AORE approaches are productive when applied to real world requirements is unknown. In this thesis, we conduct an AORE case study consisting of an experiment using a real world software requirement specification in order to:

- examine how crosscutting concerns present in real world requirement documents,
- explore the difference between crosscutting concerns in requirements and crosscutting concerns in code,
- reason whether identifying and thus managing crosscutting concerns from real world requirements is a productive practice.

ACKNOWLEDGEMENTS

I am very grateful to Dr. Terry Shepard for being a patient supervisor and for supporting this work with endless guidance, assistance, and comments.

I would like to thank Dr. Steve Easterbrook, Dr. Arno Jacobsen, Dr. Eric Yu, Flannan Lo and Nan Niu at the University of Toronto for their generous sharing of their thoughts and for their valuable suggestions. I would also like to thank all people who gave me help and support in the process of this work. Without you, this work would not happen.

I dedicate this work to my parents and my wife for their understanding, encouragement and patience. I also dedicate this work to my new born child who brought me great pleasure when I struggled with this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

API – Application Programming Interface. An application programming interface is an interface that defines the ways by which an application program may request services from libraries and/or operating systems.

AO/AOSD – Aspect Oriented Software Development. Aspect-oriented software development is an emerging software development technology that seeks new modularizations of software systems so that multiple concerns can be expressed separately and automatically unified into working systems.

COI – Condition of Interest. A set of conditions to determine which information is needed when and by whom.

EPR – Endpoint Reference. An Endpoint Reference (EPR) is an XML structure encapsulating information useful for addressing a message to a Web service.

VIRT – Valuable Information at the Right Time. A term coined by Rick Hayes-Roth to capture the importance of having the most up-to-date information available to the party by whom this information is requested [13].

Chapter 1

Introduction

In this chapter, the motivation of this thesis will be given. Then, the hypothesis and the goals of the thesis will be introduced and discussed, along with the initial presentation of a case study based on an experiment undertaken to address the hypothesis and the goals. Finally, the organization and structure of the rest of the thesis will be outlined.

1.1 Motivation

Aspect-Oriented Software Development (AOSD) [3] is an emerging software engineering paradigm. It aims at the systematic modularization and management of crosscutting concerns throughout the entire software development lifecycle. AOSD complements existing software development paradigms such as Object-Oriented Software Development and Component-Based Software Development with explicit means to capture and manage crosscutting concerns, which are often seen as being not properly handled in existing development practices. AOSD is believed to promote better achievement of the time-honored principle of separation of concerns and thus to improve the quality of software systems and the efficiency of software development. As a result, AOSD is gaining increasing attention from academic organizations as well as from industrial companies.

AOSD consists of a series of development disciplines, covering a wide span of development activities in the software development lifecycle. Among them is Aspect-Oriented Requirement Engineering (AORE). AORE focuses on the systematic identification, modularization, composition and analysis of crosscutting concerns which are evident at the requirement engineering stage. Hundreds of papers regarding AORE have been published [12]. Researchers in AORE communities believe that identifying and capturing crosscutting concerns early on, at the requirement engineering stage, will benefit downstream development activities such as architecture design and implementation [6]. The identified crosscutting concerns may offer valuable insight at the architecture design and implementation stages. They often eventually correspond to crosscutting concerns in architecture, and then in code. As a result, pedigrees of crosscutting concerns throughout the entire software development lifecycle will be established, improving the traceability of a wide range of concerns in a software system and facilitating the system's evolvability. Moreover, identifying crosscutting concerns at the requirement engineering stage may help to reveal the scope of each concern, detect potential conflicts between concerns and support trade-off negotiation and earlier decision making.

However, there is limited evidence that early identification of requirement level crosscutting concerns is a productive software engineering practice. First of all, although a great amount of literature on AORE has been published worldwide, none of these

papers, to the best of our knowledge, addresses this question. The cost of early identification and management of crosscutting concerns in requirements may outweigh the benefits. Intuitively, an argument can be made that this is especially the case when requirements are not fully developed – when there is a large amount of uncertainty and volatility.

Secondly, most proposed AORE approaches in the literature are supported with small scale, simplified, and sometimes artificial examples. The Portuguese Highways Toll Collection System [35] is a commonly referenced example in AORE literature. Its requirement document contains only three paragraphs with no more than 200 words in total. This is much simpler than an average requirement document in the real world. Consequently, there is no convincing evidence that proposed AORE approaches are feasible and productive when applied to larger scale real world projects. A careful analysis of a real world software requirement document could provide some insight into the value or lack of value of the proposed AORE approaches.

Moreover, most proposed AORE approaches in the literature aim at identifying and thus capturing crosscutting concerns from well structured, formal (or at least semi-formal) requirement documents that are organized as use cases [17], goals [44], viewpoints [35][36], etc. Only a few AORE approaches deal with crosscutting concerns in less structured requirement documents such as informal software requirement specifications

[37]. There is not sufficient evidence to show that identifying and capturing crosscutting concerns is feasible for less structured and informal requirement documents.

In addition, we have also observed in our background study (described in detail in Chapter 2) that crosscutting concerns in requirements manifest themselves in quite different forms from crosscutting concerns in code – the nature of crosscutting is different in requirements. A concern in code is considered crosscutting if it is scattered (code pieces implementing this concern appear in several modules that are intended to be separate parts of a decomposition) and perhaps also tangled (code pieces implementing different concerns are mixed within the same code module). Crosscutting in requirements may not take place in such an explicit manner [37]. Although rarely discussed in the AORE literature, the distinct characteristics of crosscutting concerns in requirements compared to crosscutting concerns in code deserve careful study. Simply borrowing and applying code level AO concepts and solutions to requirements without taking into account the distinct characteristics of crosscutting concerns in requirements may be fruitless.

Based on the above observations, we boldly hypothesize that ***identification of crosscutting concerns from requirement documents that are structured without taking aspects into account requires more work than it is worth***.

1.2 Goals

The above hypothesis is too broad to be settled in one thesis. In this case study, we make some initial investigation of the following aspects of this hypothesis:

1. Of the papers on AORE that we are aware, none of them characterizes how crosscutting concerns manifest in requirements, especially requirement documents that are constructed without awareness of crosscutting concerns. What are the characteristics of crosscutting concerns in requirements, and to what degree and how, do crosscutting concerns in requirements differ from crosscutting concerns in code?

2. Can techniques used to identify crosscutting concerns in code be used to identify crosscutting concerns in requirements? And is it possible to predict crosscutting concerns in design and in code from crosscutting concerns in requirements?

3. When would be good time to identify crosscutting concerns in requirements? Should we start from the beginning as requirements are elicited and compiled or should we start after requirements have been developed into a relatively stable state, or should we start at other stages in between?

In this thesis, an AORE case study of a real world software requirement document – the INFOD base specification [10] – will be conducted to explore answers to the above

questions and to provide evidence related to the above hypothesis. The INFOD base specification exemplifies real world software requirement documents in that it addresses a variety of concerns from different perspectives; it describes the software system under construction at different levels of abstraction; it is less structured and less complete compared with the examples used in previous AORE studies and it is subject to ongoing completion and evolution. An AORE guideline proposed by E. Baniassad et al. [6] will be adopted to experiment with the identification and capture of crosscutting concerns in the INFOD base specification. The case study starts with studying the dominant decompositions of the INFOD base specification [10]. Crosscutting concerns are identified by searching for aspectual terms, impact descriptions and scattered representations of concerns. Following this is systematic capturing and reasoning about these crosscutting concerns. Finally, a set of characteristics of crosscutting concerns in requirements is proposed on completion of this process.

1.3 Organization

The rest of this thesis is organized as follows. A comprehensive background of AOP, AORE and AOSD is given in Chapter 2, setting up the context for this thesis. Chapter 3 presents and discusses relevant research work in the AORE community. An overview of software requirement specifications is given in Chapter 4. Following is an introduction to the INFOD system and the INFOD base specification. The experiment setup of applying existing knowledge of AORE to the INFOD base specification is then described and

discussed. In Chapter 5, a series of analyses of, and discussions about data collected in the experiment are given. Finally in Chapter 6, the work of this thesis is concluded and possible directions of future work are pointed out.

1.4 Summary

In this chapter, we provided the motivation of the thesis and of our case study and experiment – investigating the presence, structure and characteristics of crosscutting concerns in requirements in a real world requirement document. Then, we presented our observations in a background study of AORE, and our hypothesis regarding the possible poor productivity of early identification of crosscutting concerns in requirements. We also posed questions which we plan to address in our case study and experiment. Finally, we presented the overall structure of this thesis.

Chapter 2

Background

In this chapter, basic concepts in the aspect-orientation world will be explained briefly. Then, an introduction to Aspect-Oriented Programming will be provided. Finally, the background of Aspect-Oriented Software Development, in particular Aspect-Oriented Requirement Engineering will be presented.

2.1 Concerns, Separation of Concerns and Crosscutting Concerns

Concern is generally defined as "a matter for consideration" [27]. In software engineering, it is best interpreted as something a stakeholder has identified to be dealt with in the software system under construction.

Separation of Concerns is a long-established principle in software engineering. It leads to the decomposition of a system into successively smaller and more manageable modules. Each module encapsulates a distinct concern of the system. Ideally, a module can be constructed with little knowledge of other modules, and changing a module does not require changing other modules [32]. Keeping concerns separated from each other in modules minimizes the overlaps between them and isolates changes to one concern from affecting other concerns. It has long being recognized that clean separation of concerns

will reduce the complexity of a software system, improve its comprehensibility, promote reusability and facilitate evolution, adaptation and customization of the system [34].

A software system consists of various kinds of concerns that are relevant to different stakeholders, and at different development stages of the software lifecycle [34]. There are many ways of separating and modularizing different kinds of concerns in a software system. For example, concerns of a software system at the requirement stage can be modularized as features or use cases; concerns at the implementation stage can be modularized as classes or procedures. Each way of modularization gives rise to a particular decomposition structure of a system. Traditional software development paradigms can not apply multiple decompositions simultaneously – only one dominant decomposition structure can be chosen at a time. No matter how well a system is decomposed, not all concerns can be well modularized. Only a portion of concerns is separated properly, always at the cost of other concerns. This is known as the *Tyranny of Dominant Decomposition* [34]. The concerns that do not fit well in the dominant decomposition structure are spread over the system and tangled with other concerns. These concerns are collectively referred to as *Crosscutting Concerns*.

The presence of crosscutting concerns is considered to impair the modularity, changeability and evolvability of a system. For example, consider developing a data structure library which includes commonly used data structure types such as vector, stack, and map. Algorithms pertinent to a specific data structure type can be well

encapsulated in the corresponding code module, e.g. class. However, concerns such as concurrent accessing and memory management, e.g. allocating and reclaiming memory can not be modularized in separate classes. Their implementations are scattered in all data structure types and are tangled with each other within each data structure type. For example, allocating memory in these data structure types has to be protected by locks to avoid race condition. Also, the code to implement these concerns is mingled with the code of each data structure type, making the code of these data structure types harder to understand and maintain. Moreover, as this library evolves, changes to memory management, for example, are very likely to propagate to all data structure types.

2.2 Aspect-Oriented Programming

Aspect-Oriented Programming (AOP) was first introduced by G. Kiczales et al. [21]. Their work describes decomposition mechanisms complementary to traditional development paradigms with the goal that both non-crosscutting concerns and crosscutting concerns can be well modularized. The basic idea of AOP is to separate core concerns from crosscutting concerns, encapsulate the otherwise scattered and tangled crosscutting concerns in separated modularization units called aspects and weave or integrate code modules of core concerns and relevant aspects to form a holistic software system [24].

AspectJ [4] is one of the most influential implementations and the widely used de-facto standard of AOP. AspectJ uses a join-point model to achieve the separation of crosscutting concerns. The basic elements of the join-point model are Join Points, Pointcuts and Advices. A join point is an identifiable point in the execution path of a program [24]. It can be a call to a method, an access to a member variable of an object or an arbitrary point in a method body. A pointcut is a program construct that selects join points and collects context at those join points [24]. For example, a pointcut can select a join point that is a call to a method. This pointcut also captures the method's runtime context such as actual parameters to this method and the current call stacks of this method. An advice is a piece of code to be executed at the join points that have been selected by a pointcut [24]. Advice can be specified to execute before, after or around a join point. Aspects, like classes in OOP, are encapsulation units in AOP. An aspect consists of a set of cohesive pointcut declarations and advices to implement a crosscutting concern in a modularized manner. In addition, an aspect can have its own data members and methods. A sample aspect in AspectJ syntax is shown in Figure 2.1.

.

```
class Vector {                                                    ❶
  Item data[];
  Item getItem(int idx) {
      return data[idx];
  }
  void addItem(Item item) {
      data[data.length + 1] = item;
  }
  void removeItem(int idx) {
      for(int I = idx; i < data.length; i++)
          data[i] = data[i + 1];
      data.length -= 1;
  }
}

aspect ConcurrentAccessControlAspect {                            ❷
  Lock lock;
  pointcut access()
      : call(Vector.*Item(..));                                   ❸
  before() : access() {                                           ❹
      lock.lock();
  }
  after() : access() {                                            ❺
      lock.unlock();
  }
}
```

Figure 2.1 Sample AspectJ Aspect

In the above example, ❶ is a vector class that implements a vector data structure with three basic operations: getItem(), addItem() and removeItem(). This class is about to be

crosscut. ❷ is the definition of an aspect that is used to make atomic accesses to the data inside Vector sequential by using a Lock object. No other accesses can be made until the current access finishes. ❸ is a pointcut declaration that selects all data accessing calls to Vector as target join points. ❹ and ❺ are the definition of advices. ❹ defines that the access lock is locked before a data access call and ❺ defines that the access lock is freed after a data access call.

Most AOP implementations modularize non-crosscutting concerns as classes or similar program constructs and modularize crosscutting concerns as aspects. In addition to AspectJ, AOP implementations include Aspect C++ [2], JBoss AOP [19], and others. AOP is intended to encapsulate the otherwise scattered and tangled implementation of crosscutting concerns, and thus to improve the comprehensibility and maintainability of the entire software system.

2.3 Early Aspects and Aspect-Oriented Software Development

In name as well as in practice, AOP is a programming paradigm. It is limited to dealing with crosscutting concerns at the implementation stage of software development. However, dealing with crosscutting concerns at the implementation stage is often insufficient, since some important crosscutting concerns are well presented prior to the implementation stage. For example, performance and security are widely recognized crosscutting concerns that need to be handled with great care at the early design stages

such as the system architecture design. In fact, activities at early development stages often set important design decisions, and thus have a large influence on the whole system [35].

Modern software systems run in highly volatile environments where business rules often change rapidly. The systems must be easy to adapt and evolve. If not handled properly, crosscutting concerns may reduce adaptability and evolvability of software systems. Instead of being limited to the implementation stage, researchers in the AO community believe that systematic separation of crosscutting concerns carried out from the early stages of software development, such as the requirement engineering stages and the architecture design stages, and on through the entire software lifecycle will produce more aligned artifacts in each development stage, and thus improving the traceability through out the entire software lifecycle [1]. Maintainability and evolvability of the software system could then be improved as well.

A number of papers have been published [5][7][17][28][35][36][40], attempting to generalize the concepts that arise from the realm of AOP, and to apply them to other development activities. Several Aspect-Oriented approaches have been proposed, aiming at providing systematic treatment of crosscutting concerns throughout the entire software development lifecycle, including requirement engineering and architecture design as well as implementation. Aspect oriented approaches that address crosscutting concerns prior to the implementation stage are collectively referred to as *Early Aspects* approaches.

"Early" signifies the occurrence of crosscutting concerns before the coding stage in any development iteration [6].

Crosscutting concerns appearing at the requirement engineering stage are referred to as requirement level crosscutting concerns or requirement aspects. E. Baniassad et al. define requirement aspects as:

"A requirements aspect, then, is a concern that cuts across other requirement level concerns or artifacts of the author's chosen organization. It is broadly scoped in that it's found in and has an (implicit or explicit) impact on more than one requirement artifact." [6]

From this definition we can see that crosscutting concerns in requirements are requirement statements that appear in multiple requirement artifacts in a requirement document but refer to same concerns, and have influence, either implicit or explicit, on more than one requirement artifacts.

Concerns that are derived from non-functional concerns – qualities of the system, such as data consistency, security, and portability – often become requirement level crosscutting concerns that are scattered and tangled with other requirements. They are broadly scoped in that they are found in and have implicit or explicit impacts on more than one requirement artifact. Existing requirement engineering approaches lack means to deal

with crosscutting concerns. For example, use cases [16] are good at capturing functional concerns of a system – what users can do with the system. However, most use cases practitioners tend to leave non-functional concerns out of use case modeling because non-functional concerns are hard to model as use cases. For this reason, Jacobson revised his use case approach in [17], using application use cases to capture functional concerns and infrastructure use cases to capture non-functional concerns. Application use cases are traditional use cases that describe how actors interact with the system to perform the desired functionality. Infrastructure use cases are first introduced in [17] to describe what the system does to add qualities such as usability and reliability to steps of application use cases. An infrastructure use case is a kind of template use case describing a series of action patterns that are bound to steps of application use cases. The detail of this use case based AOSD approach is not provided in this thesis.

In addition to non-functional concerns, functional concerns can become crosscutting concerns as well [6]. For example, consider a system which consists of multiple service nodes and each service node only provides service to callers that belong to certain groups. In this system, the functional concern – checking which group a caller belongs to – is a crosscutting concern which cuts across all service nodes. Similarly, keeping audit history of all users transactions in a banking system [6] is also a functional crosscutting concern. Existing AORE approaches also lack means to deal with functional crosscutting concerns.

Aspect-Oriented Requirement Engineering (AORE) approaches have been proposed to identify, capture, and manage crosscutting concerns in requirements. For example, an AORE approach has been proposed by A. Rashid et al. in [35][36]. In this approach non-crosscutting concerns or base concerns are referred to as requirements while crosscutting concerns are referred to as concerns – an adapted notion from PREView [41] concerns, and a concern can constrain more than one requirement. This approach identifies both concerns and requirements at the beginning. Requirements are modeled with traditional requirement engineering mechanisms. A constraint matrix is then established to visualize how a concern constrains requirements. If a concern constrains more than one requirement, it is seen as a candidate aspect. After identification of aspects, composition rules are derived that specify which aspects are applied to which requirements, facilitating conflict detection and resolution.

Aspect-Oriented methodology is also proposed for software architecture. Software architecture generally refers to the coarse grained structure or structures of a software system, which specify software components, the externally visible properties of these components, and the relationships among them [23]. It is generally accepted that architectures of modern software systems are too complex to be described in a simple one-dimensional fashion and must be described as a set of views [9][22]. Each view shows particular types of elements and the corresponding relationships between them. For example, logical views address the functional perspectives of a system – what a system should do and what major components a system contains; process views address

the concurrent perspectives of a system at runtime – tasks, threads, or processes as well as their interactions; deployment views address how a software system relates to its deployment and execution environment [22][23]. Crosscutting concerns evident at the architecture design stage may crosscut architecture views, part of views, or other architectural elements [6]. They are referred to as architecture level crosscutting concerns or architecture aspects. Architecture level crosscutting concerns can be scattered in architecture design documents and tangled with other architecture concerns, resulting in poor quality architecture designs. A number of approaches, for example [20][33][42], have been proposed to deal with architecture level crosscutting concerns. These approaches are collectively referred to as Aspect-Oriented Architecture Design (AOAD) approaches.

Early Aspects approaches and AOP approaches altogether form an emerging software development paradigm – *Aspect-Oriented Software Development* (AOSD). The AOSD techniques aim at providing means for the systematic identification, modularization, and composition of crosscutting concerns from the early stages of software development and all the way through software development processes. A great number of papers regarding AOSD have been published, covering a wide span of development activities including requirement analysis, domain modeling, architecture design, and programming. Researchers in the AOSD community believe that by handling crosscutting concerns in a systematic fashion throughout an entire software development process, traceability and comprehensibility of the artifacts of given software system development, as well as the

adaptability, configurability, maintainability and evolvability of the developed or modified system will be improved greatly.

2.4 Summary

In this chapter, we briefly explained some basic concepts of Aspect-Orientation (AO) such as separation of concerns and dominant decomposition. After that we recapped Aspect-Oriented Programming, which was invented at the end of last century and is seen as a promising implementation paradigm, promoting improved modularity. Finally, we introduced Aspect-Oriented Requirement Engineering and Aspect-Oriented Architecture Design.

Chapter 3

Related Work

In this chapter, research work that is significant to our experiment will be discussed. This research work covers identification and management of crosscutting concerns in requirements, discovery of crosscutting concerns from software requirement specifications, and characterization of crosscutting concerns in requirements.

3.1 Identification of Requirement Crosscutting Concerns

E. Baniassad et al. have proposed an approach to handling crosscutting concerns in relatively well organized requirement documents [6]. This approach starts with a careful study of the dominant decomposition of a given requirement document, because understanding dominant decomposition is an important prerequisite of handling crosscutting concerns. This approach consists of four phases – identifying crosscutting concerns in requirements, capturing crosscutting concerns as requirement aspects (the term ***aspect*** is borrowed from AOP), specifying composition rules of aspects, and analyzing aspects. In the phase of identifying crosscutting concerns, requirement developers evaluate requirement documents such as software requirement specifications to find crosscutting concerns. Aspectual terms normally describing quality attributes of a system like security and performance, impact descriptions of one requirement's influence

over other requirements, and scattered or repeated descriptions of requirements appearing in multiple places throughout the requirement document are often considered indicators of crosscutting concerns in requirements. In the phase of capturing crosscutting concerns, the authors suggest that requirement developers should reorganize the requirement document in some modularized fashion in order that each concern, including each crosscutting concern, is captured in a separate requirement artifact such as section or paragraph, reducing scattering and tangling as much as possible. The impacts of one crosscutting concern on other concerns, for example one concern constraining the fulfillment of another concern, are explicitly specified as composition rules in the phase of composition. The composition rules can be specified in a way analogous to the pointcut specification in AOP. Finally in the phase of analysis, identified crosscutting concerns are evaluated to explore their scope of influence, reveal their potential conflicts between each other and provide reasonable trade-off solutions.

Two types of representation of crosscutting concerns in requirements are presented by Baniassad et al [6] – crosscutting concerns that are specified in a scattered manner and crosscutting concerns that are specified in a non-scattered manner. Consider the following sample requirements extracted from Baniassad et al. [6]:

1. Pay interest of a certain percent on each account making sure that the transaction is fully completed and an audit history is kept.

2. Allow customers to withdraw money from their accounts, making sure that the transaction is fully completed and an audit history is kept.

Two crosscutting concerns are presented in this example – "transaction is fully completed" and "audit history is kept". These two crosscutting concerns are specified in a scattered manner.

A non-scattered version of this example is also presented by Baniassad et al [6]. It is extracted as follows:

1. Pay interest of a certain percent on each account.
2. Allow customers to withdraw money from their accounts.
3. All transactions are fully completed.
4. Keep an audit history of all transactions.

In this example, "transaction is fully completed" and "audit history is kept" are specified in a modularized manner – they are specified in their own sections. This version reduces scattering in the requirement document but it makes the impacts of these two crosscutting concerns implicit. "All transactions" can not be determined until the requirement document is complete. In addition, tangling between the two crosscutting concerns (transaction completion and audit) and the other two base concerns (pay interest and withdraw money) becomes implicit as well, making conflict detection and trade-off

negotiation between them even harder. We can see from this example that whether "transaction is fully completed" and "audit history is kept" conflict with each other and, if they do, where they conflict, can not be known until all transactions are identified.

To prompt non-scattered specification of crosscutting concerns and keep the impacts of crosscutting concerns explicit at the same time, E. Baniassad et al. have proposed an AO solution which records requirement crosscutting concerns in an intuitive and modularized manner [6]. In this solution, each crosscutting concern is specified in two separate sections – one for the concern itself and the other for the concern's impacts. We use another example from their work [6]:

1. Pay interest of a certain percent on each account.
2. Allow customers to withdraw money from their accounts.
3. All transactions are fully completed;

   List of transactions includes (1) and (2).
4. Keep an audit history of all transactions;

   List of transactions includes (1) and (2).

The text in italic style is the specifications of crosscutting concerns' impacts. We can see that in addition to the reduced scattering, the previously implicit impacts of each crosscutting concerns are specified in an explicit manner as well. From the above examples we can see that E. Baniassad et al. [6] suggest structuring requirement

documents so that crosscutting concerns are captured separately and their otherwise implicit impacts are specified explicitly. The two types of representation of crosscutting concerns in requirements that they present [6] contribute to our classifications of crosscutting concerns, which will be introduced in Chapter 4.

The approach described by Baniassad et al. in [6] assumes well-organized requirement documents in which concerns are separately specified in their own parts of the requirement artifacts such as separated sections and paragraphs. However, such requirement documents are not often available. This prerequisite limits to some extent the applicability of this approach. Also, studying requirement artifacts to understand the dominant decomposition, searching for indicators of crosscutting concerns, and restructuring requirement artifacts, are laborious, tedious and error-prone jobs. There is no evidence to show that this approach can be applied to real requirement documents in a productive manner.

In addition to dealing with crosscutting concerns in requirements, the approach proposed by Baniassad et al. in [6] also addresses identifying and managing crosscutting concerns at the architecture stage and depicts the interaction between the crosscutting concern identification and management activities at the requirement stage and the activities at the architecture stage. This part is not very relevant to our experiment, and thus will not be discussed further in this thesis.

3.2 Crosscutting Concerns in Requirement Specifications

L. Rosenhainer [37] suggests two techniques which can be used to identify requirement level crosscutting concerns from existing software requirement specifications. One technique is to look for crosscutting concerns during requirement inspection in a manual manner. In this technique, a non-functional requirement is selected by experience as a starting point. Manual search against the whole requirement specification is done to find if other requirements are constrained by this requirement. If so, this requirement is marked as a crosscutting concern. The other technique uses information retrieval programs to search for crosscutting concerns in a semi-automatic manner. In this technique, an information retrieval program is used to search the whole requirement specification for pre-defined statements or terms that are believed to be crosscutting. Although the searching job can be automated to some extent, examination of each candidate crosscutting concern must be done manually. Thus both techniques require intensive manual work.

We believe that these two proposed techniques have defects. First, they are based on the assumption that crosscutting concerns in requirements come from non-functional concerns which can be manually picked up as starting points or can be covered by a set of pre-defined statements or terms. Even used together, they will not discover all crosscutting concerns in a requirement specification. This is partly because both non-functional concerns and functional concerns can be crosscutting [6]. Also, a given crosscutting concern may be expressed differently in different places in a requirement

specification. It may not be possible to create a pre-defined set of statements or terms that covers all the forms of expressions of all crosscutting concerns that may appear. Both techniques require picking all suspects that might be crosscutting. If a crosscutting concern fails to be picked up as a starting point, it will not be identified as a crosscutting concern using either of these two techniques. Moreover, since both techniques involve a lot of manual work, the efficiency of them is problematic. For example, suppose we adopt the first technique. If we have picked 5 non-functional requirements as starting points, we have to go through the requirement specification 5 times – once per starting point. A real world requirement specification often contains far more than 5 non-functional requirements. This technique is obviously labor and time intensive. L. Rosenhainer [37] does not provide any evidence about the efficiency of the two proposed techniques.

In addition to the two proposed techniques which we have discussed above, four stages when identification of crosscutting concerns in requirements is conceivable are pointed out by Rosenhainer in [37]. They are:

1. Identifying crosscutting concerns during requirements modeling.

2. Identifying crosscutting concerns while producing software requirement specifications.

3. Identifying crosscutting concerns from existing software requirement specifications.

4. Identifying crosscutting concerns when they are detected during the development process.

L. Rosenhainer [37] emphasizes the third stage because he believes that there are countless existing software requirement specifications which are built without consideration of crosscutting concerns, and aspect-oriented requirement engineering techniques for stages (1) and (2) will not be widely accepted in the near future. However, we are very doubtful that mining crosscutting concerns from existing requirement documents can be productive in terms of costs and benefits, especially when mining crosscutting concerns from existing requirement specifications that are built without awareness of crosscutting concerns. If these requirement specifications are built without aspect-oriented thinking, they are very likely not aspect-mining friendly, for example, describing a same concern using different terms and phases in different locations, making aspect mining harder than it might be.

3.3 Taxonomy of Requirement Crosscutting Concerns

Having observed that various approaches are proposed in the aspect-orientation community using different concepts and notions about what a requirement level crosscutting concern is, N. Niu et al. [29] point out that these approaches may not converge any time soon. They conclude that attempts to make a unified approach to requirement level crosscutting concerns will be arduous and unproductive in most cases. Instead of giving a concise and closed definition of crosscutting concerns in requirements, N. Niu et al. [29] apply domain analysis to existing aspect-oriented requirement engineering approaches and propose an approach using a feature diagram to characterize what features a requirement level crosscutting concern must have or may have. The feature diagram as proposed in [29] shows that a crosscutting concern in requirements must be associated with an intent or purpose of existence; a crosscutting concern must be the result of a chosen dominant decomposition scheme; both non-functional concerns and functional concerns can be crosscutting concerns. The feature diagram also shows that crosscutting concerns in requirements can manifest as design or implementation artifacts or as influential factors that have impacts on development activities or decision making.

In addition, inspired by the Twin Peak model proposed by B. Nuseibeh [30], N. Niu et al. [29] propose an asymmetric, iterative and parallel approach to the development of concerns. In this approach, one process creates and expands base (non-crosscutting) concerns and another process develops crosscutting concerns. These two processes do not

exist in isolation. There are interactions between them – identification of crosscutting concerns improves the base structure's modularity and a well-modularized base structure eases discovery of crosscutting concerns. Also, since it would be imprudent (even impossible) for one to fully develop either all base concerns or all crosscutting concerns independently before developing any of the other group of concerns, these processes proceed iteratively and in parallel.

3.4 Summary

In this chapter, we discussed research work that is significant to our experiment. In [6], a general approach of identifying crosscutting concerns in requirements is proposed and a definition of requirement aspect is given. In [37], some unique characteristics of crosscutting concerns in requirements are discussed and opportunities to identify and manage crosscutting concerns in requirements are enumerated and explained. In [29], a feature diagram is proposed that depicts what characteristics requirement crosscutting concerns must have or may have. An iterative and parallel approach of concern development in the requirement engineering stage is proposed in [29] as well. The research work summarized in this chapter has great influence on our experiment design. For example, our overall experiment procedure is inspired by the approach introduced in [6]. We will present the details of our experiment design in Chapter 4.

Chapter 4

Experiment Design

In this chapter, following an overview of our experiment, a brief introduction to Software Requirement Specifications will be given. The INFOD project will be introduced, along with the target software requirement specification – the INFOD base specification. Then, our experiment approach will be proposed and discussed. The types of experiment results will be explained and discussed as well.

4.1 Experiment Overview

In our experiment, techniques from the AORE literature [6] as well as techniques we have developed will be used to identify crosscutting concerns from an existing software requirement specification, which was developed without awareness of crosscutting concerns. The identified crosscutting concerns will be analyzed and discussed. Characteristics of crosscutting concerns in requirements found in our experiment will be proposed. Furthermore, four speculations that we propose below will be considered in light of our experiment results. The speculations are listed as follows:

1. *Crosscutting concerns in requirements are largely different from crosscutting concerns in code in terms of their representation.* In code, a crosscutting

concern normally presents as scattered and tangled code pieces. However, in requirements, particularly requirement specifications, the representation of a crosscutting concern – the specification of this concern – can be scattered or well localized.

2. *The influence of a requirement level crosscutting concern is normally much broader than the influence of a code level crosscutting concern.* A requirement level crosscutting concern can have influence on multiple requirement artifacts as well as architectural design artifacts and code modules.

3. *The influence of requirement level crosscutting concerns can be specified either explicitly or implicitly, or both.* In the implicit case, knowing which concerns are influenced is sometimes postponed to later development stages such as the design stage.

4. Since the representation of requirement level crosscutting concerns can be well localized and the influence of requirement level crosscutting concerns can be implicitly specified, *identification of requirement level crosscutting concerns is much more complicated than identification of code level crosscutting concerns.*

4.2 Software Requirement Specifications

Preparation of a software requirement specification is one of the most widely adopted techniques for the documentation of requirements. A software requirement specification aims at providing an unambiguous and complete description of a software system to be developed. A software requirement specification should describe as precisely and completely as necessary the behaviors of the system under construction under various conditions and from different perspectives. It states the functionalities and capabilities the system must provide. It also specifies the constraints the system must respect while providing its expected services, e.g. quality attributes to hold and industry standards to align with. A software requirement specification should only address what must be built, with minimal description of implementation. A recommended practice for software requirement specification is IEEE Std. 830 – 1998 (under revision) [14]. It suggests that a good software requirement specification should address factors such as functionality, external interfaces, performance, and quality attributes of the system but should not contain design, construction, testing, or project management details other than known design and implementation constraints. It also suggests that a good software requirement specification should be correct, unambiguous, complete, consistent, verifiable, modifiable and traceable. In a highly disciplined process e.g. the waterfall model [38], the software requirement specification is one of the most important artifacts in the software development lifecycle. It is the basis for all subsequent development activities such as project planning, design, and coding, as well as the foundation for testing and user documentation [43].

Admittedly, few software projects successfully follow a highly disciplined process like the waterfall model, emphasizing the importance of early and complete software requirement specifications. Agile software development [25] is a case in point. Instead of establishing all the details of a requirement specification up front, agile projects put more value on evolutionary refinement and adaptation of requirements – the requirements evolve over the early iterations of the development process [26]. As this thesis is focused on the identification and analysis of crosscutting concerns in software requirement specifications, discussion about agile projects is beyond the scope of this thesis. However, the results of this thesis may still be applicable to agile projects. In fact, requirement activities do exist in agile projects. For example, user stories are used to capture user requirements in Extreme Programming [25] projects. To what extent the results of this thesis are applicable to agile projects needs to be evaluated in future studies.

Our experiment is carried out with respect to a representative real world software requirement specification – the INFOD base specification [10], which is informal, and is evolving and being developed without explicit consideration of crosscutting concerns. Therefore, the output of our experiment is expected to be applicable mostly to similar requirement documents. The INFOD project [15] and the INFOD base specification [10] will be introduced in the following section.

4.3 INFOD and the INFOD Base Specification

INFOD stands for Information Dissemination. The INFOD project is hosted in the Open Grid Forum (OGF) [31], which is an open community committed to driving the rapid evolution and adoption of applied distributed computing. The INFOD project aims at proposing a recommended standard of timely information dissemination for the OGF community.

The INFOD project has proposed a Valuable Information at the Right Time (VIRT) [13] model to support the timely delivery of valuable information in a wide range of applications. In traditional information dissemination models such as Java Message Service (JMS) [18], publishers provide information as messages and consumers receive messages. Consumers also specify what information from which publishers is of interest by means of subscriptions. The INFOD VIRT model differs from the traditional information dissemination model. It consists of publishers, consumers, subscribers and the INFOD registry. In the INFOD VIRT model, publishers are responsible for generating messages and sending messages to designated consumers; consumers act as recipients of messages from publishers; and the role of creating subscriptions is separated and assigned to subscribers. The INFOD registry is used to capture information about publishers, consumers and subscribers. This information is used to direct information flows. The INFOD VIRT model makes use of vocabularies and constraints to represent Conditions of Interest (COI) – ways of characterizing which information is needed when and by whom. The structures of properties of classes of publishers, consumers and

subscribers are described in terms of property vocabularies and the actual properties of publishers, consumers, or subscribers are described as property vocabulary instances. Publishers, consumers and subscribers are able to specify constraints on each other with reference to the properties to determine whom they want to interact with, and on what basis. Data vocabularies are used to define the structure of messages to be published. For example, a car dealer can send promotion flyers to residents who live within 30km distance and are interested in SUVs and at the same time some local residents can set their preference to accept flyers regarding GM cars only. In this example, the car dealer is an information publisher and target residents are consumers. The conditions *living within certain distance* and *with certain purchase preference* are specified as vocabularies. Each INFOD system matches publishers and consumers with respect to subscriptions in the system as well as property constraints of publishers and consumers. In addition, the INFOD VIRT model provides the capability to adapt information flow when properties of publishers, consumers and subscribers are changed. For example, if Mr. Smith was a target resident in the above example and accepted promotion flyers from the car dealer, he might change his preference to sedans. The INFOD system will react to this change and notify the car dealer not to send flyers to Mr. Smith any longer. This is a unique feature of INFOD [10]. The INFOD VIRT model is designed to channel information flow from thousands of sources to thousands of destinations with great flexibility.

The INFOD base specification specifies the INFOD VIRT model. Various documents available at [15] such as discussion threads, meeting minutes and errata show that the

specification is still in an immature status, with most effort spent on eliciting and clarifying the functionalities and capabilities of the model. In our experiment, we use the latest release version of the INFOD base specification at the time our study started as our study target. It was released on May 5, 2007 and published on July 3, 2007. We have reviewed a newer draft of the specification [11] that is under development. However there is no difference between this new draft and the version we are currently using from the point of view of handling aspects, nor is there a large change in the aspects to be considered.

Although not explicitly specified, there is only a single registry in a given INFOD system that captures information about external participants and vocabularies, matches publishers and consumers using specific subscriptions, and notifies publishers about which messages are to be sent to which consumers. We refer to this single registry in a given INFOD system as "the" INFOD registry throughout this thesis.

4.4 Experiment

4.4.1 Criteria of Crosscutting Concerns in Our Case Study

E. Baniassad et al. give a definition of crosscutting concerns in requirements in their work [6], as we quoted in Chapter 2. This definition is not immediately applicable to our case study. It is made based on the assumption that requirement documents are well structured – where possible, each concern is captured separately in a requirement artifact.

However, our study target, the INFOD base specification, is less structured – some concerns are specified in multiple sections or paragraphs while some other concerns are specified within one section or paragraph. We also noticed in our initial investigation of the INFOD base specification that a crosscutting concern in requirement specifications may not necessarily be specified in a scattered manner. For example, consider the following requirement statement extracted from the INFOD base specification.

"INFOD uses existing security mechanisms to ensure that the dissemination of information happens according to security policies." Line 175 in Appendix B

This requirement statement appears once in its own section in the INFOD base specification. This statement captures a security related concern that has a system wide influence on the INFOD. To reason about any information dissemination related concern, this concern must be taken into account. This concern is therefore considered a crosscutting concern.

We revise the definition of crosscutting concerns in requirements provided in [6] so as to accommodate the situation where a crosscutting concern appears just in one place in a requirement document. Our working definition of crosscutting concerns in requirements is given as follows.

A crosscutting concern in requirements is a concern that is found in one place or multiple places in a requirement document but has broadly scoped influence, either explicit or implicit, on multiple requirement artifacts. To completely understand these requirement artifacts, this crosscutting concern must be taken into account.

This working definition is still too abstract to be applied to the INFOD base specification. We derive four criteria from this definition to help to discover crosscutting concerns in the INFOD base specification. These four criteria are:

1. Requirement statements that have broadly scoped influence such as quality attributes of a system can be seen as indicating crosscutting concerns.

2. Requirement statements that describe influence of a concern over other concerns can be seen as indicating crosscutting concerns.

3. Requirement statements that appear in multiple places referring to similar terms, concepts or behaviors can be seen as indicating crosscutting concerns.

4. Requirement statements that when they are modified, some other statements in a requirement document must be changed accordingly can be seen as indicating crosscutting concerns.

Where these indicators appear in the INFOD base specification will be studied with great care to see if they imply crosscutting concerns.

4.4.2 Process of Our Case Study

Our experiment follows the general guideline proposed in [6]. According to this guideline, identification of crosscutting concerns should start with identifying the dominant decomposition in the INFOD base specification. Each concern in the specification should then be examined with respect to this dominant decomposition. If a concern has influence crosscutting the dominant decomposition structure, it is intended to be considered as a crosscutting concern.

From our initial analysis of the INFOD base specification, we have observed that the specification presents the INFOD system at different levels of abstraction, from abstract and coarse-grained specification to concrete and fine-grained specification. Dominant decomposition at each level of abstraction varies to some degree. Therefore, the examination of dominant decomposition in the INFOD base specification needs to be

done separately at each level of abstraction. We will discuss these levels of abstraction in Chapter 5.

We have observed that the representation of a crosscutting concern in requirements – how it is specified – can be localized or scattered, regardless of its influence over other concerns. We have also observed that the influence of a crosscutting concern in requirements can be specified in an explicit manner, such as having explicit references to its influenced concerns, or in an implicit manner. A classification scheme of crosscutting concerns in requirements with respect to how they are specified and how their influences are expressed is proposed. This classification scheme will be applied to crosscutting concerns. The detailed description of this classification scheme will be provided in Chapter 5.

As discussed in Chapter 3, a parallel and progressive approach of developing concerns at the requirement stage is proposed in [29]. In that approach, one process creates and expands the base model and the other process creates crosscutting concerns that crosscut the base model. The two processes interplay with each other. Identification of crosscutting concerns improves the modularity of the base model and the improved modularity of the base model in turn facilitates the identification and evaluation of crosscutting concerns. This approach is not immediately applicable to our experiment because it aims at building base concerns and crosscutting concerns in the process of eliciting and documenting requirements, while our experiment is designed to identify

crosscutting concerns from existing requirements. Nevertheless, a similar parallel and progressive process does exist in our experiment. Understanding dominant decompositions and identifying crosscutting concerns proceed in parallel. Understanding dominant decompositions helps to identify crosscutting concerns and identified crosscutting concerns promote more comprehensive understanding of dominant decompositions. In this sense, our experiment is not a simple linear process. It proceeds iteratively instead. The understanding of crosscutting concerns and their influence on the system is obtained in a progressive manner.

4.5 Designed Experiment Output

4.5.1 Annotated Version of the INFOD Base Specification

In this experiment, we have annotated all the crosscutting concerns in the INFOD base specification using XML-like tags. A crosscutting concern tag includes an opening tag – **`<x-concern>`** and a closing tag – **`</x-concern>`**. A crosscutting concern tag has a mandatory attribute – id – that is used to assign a unique identifier to a crosscutting concern. The identifier of a crosscutting concern starts with letter X – standing for crosscutting and is followed by 3 digits. The first digit indicates the level of abstraction in the INFOD base specification where a crosscutting concern is located – 0: Vision Level, 1: Concept Level, 2: Architectural Level, 3: Operational Level and 4: Detailed Level. The definition of these levels will be provided in Chapter 5. The remaining two digits form a serial number of a crosscutting concern at a level of abstraction. For example, X303

refers to a crosscutting concern that is located at the Operational Level and is the third crosscutting concern being identified at the Operational Level. The order in which a crosscutting concern is identified may not be the same as the order it appears in the INFOD base specification. In addition, a crosscutting concern tag may have an optional *'part'* attribute that is used to indicate how many scattered parts this crosscutting concern has if this crosscutting concern is scattered in the INFOD base specification. The *'part'* attribute looks like a fractional number. The denominator indicates the total number of the scattered parts of a crosscutting concern throughout the INFOD base specification. The numerator indicates the serial number of this occurrence of the crosscutting concern in the INFOD base specification. All attributes of a crosscutting concern tag are specified in its opening tag. A complete crosscutting concern tag with the enclosed crosscutting concern will look like the following:

```
<x-concern id=X304 part=1/2>
Each entry has a name and description, both of which are optional, not
necessarily unique and have string values. They are also both expected to
be meaningful to humans.
</x-concern id=X304>
```

The crosscutting concern shown above is located at Operational Level and it is the fourth identified crosscutting concern at Operational Level. So its identifier is X304. This crosscutting concern has two scattered parts in the INFOD base specification and the

specification enclosed within this tag is the first occurrence of this crosscutting concern in the INFOD base specification. So the part attribute is 1/2. We will not provide a name for each crosscutting concern tag because some crosscutting concerns are too complicated to be described in a few words. Lengthy names would make crosscutting concern tags too hard to read. Instead we provide, at the end of this thesis, a list of crosscutting concerns with a detailed description of each crosscutting concern. This list of crosscutting concern will be discussed at the next sub-section.

We have also restructured the INFOD base specification a little bit so that the structure of the specification becomes clearer. The restructure will not change the meaning of the original text. Furthermore, in order to make the specification terse, we have removed some text such as glossaries and appendixes which we believe is not relevant to the identification of crosscutting concerns. The restructured and annotated version of the INFOD base specification will be provided separately as Appendix B of this thesis.

4.5.2 List of Crosscutting Concerns

We will produce a list of crosscutting concerns identified in the INFOD base specification during our experiment. Each crosscutting concern is recorded in a tabular format which we designed. This tabular format is shown in Table 4.1.

Id and short name of a crosscutting concern	
Description	
Description of a crosscutting concern	
Where It Appears	
Where in the INFOD base specification a crosscutting concern appears in terms of line numbers, figure id, etc.	
Concerns Influenced	
Which concerns are influenced by a crosscutting concern	
Classification based on Crosscutting Representations	
Rationale of why a crosscutting concern is classified as it is in terms of crosscutting representation	

Table 4.1 Crosscutting Concern Record Table

The identifier and the short name of a crosscutting concern are provided in the first row of this table. The description field records the in-depth description and discussion of this crosscutting concern for example why this concern is seen as a crosscutting concern. The where-it-appears field records the occurrences of a crosscutting concern throughout the INFOD base specification. An occurrence can be specified as line numbers if a crosscutting concern is specified in a textual format, can be identifiers of figures if a crosscutting concern is specified in a graphic format, etc. The concerns-influenced field is used to record which concerns in the INFOD base specification are influenced by this crosscutting concern. The classification-based-on-crosscutting-representation field

records the classification of a crosscutting concern in term of its crosscutting representation. The rationale of why this crosscutting concern is classified to a particular category of crosscutting representation is provided as well.

The list of crosscutting concerns is divided into three sub lists, each addressing crosscutting concerns at a level of abstraction in the INFOD base specification. This list of crosscutting concerns is provided in Appendix A of this thesis.

4.6 Summary

In this chapter, we briefly introduced the software requirement specification and its importance in highly disciplined development processes. The INFOD project is introduced, as is the target software requirement specification in our experiment – the INFOD base specification. The experiment approach is then discussed, including our process of identifying crosscutting concerns from existing software requirements specifications, levels of abstractions in the INFOD base specification and classifications of crosscutting concerns. Finally, the designed types of output of our experiment are introduced, including a restructured and annotated version of the INFOD base specification (Appendix B) and a list of all crosscutting concerns in the INFOD base specification (Appendix A).

Chapter 5

Experimental Evaluation

In this chapter, the results of our experiment will be evaluated and discussed. First of all, different levels of abstraction presented in the INFOD base specification will be discussed. Then, a classification of crosscutting concerns found in the INFOD base specification will be proposed. Following will be discussion of this classification with reference to some of the crosscutting concerns found in the INFOD base specification. A list of sample crosscutting concerns found in the INFOD base specification will be given, along with an in-depth analysis for them. Finally, characteristics of crosscutting concerns in requirement specifications that are generalized from our experiment will be presented.

5.1 Levels of Abstraction in the INFOD Base Specification

In a software requirement specification, the system under construction is normally described at different levels of abstraction. A specification normally starts with high-level business objectives which the system is expected to achieve [43]. Then, the system is specified from a coarse-grained description to more fine-grained, concrete and verifiable specifications. These different levels of abstraction are sometimes separated into different requirement documents [43]. The way in which the system is specified can be different at different levels of abstraction. This may result in different dominant decomposition

structures of a requirement specification at different levels of abstraction. For example, the primary concerns of a system can be organized with respect to types of business objectives, such as must-have and nice-to-have, and then the rest of the specification can be organized around these high level business objectives. Alternatively, the primary concerns of a system can be organized with respect to groups of use cases when the specification is based on use cases. As proposed in [6], different dominant decomposition structures can lead to different sets of crosscutting concerns. We believe that analysis and evaluation of crosscutting concerns should be done separately at each level of abstraction.

In the INFOD base specification, we have found 5 levels of abstraction, namely Vision Level, Concept Level, Architectural Level, Operational Level and Detailed Level. All these 5 levels are documented within the same requirement specification – the INFOD base specification [10]. Discussion about these levels of abstraction is provided as follows.

5.1.1 Vision Level and Concept Level

The Vision Level is the highest level of abstraction in the INFOD base specification. The specification at the Vision Level focuses on high-level business objectives of INFOD. The overall business objective of INFOD is proposed. That is, INFOD will provide the objective of Valuable Information at Right Time (VIRT) [13] to a wide range of applications.

Following the Vision Level in the INFOD base specification is the Concept Level. More details of INFOD are provided at this level. A conceptual model of the system – the INFOD VIRT model – is proposed at this level to support the abstract and intangible business objective introduced in the Vision Level. This model illustrates principal INFOD VIRT concepts and acts as a source of inspiration for formulating the INFOD system. This model describes the overall functionality of the INFOD system in terms of VIRT entities, e.g. publishers, consumers and subscribers, and relations among these entities. The specification of this model is organized with respect to VIRT entities – the roles VIRT entities play and the interaction between and among VIRT entities. The INFOD VIRT model defines a mechanism of information dissemination, through which the high-level business objective proposed at the Vision Level can be satisfied.

At the Vision Level and the Concept Level, the INFOD base specification intends to clarify what INFOD is about and what is the primary business objective of INFOD, rather than proposing a concrete system to be developed. The specifications at these two levels are too abstract to have clear dominant decomposition structures.

5.1.2 Architectural Level

The INFOD VIRT model proposed at the Concept Level gives rise to an INFOD system, which is formulated at the Architectural Level. Architecture in software engineering literature often refers to descriptions about high level, coarse-grained organizations of a system, selections of structural elements and their interfaces by which a system is

composed, as well as other important subjects such as a system's operational context [23]. Here we use the term "architecture" in a similar sense: the INFOD base specification at this level describes in coarse granularity the primary sub-systems of the INFOD system, the functionalities assigned to each sub-system, interactions among sub-systems, and the boundary of the INFOD system.

At the Architectural Level, the INFOD base specification addresses the system from two perspectives – a structure perspective and a constraint perspective.

From the structure perspective, the organizational structure of the INFOD system is specified. The most important subsystem introduced at this level is a centralized INFOD registry. The external participants, including publishers, consumers, and subscribers, as well as external data sources, also become subsystems that exist outside the INFOD registry and participate in information dissemination. The INFOD registry captures information about the INFOD VIRT entities as resources, matches publishers and potential consumers according to the captured information, and sends notifications to publishers about which messages should be sent to which consumers. Publishers provide required information and act as sources of information flow. Consumers consume information and act as destinations of information flow. Subscribers set up subscriptions in the INFOD registry so as to allow the registry to direct information flow as specified in a given INFOD system.

From the constraint perspective, a key concept in the INFOD VIRT model – Condition of Interest (COI) – is specified in terms of property constraints and data constraints. For example a publisher can specify property constraints against properties of a set of consumers to whom information will be delivered. Figure 5.1, which is replicated from Figure 2 in the INFOD base specification, shows the constraint relationships among publisher entries, consumer entries, subscriber entries and data source entries.

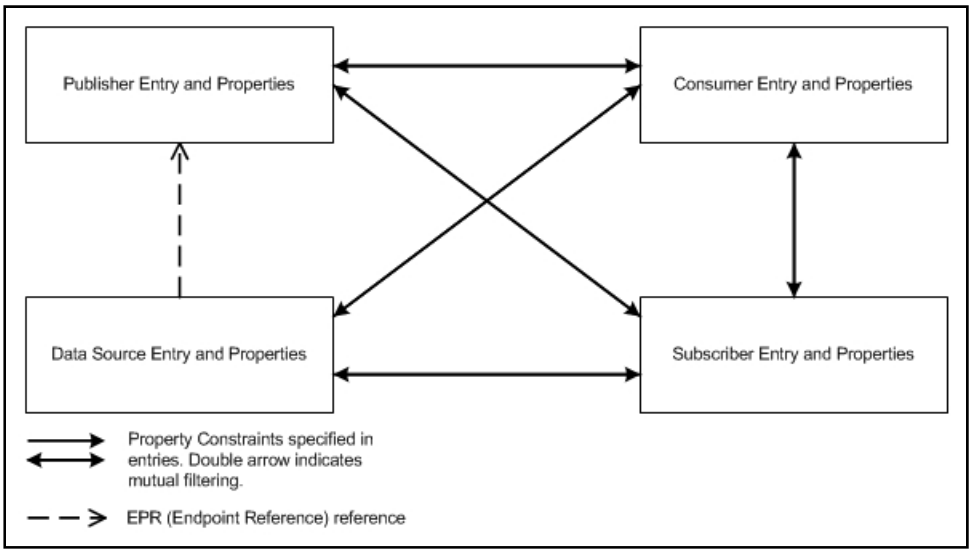

Figure 5.1 Property Constraints among INFOD Entries

As shown in the above figure, publishers, consumers, subscribers and data sources (where information comes from) can have property constraints against each other so that the routes of information dissemination within a given INFOD system can vary

dynamically. For example, a consumer can have property constraints against publishers so that only information flows from certain publishers are accepted. Data sources represent sources of information flow belonging to publishers. The belonging relationship between data sources and publishers is represented as EPR (Endpoint Reference) link from data sources to publishers. How such dynamic information dissemination is achieved in an actual INFOD system is beyond the scope of this thesis and is not presented.

The INFOD base specification at the Architectural Level, considered from the structure perspective, is organized with respect to four classes of principal INFOD sub-systems: the INFOD registry, and external publishers, consumers and subscribers. The structure and functionalities of the INFOD registry are explicitly specified in the INFOD base specification. However, the structures and functionalities of publishers, consumers and subscribers are not fully provided. Therefore, we can speculate that the specification at this level is focused on specifying the central INFOD registry. As a result, the dominant decomposition of the specification at this level is mainly associated with the INFOD registry – what are the functions of the INFOD registry, what types of resources are held in it, how external participants like publishers and consumers are represented in the INFOD registry, and what are the interactions between external participants and the INFOD registry.

The INFOD base specification at the Architectural Level, considered from the constraint perspective, specifies property constraints applied to INFOD VIRT entities and data constraints applied to subscriptions. The specification is organized as property constraints and data constraints, which are two separate categories. Within the property constraints category, the specification is further organized with reference to elements in the INFOD registry that are depicted in the structure perspective.

5.1.3 Operational Level

At the Operational Level, the focus of the INFOD base specification shifts from the overall structural organization and functionality allocation of the entire system to how sub-systems provide their assigned functionality in terms of more concrete operations. What constrains the provision of the operations of sub-systems is specified as well.

The INFOD base specification at this level focuses on how the INFOD VIRT entities are captured and managed as resources in the INFOD registry, what properties these entities may have and what operations the INFOD registry provides in order to manage different types of resources. In addition, a basic dependency rule that governs creation, modification and removal operations on types of resources is specified as well.

The types of resources managed in the INFOD registry are grouped according to their contributions to the INFOD VIRT model. The types Publisher Entry, Consumer Entry and Subscriber Entry represent external participants of information dissemination and are

grouped together. The types Data Source Entry and Data Vocabulary represent where information comes from, what information should be sent and when to send it, and are grouped together. The types Property Vocabulary and Property Vocabulary Instance are used to describe properties and types of properties of entries and are grouped together. The type Subscription represents COI. It is grouped by itself. The specification at the Operational level is organized with respect to these groups of types of resources. Within a group, major attributes and operations of each type of resource are specified.

5.1.4 Detailed Level

At the Detailed Level, the specification goes even further down to the fine detail of the INFOD system. The structure of each type of resource is defined in detail, including data types of attributes and allowed range of values for each attribute. The operations the INFOD registry provides to each type of resources are defined in detail as Application Programming Interface (API) calls. The parameters of API calls, the results of API calls and the behavior of the INFOD registry in response to API calls are specified in fine detail. In addition to the fine details of API calls provided by the INFOD registry, the notification from publishers to consumers and the notifications from the INFOD registry to publishers, consumers and subscribers are specified in fine detail as well.

The specification at this level is organized as two categories – API calls and notifications. The specification of API calls is structured with respect to types of resources. The API calls corresponding to a type of resource are organized together as an interface, for

example the Managing Publisher Entries interface. The specification of notifications is structured with respect to directions of notification – from publishers to consumers and from the INFOD registry to publishers, consumers or subscribers.

5.1.5 Varied Dominant Decompositions at Each Level of Abstraction

Analysis of the INFOD base specification supports our claim that the dominant decompositions of the INFOD base specification are different at different levels of abstraction. The specification at the Vision Level has no clear decomposition structure, since it is just a simple statement about what goals INFOD aims to achieve. A domain model of INFOD is proposed at the Concept Level. Although the specification at this level is organized with respect to distinct domain entities, the dominant decomposition is also not clearly defined. A tangible system is introduced at the Architectural Level. The dominant decomposition at this level is related to the principal components in the system, in particular the INFOD registry. At the Operational Level, the specification takes a view point of types of resources managed in the INFOD registry and provides more detailed descriptions of these types of resources as well as operations associated with them. The specification at this level is grouped with respect to closely related types of resources and within each group the specification is further organized by types of resources. At the Detailed Level, the specification focuses on how the INFOD registry provides its services – managing resources, matching publishers and consumers, and notifying publishers. The specification at this level is divided into two parts – service interfaces provided by the INFOD registry and notifications from the INFOD registry to external participants, e.g.

publishers and consumers. The specifications of service interfaces are organized with respect to types of resources managed in the INFOD registry and the specifications of notifications are organized with respect to types of external participants. We can see that the dominant decompositions at the Operational Level and the Detailed Level of the INFOD base specification are mainly structured with respect to the types of resources managed by the INFOD registry.

5.2 Classification of Crosscutting Concerns in INFOD

We have found in our experiment that the representations of requirement level crosscutting concerns are largely different from the representations of code level crosscutting concerns. In code, crosscutting concerns manifest themselves as scattered and tangled code pieces that cut across the structure of a particular code level dominant decomposition. In requirement specifications, crosscutting concerns manifest in a less well defined and less obvious fashion. In a requirement specification, a crosscutting concern does not have to be scattered to crosscut other concerns [37]. The representations of crosscutting concerns in requirement specifications – statements to specify them – can be either scattered or well localized. A crosscutting concern is scattered if its specification appears in multiple places of a requirement document. If the specification of a crosscutting concern appears once in one place of a requirement document, we say this crosscutting concern is localized. For example, consider the following crosscutting concern identified in the INFOD base specification:

"A basic dependency rule governs the creation, modification and removal of resources within the INFOD registry: only resources that are registered in the INFOD registry can be referenced."(X310)

This concern is specified in a well localized manner within its own paragraph. However, this concern has broad influence on the creation, modification and removal operations of all types of resources managed in the INFOD registry. To understand the behavior of any of these operations, this concern must be taken into account. This concern therefore is a crosscutting concern.

In addition to the scattered or localized representations of crosscutting concerns in requirements, we have also noticed that the influence of crosscutting concerns can be expressed explicitly or implicitly. If the influence of a crosscutting concern is expressed explicitly, the concerns influenced by this concern are either explicitly referenced in the specification of this crosscutting concern, or can be known immediately by studying where this crosscutting concern appears. On the other hand, if the influence of a crosscutting concern is expressed implicitly, the concerns influenced by this crosscutting concern can not be known immediately until the potentially influenced concerns are eventually discovered, or can not be revealed at the requirement stage. For example, consider the following crosscutting concern identified in the INFOD base specification:

"The act of creation involves storing information and returning the EPR of the entry. The creation operation will often store the EPR of the external object. This is the only place the external EPR, identifying the external object, is stored. All other references to EPRs are to EPRs of resources." (X305)

This crosscutting concern defines a common behavior that is shared by the creation operations of 'the entry' that explicitly refers to publisher entries, consumer entries and subscriber entries because this statement is included in a section titled – 1.1.1.1 Publisher Entry, Consumer Entry and Subscriber Entry. Therefore, the influence of this crosscutting concern is seen as explicitly expressed.

Based on the above observations, we propose a classification scheme for crosscutting concerns with respect to the ways their specifications are represented as well as the ways their influence is expressed. This gives us four categories for crosscutting concerns in the INFOD base specification shown in Table 5.1, which are applied to the crosscutting concerns identified in the INFOD base specification.

	Localized Specification	**Scattered Specification**
Explicit Influence	XC1	XC3
Implicit Influence	XC2	XC4

Table 5.1 Categories for Crosscutting Concerns in the INFOD Base Specification

The definitions as well as discussions about these four categories for crosscutting concerns in the INFOD base specification are provided as follows.

XC1: A crosscutting concern that is specified in a localized manner. Its influence on other concerns is specified in an explicit manner. Here "explicit" means that all influenced concerns can be enumerated immediately. In case the influence is specified with reference to a collection of concerns, such as all of or each of, all concerns in the collection must be immediately enumeratable concerns.

For example, consider the following crosscutting concern identified in the INFOD base specification:

"The messages flow directly from the publishers to the consumers making use of a notification system similar to WS-Notification." (X203)

This crosscutting concern limits the notification system from publishers to consumers to be a notification system similar to WS-Notification. This concern has influence on both publishers and consumers because both sides of the notification must agree on a certain notification system. We can see that this crosscutting concern is specified in a well localized manner within its own paragraph and with its influenced concerns – publishers and consumers – explicitly referenced. It is therefore an XC1 crosscutting concern.

XC2: A crosscutting concern that is specified in a localized manner. Its influence on other concerns is specified in an implicit manner or is not specified. It can not be determined immediately which concerns are cut across by this crosscutting concern – this must wait until its potentially influenced concerns are eventually discovered. For a crosscutting concern, if its influence is specified with respect to a collection of concerns but this collection of concerns can not be enumerated immediately, or even after the requirement specification is completely developed, this crosscutting concern is seen as an XC2 crosscutting concern. In the worst case, a concern seems to be crosscutting but whether it is or is not a crosscutting concern can not be determined in the requirement stage and has to be postponed to a post-requirement stage. In this case, we also see this concern as an XC2 crosscutting concern.

For example, consider the following crosscutting concern identified in the INFOD base specification:

"INFOD uses existing security mechanisms to ensure that the dissemination of information happens according to security policies." (X206)

This crosscutting concern specifies that security mechanisms should be applied to each INFOD system to enforce the security of information dissemination in that system. This

crosscutting concern has system wide influence – all components which contribute to information dissemination in INFOD may take this crosscutting concern into account. However, at this point, we can not determine which components inside an INFOD system will be affected since different security mechanism and different architecture choices can lead to different sets of affected concerns. This decision can not be made until the system architecture and the security mechanisms are chosen at a later development stage, e.g. the architecture design stage. We can see that this crosscutting concern is locally specified but its influence can not be determined. It is therefore an XC2 crosscutting concern.

XC3: A crosscutting concern that is specified in a scattered manner. Its specification appears in multiple places in a requirement specification and is mixed inline with the specifications of the concerns it cuts across. The influence of XC3 crosscutting concerns is explicit. Either XC3 crosscutting concerns have direct references to their influenced concerns, or their influenced concerns are known immediately at where their specification parts appear.

For example, consider the following crosscutting concern identified in the INFOD base specification:

"As part of the processing of a <replace operation> message, the INFOD registry MUST replace the entire INFOD metadata for the entry representing the <resource>. All previously defined values MUST be deleted. The <replace operation> differs from the

<corresponding creation operation> in that it replaces an existing <resource> and assigns the original EPR to the replaced <resource>." (X414)

This crosscutting concern appears in several places and is tangled with the specifications of the replace operations of publisher entries (line 303 – 306, see Appendix B for line number references), subscriber entries (line 493 – 497), consumer entries (line 682 – 686) and subscriptions (line 889 – 893). We parameterized the specification of this crosscutting concern using placeholders such as <resource> to replace the actual operations and types of resources in order to get the above specification template. This template defines a sequence of actions the operations to replace existing resources in the INFOD registry must perform. This crosscutting concern is specified in a scattered manner and the concerns that it influences are explicit if indirectly given, by where they appear. It is therefore an XC3 crosscutting concern.

XC4: A crosscutting concern that is specified in a scattered manner. Its specification appears in multiple places in a requirement specification. Different from XC3 crosscutting concerns, XC4 crosscutting concerns have implicit influence which can not be known until the potentially influenced concerns are discovered, or which can not be revealed at the requirement stage and knowing which concerns are influenced by an XC4 crosscutting concern has to be postponed to a later development stage such as design or coding.

For example, consider the following crosscutting concern identified in the INFOD base specification:

"When used, the registry MUST notify the external participant about changes relevant in the registry."(X404)
"The notification to an external participant is conditional on the information in its entry." (X404)

This crosscutting concern specifies that the INFOD registry will send notification to external participants, depending on the notification parameters specified in the corresponding creation and modification operations, when relevant changes happen inside the INFOD registry. This crosscutting concern appears in several places in the INFOD base specification such as line 242 – 243, line 350 – 351 and line 1660 and has influence on creation and modification operations provided to external participants as well as components in the INFOD registry responsible for monitoring changes and sending notifications. This crosscutting concern seems to be an XC3 crosscutting concern. However, we can not know what changes in the INFOD registry are relevant to what external participants and thus can not know what components inside the INFOD registry are influenced by this concern and how they are influenced. This crosscutting concern is therefore considered an XC4 crosscutting concern. Elaborated discussion of this crosscutting concern will be provided later in this chapter.

We have found in the INFOD base specification that the scattered specification parts of some XC3 crosscutting concerns are expressed in similar ways, for example the above crosscutting concern, while the scattered specification parts of other XC3 crosscutting concerns are expressed in different ways. This leads to two sub-categories of XC3 crosscutting concerns: homogeneously specified XC3 crosscutting concerns whose scattered specification parts are expressed in similar ways and heterogeneously specified XC3 crosscutting concern whose scattered specification parts are expressed in different ways. For example, consider the following XC3 crosscutting concern identified in the INFOD base specification:

"Each entry is identified in the registry by a unique EPR (endpoint reference)."(X303)

"Data Source Entries, like other entries have their own EPR." (X303)

This XC3 crosscutting concern appears at two places in the INFOD base specification, accordingly line 98 and line 132. Although these two lines look different, they refer to the same concern – each entry is uniquely identified by an EPR. Therefore, this crosscutting concern is a heterogeneously specified XC3 crosscutting concern. In an extreme case where the scattered specification parts of an XC3 crosscutting concern are expressed with great difference – different wording and different key words – it is very likely that they will be treated as several different crosscutting concerns and thus the underlying unity of them will be missed.

As scattered crosscutting concerns, we believe that XC4 crosscutting concerns should have the above two subcategories as well. However, we have found only one instance of XC4 crosscutting concerns – X404 in the INFOD base specification. We therefore do not make these subcategories for XC4 crosscutting concerns.

We believe that the presence of heterogeneously specified XC3/XC4 crosscutting concerns makes identification of crosscutting concerns from requirement specifications even harder. We will discuss this later on in this chapter.

We have also found a correlation between which of the four classifications of crosscutting concerns are used and the levels of abstraction in the INFOD base specification. At higher levels of abstraction of the INFOD base specification such as the Architectural Level and the Operational Level, crosscutting concerns tend to be XC1 or XC2 crosscutting concerns. In fact, all the crosscutting concerns we have identified at the Architectural Level are either XC1 or XC2. At the Operational Level there are only 2 XC3 crosscutting concerns. On the other hand, at the most detailed level of abstraction of the INFOD base specification – the Detailed Level – most identified crosscutting concerns are XC3 or XC4 crosscutting concerns. This is because, to our understanding, the INFOD base specification at the Architectural Level and the Operation Level is largely specified in a coarse-grained manner and most concerns, both non-crosscutting and crosscutting concerns, are raised in a discrete manner, meaning that crosscutting concerns at these two levels tend to be specified in their own rights in separated sections

or paragraphs. At the Detailed Level, the functional details of the INFOD registry – its API functions and notifications – form the main body of the specification and crosscutting concerns that interact with elements in this main body – function calls, parameters, return values, etc. – are spread across this main body. The above correlation is observed in the INFOD base specification. Whether or not it is applicable to other requirement specifications needs to be examined in future work.

5.3 Analysis of Sample Crosscutting Concerns in INFOD

An in-depth analysis of crosscutting concerns in the INFOD base specification is presented in this section. Instead of exhaustively discussing all crosscutting concerns at each level of abstraction (as identified in Appendix A), we choose some representative ones to analyze and present in this thesis. The analysis is provided in the following sub-sections, each addressing crosscutting concerns at a level of abstraction.

As mentioned earlier in this chapter, there is no clear decomposition structure of INFOD at the Vision Level, nor is there one at the Concept Level, so we have skipped identifying and analyzing crosscutting concerns at these two levels. The analysis starts at the Architectural Level, then the Operational Level, and finally the Detailed Level.

5.3.1 Sample Crosscutting Concerns at the Architectural Level

- **Crosscutting Concern X202**

"The main objective of the INFOD registry is to notify publishers which information has to be delivered to which consumer." Line 48 in Appendix B

This crosscutting concern indicates in general a communication agreement between the INFOD registry and publishers, the INFOD registry notifying publishers which information has to be sent to which consumers. Both sides of the communication – the INFOD registry and publishers – must comply with this communication agreement; otherwise the communication is broken. In this sense, this concern constrains the specifications, and thus the behavior, of the INFOD registry and publishers. For example, the INFOD registry must use a notification mechanism which is also supported by publishers and notifications sent to publishers must be understandable to publishers. On the other side, publishers must react to notifications in ways that are expected by the INFOD registry. This concern also implies that publishers do not need to poll the INFOD registry to query which information has to be delivered to which consumers. It has strong influence on the behavior of publishers. This crosscutting concern, therefore, is an XC1 crosscutting concern in that it is specified locally and all its influenced concerns – the INFOD registry and publishers – are explicitly specified.

- **Crosscutting Concern X204**

"Each entry can specify a set of property constraints referencing information related to other entries." Lines 55 – 56 in Appendix B

This concern defines a constraining rule that each entry in the INFOD registry can have a set of property constraints with reference to information of other entries in the registry. This rule is applied to all types of entries in the INFOD registry and thus has broad influence on the definitions of types of entries in the INFOD registry, including publisher entries, subscriber entries, consumer entries and data source entries. For example, instances of these types of entries must have attributes, which can be empty, to represent property constraints. This concern is specified in a localized manner in the INFOD specification. Its influence over other concerns is specified with respect to "each entry", which collectively refers to all types of entries in the INFOD registry. These are well known at this point of specification. We therefore believe the influence of this crosscutting concern is explicitly specified and classify this crosscutting concern as an XC1 crosscutting concern.

- **Crosscutting Concern X206**

"INFOD uses existing security mechanisms to ensure that the dissemination of information happens according to security policies." Lines 175 – 176 in Appendix B

This concern specifies that existing security mechanism should be applied to INFOD so as to ensure that information dissemination happens in a secure manner. Apparently this concern has system wide influence on INFOD. However, we do not know exactly at this point which concerns or which parts of the system are affected. This crosscutting concern

is classified as an XC2 crosscutting concern in that it is locally specified but its influenced concerns are only implicitly provided.

- **Crosscutting Concern X207**

"Lifetime management should provide a mechanism by which resources may be destroyed after a period of time unless the scheduled termination time is extended." Lines 185 – 186 in Appendix B

This concern specifies that the INFOD registry has to provide a mechanism to destroy resources after a period of time. It also specifies that the scheduled termination time can be extended. This concern seems to be a crosscutting concern in the sense that it may have influence on the behaviors of the INFOD registry and external participants such as publishers and consumers. However, it is impossible to decide whether or not this concern is crosscutting at the requirement stage. This is because this concern is specified in so general a way that there may be several different solutions to satisfy this concern. For example, to extend the scheduled termination time of a publisher entry, either (1) the associated publisher (an external participant) sends keep-alive signals periodically to the INFOD registry to extend the scheduled termination time of this publisher entry or (2) the INFOD registry actively monitors the activities of the associated publisher so that the scheduled termination time is not extended unless the publisher has activities in a period of time. In the former case, this concern becomes crosscutting because the INFOD registry must provide an API call to receive keep-alive signals from the external

participants such as publishers and external participants must somehow invoke this API call. In the latter case, the concern can be well localized somewhere in the INFOD registry, monitoring activities of external participants and maintaining lifetime of their associated resources. This concern is specified in a localized manner and whether or not it is a crosscutting concern can not be decided immediately at the requirement stage. It is therefore an XC2 crosscutting concern.

- **Crosscutting Concern X201**

The reason why we put this crosscutting concern as the last example crosscutting concern at Architectural Level is that it arises from consideration of a Figure instead of specification text. Figure 5.2, duplicated from Figure 1 in the INFOD base specification, shows the relationships between types of resources in the INFOD registry.

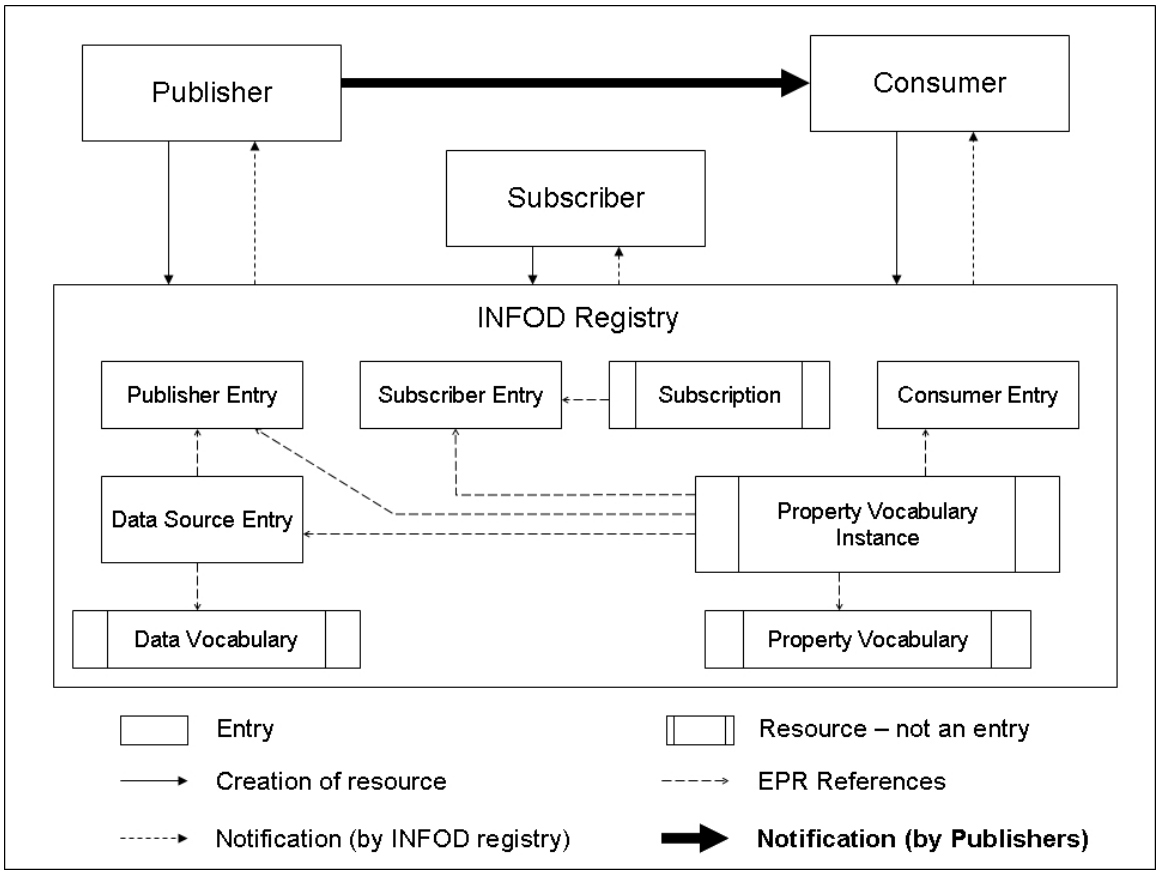

Figure 5.2 INFOD Resources

For example, data source entries have references to publisher entries and data vocabularies, and subscriptions have references to subscriber entries. This crosscutting concern has influences on the definitions of types of resources shown in Figure 5.2. Because this crosscutting concern is represented as a figure, it is well localized and all its influenced concerns are clearly illustrated. It is therefore classified as an XC1 crosscutting concern. To our knowledge, no work in the AORE community deals with crosscutting concerns in requirement specifications which are given as figures. This kind

of crosscutting concern may not be discoverable using text searching techniques. We will discuss this kind of crosscutting concern later on in this chapter.

5.3.2 Sample Crosscutting Concerns at the Operational Level

- **Crosscutting Concern X301**

"Each resource type has calls to create and drop it from the registry."

Line 94 in Appendix B

This crosscutting concern specifies that each type of resource in the INFOD registry has operations to create and to drop their instances from the registry. This crosscutting concern has influence on the services the INFOD registry provides to each type of resource. This crosscutting concern is specified in a localized manner with explicit reference to its influenced concerns. It is therefore classified as an XC1 crosscutting concern.

- **Crosscutting Concern X302**

"Some resources have a call to replace them in the INFOD registry."

Line 95 in Appendix B

This crosscutting concern differs from crosscutting concern X301 in that it refers to the term "some". At this point of the INFOD base specification, we don't know which types of resources have replace operations and which types of resources don't. In this sense, we

classify this crosscutting concern as an XC2 crosscutting concern since its influenced concerns are not explicitly provided.

- **Crosscutting Concern X303**

"Each entry is identified in the registry by a unique EPR (endpoint reference)." Line 98 in Appendix B

"Data Source Entries, like other entries have their own EPR." Line 132 in Appendix B

This concern specifies that every entry in the INFOD registry is identified by a unique EPR (End-Point Reference). It thus has influence on the definitions of types of entries in the INFOD registry. This crosscutting concern is specified in two places in the INFOD base specification and its scattered specification parts are mixed with the concerns it crosscuts. It is therefore an XC3 crosscutting concerns. Also, as we have discussed earlier in this chapter, this XC3 crosscutting concern is a heterogeneously specified XC3 crosscutting concern.

- **Crosscutting Concern X308**

"No information starts flowing in an INFOD system until a subscription is created." Line 147 in Appendix B

This crosscutting concern defines a pre-condition for the information dissemination in an INFOD system. It has broad influence on the components responsible for information

dissemination in an INFOD system. However, at the requirement engineering stage, it is impossible to tell which parts of the system are affected. This crosscutting concern is therefore classified as an XC2 crosscutting concern.

- **Crosscutting Concern X310**

"A basic dependency rule governs the creation, modification and removal of resources within the INFOD registry: only resources that are registered in the INFOD registry can be referenced." Line 161 – 162 in Appendix B

This concern specifies a basic dependency rule that governs creation, modification and removal of resources within the INFOD registry. It constrains creation, replace and drop operations on resources in that each operation must ensure this dependency rule always holds at the end of the operation. This concern is classified as an XC1 crosscutting concern because it is locally specified with respect to "all" creation, modification and removal operations of all types of resources in the INFOD registry.

5.3.3 Sample Crosscutting Concerns at the Detailed Level

- **Crosscutting Concern X401**

"As part of the processing of a CreatePublisherEntry request message, the INFOD registry MUST create an INFOD entry and an EPR representing the publisher entry." Lines 203 – 204 in Appendix B; analogous statements in Lines 426 – 427, 617 – 618, 810 – 811, 1053 – 1054, 1151 – 1152, 1247 – 1248, 1355 – 1356 in Appendix B

This crosscutting concern defines a set of common actions of all creation operations provided by the INFOD registry. It has influence on the specification and thus the implementation of all creation operations. This crosscutting concern is specified in a scattered manner and is mixed inline with the specifications of creation operations. It is therefore classified as an XC3 crosscutting concern. In this example, if all publisher entry related terms are replaced with a place holder, say "resource" or "entry", we can see that all its scattered specification parts are very close to each other, in spite of some minor wording changes. This crosscutting concern is therefore a homogeneously specified XC3 crosscutting concern.

- **Crosscutting Concern X403**

"A property constraint MUST be formulated as an XQuery. … Note that the XQuery statement MUST be encoded correctly, i.e. characters such as '>' would be represented as '>'" Lines 234 – 240, 342 – 348, 457 – 463, 533 – 539, 648 – 654, 722 – 728, 846 – 854, 934 – 941, 1387 – 1393, in Appendix B

This crosscutting concern specifies that property constraints passed to operations should be constructed as XQuery expressions and should be encoded correctly. This crosscutting concern has influence on all operations which accept property constraints as input parameters. The specification of this crosscutting concern appears in several places in the INFOD base specification and is mixed with the specifications of the influenced

operations. It is therefore classified as an XC3 crosscutting concern. We have noticed that all scattered specification parts of this crosscutting concern are identical and this crosscutting concern can be easily transformed to a XC1 crosscutting concern – specified once in its own section with a complete list of all operations to which it applies. This sample crosscutting concern shows that highly homogeneously scattered XC3 crosscutting concerns can be collapsed to XC1 crosscutting concerns and XC1 crosscutting concerns can be expanded to highly homogeneously scattered XC3 crosscutting concerns.

- **Crosscutting Concern X404**

"When used, the registry MUST notify the external participant about changes relevant in the registry." Lines 242 – 243, 350 – 351, 465 – 466, 541 – 542, 656 – 657, 730 – 731 in Appendix B

"The notification to an external participant is conditional on the information in its entry." Lines 1613, 1660, 1693 in Appendix B

This crosscutting concern specifies that notification to external participants is conditional and is dependent on a parameter they set when they register themselves to the INFOD registry. This crosscutting concern has influence on the creation and modification operations the INFOD registry provides to external participants in that these operations must accommodate parameters to indicate if external participants want to be notified or not. This crosscutting concern also has influence on components inside the INFOD

registry that are responsible for monitoring changes inside the INFOD registry and sending notifications to external participants in that if an external participant does not want to be notified, these components will not generate and send notifications. However, we do not know what components inside the INFOD registry are influenced because the specification of this crosscutting concern does not specify what changes are relevant to what types of external participants. This crosscutting concern is therefore classified as an XC4 crosscutting concern. If this uncertainty is resolved in a future version of the INFOD base specification, this crosscutting concern can become an XC3 crosscutting concern.

We can see in the INFOD base specification that the specification parts of X404 regarding creation and modification operations and the specification parts of X404 regarding notifications are expressed differently and therefore these parts of X404 can be seen as two different XC3 crosscutting concerns – one specifying that creation and modification operations of external participants must have a parameter to indicate if external participants want to be notified about related changes, the other specifying that notifications to external participants are conditional. However, when this crosscutting concern is seen as two distinct crosscutting concerns, the intrinsic link between them – on what condition an external participant will get notification – is missing. Therefore, this crosscutting concern is a heterogeneously scattered XC4 crosscutting concern and can not be seen as two.

5.4 Characteristics of Crosscutting Concerns in Requirements

As a result of our experiment, we have proposed some characteristics of crosscutting concerns in requirement specifications. These characteristics make it clear that crosscutting concerns in requirements are quite different from crosscutting concerns in code, as will be discussed below. One implication of these characteristics is that identifying crosscutting concerns from requirement specifications turns out to be a labor intensive and error-prone task. These characteristics arise from the following areas:

1. Crosscutting concerns in requirement specifications can be specified in either a scattered or a localized manner.

2. Scattered specification parts of crosscutting concerns in requirement specifications can be either homogeneous or heterogeneous.

3. Tangling between concerns in requirement specifications can be expressed either explicitly or implicitly.

4. In requirement specifications, dominant decompositions at different levels of abstraction can be different.

5. In requirement specifications, crosscutting concerns can be expressed in a format other than plain text.

6. The influence of crosscutting concerns in requirements is much broader than that of crosscutting concerns in code. This influence must be studied carefully.

Each of these will be discussed in turn in the following subsections.

5.4.1 Scattered or Localized Specification

Scattering is about the representation of crosscutting concerns rather than the crosscutting concerns themselves. It means that the representation of a crosscutting concern is spread out rather than localized in an artifact [3]. In code, the representation of a crosscutting concern is the set of code pieces that implement this concern. The code pieces of a crosscutting concern are spread over multiple code modules, e.g. functions and classes, instead of being well modularized within one code module. In requirement specifications, this phenomenon is completely different. The representations of crosscutting concerns in requirement specifications – the specification text of these concerns – can be either spread over several requirement artifacts or be well localized. Among the crosscutting concerns we have identified from the INFOD base specification, the specification text of XC1 crosscutting concerns and XC2 crosscutting concern is localized while the specification text of XC3 and XC4 crosscutting concerns is scattered.

This characteristic of crosscutting concerns in requirement specifications has great impact on the identification of crosscutting concerns from requirement specifications. The tools-

assisted automatic or semi-automatic AORE approaches such as the approaches proposed in [37] may not work well with the crosscutting concerns specified in a localized manner as we have observed in the INFOD base specification because these approaches use information retrieval techniques that search for terms that appear repeatedly, but locally specified crosscutting concerns do not appear more than once in a requirement specification. Manual inspection must be involved to find these crosscutting concerns. Not surprisingly, manually identifying crosscutting concerns from a requirement specification is a labor-intensive and error-prone task. Also, manual identification is subjective. Requirement engineers with different background or different understanding may produce different sets of crosscutting concerns.

5.4.2 Homogeneously and Heterogeneously Scattered Specification

In a requirement specification, the scattered specification parts of an XC3/XC4 crosscutting concern can be either identical or analogous to each other or largely different from each other. In the former case we say this crosscutting concern is homogeneously scattered and in the latter case we say this crosscutting concern is heterogeneously scattered. For a heterogeneously scattered crosscutting requirement concern, its scattered parts may look different, and may even be recognized as different concerns. However, there is some intrinsic or conceptual (there is an underlying concept that provides the link) link between these parts. For example, consider a concern that addresses collaboration between two components. This concern is specified separately in the specifications of the two components in a scattered manner. The two scattered parts of

this concern may look different, and thus may be recognized as two distinct concerns. However, if the scattered part in one component is modified, the part in the other component needs to be modified accordingly; otherwise the collaboration between two components becomes inconsistent. In the INFOD base specification, crosscutting concern X405 – "The message MUST be structured according to the WS-Base Faults specification" – is one of the homogeneously scattered crosscutting concerns. It is specified repeatedly in several places throughout the INFOD base specification in exactly the same manner. On the other hand, crosscutting concern X304 is specified from line 99 – 100 as "each entry has a name and description, both of which are optional, not necessarily unique and have string values" and is specified at line 133 as "Data Source Entries, like other entries, have an optional name and description". The two scattered parts of this crosscutting concern can be seen as two distinct concerns. This crosscutting concern is therefore considered to be heterogeneously scattered.

Homogeneously scattered crosscutting concerns can be identified by searching for similar terms or similar descriptions in a requirement specification. Heterogeneously scattered crosscutting concerns are likely to be more difficult to find, and maybe impossible to find by mechanical search no matter what searching technique is applied. The reason is that the scattered specification parts of a crosscutting concern can be expressed differently, although they have similar meanings or they refer to a concept that implies intrinsic linkage.

Keyword matching might be useful to initiate searches for this kind of crosscutting concerns. In the above example, the keywords “entry” and “name” provide some important clues to identify X304. The challenge is to know what sets of keywords to search for, and how to interpret the search results. To what degree keyword matching can help to identify heterogeneously scattered crosscutting concerns is unknown until quantitative data regarding this is collected.

5.4.3 Explicit or Implicit Tangling

In code, tangling between concerns is always explicit because code pieces of different concerns are directly intermixed together [3]. In requirement specifications, tangling between concerns can be either explicit or implicit. There are two types of explicit tangling. The first one is that a crosscutting concern is scattered and its scattered specification parts are mixed inline with specifications of the concerns it crosscuts. The second one is that a crosscutting concern is localized and its specification has explicit references to the concerns it crosscuts. The scope of an explicitly tangled crosscutting concern can be known by studying this crosscutting concern directly – studying all its scattered specification parts if it is scattered or studying its references to other concerns if it is localized. Implicit tangling, on the other hand, is that the specification of a crosscutting concern is not mixed with specifications of the concerns it crosscuts and its specification does not have explicit references to concerns it crosscuts. The scope of an implicitly tangled crosscutting concern can not be revealed without exhaustive searches

through the entire requirement document [6], and sometimes can not be known at the requirement stage.

In the INFOD base specification, XC1 crosscutting concerns are explicitly tangled because they have explicit references to the concerns they crosscut. Which concerns they crosscut can be known from their specifications. XC3 crosscutting concerns are also explicitly tangled because their scattered specifications either explicitly reference the concerns they crosscut or are intermixed with the specifications of the concerns they crosscut. By studying all the places they appear, which concerns they crosscut can be inferred. XC2 crosscutting concerns, on the other hand, are implicitly tangled. This is because specifications of XC2 crosscutting concerns do not mix with specification of the concerns they crosscut and do not have explicit references to the concerns they crosscut. For example, consider the following XC2 crosscutting concern identified in the INFOD base specification:

"INFOD uses existing security mechanisms to ensure that the dissemination of information happens according to security policies." (X206)

This crosscutting concern has a system wide influence. There is no surprise that a great number of concerns regarding information dissemination are constrained by this crosscutting concern. However, we cannot know exactly which concerns are constrained. We can wait until the INFOD base specification is more fully developed with all

information dissemination relevant concerns explicitly specified, or postpone reasoning about this crosscutting concern to later development stages. This crosscutting concern is therefore considered to be implicitly tangled. XC4 crosscutting concerns are also considered to be implicitly tangled because not all of their influenced concerns are explicitly specified. The only example of XC4 crosscutting concerns we have found in the INFOD base specification is X404.

5.4.4 Varied Dominant Decompositions at Each Level of Abstraction

Crosscutting concerns are the result of a chosen dominant decomposition of a system [6]. As we have discussed early on in this chapter, the requirement specification of a system can have different dominant decompositions at different levels of abstraction and thus have different sets of crosscutting concerns. For example, at a higher level of abstraction of a requirement specification, the specification can be organized with respect to features and crosscutting concerns can be business rules that limit the provision of some features. On the other hand at a more detailed level of abstraction of a requirement specification, the specification is structured with respect to groups of functions and crosscutting concerns can be common behaviors that some functions perform. One prerequisite for identifying different sets of crosscutting concerns at different levels of abstraction of a requirement specification is to figure out the boundaries between levels of abstraction. This is necessarily a manual job. Also, demarcating levels of abstraction in a requirement specification is not a simple black and white question. Each requirement engineer may

have his or her own interpretation. This will make identifying crosscutting concerns from requirement specifications hard and possibly even unproductive.

5.4.5 Rich Format of Requirement Specification

In code, crosscutting concerns can only manifest themselves as code pieces. However, we have noticed in our experiment that crosscutting concerns in requirement specifications can be expressed in formats other than plain text. In the INFOD base specification, crosscutting concerns such as X201 and X205 are specified as figures depicting the relationships of elements in INFOD system.

As far as we know, no work in the AORE community addresses identifying crosscutting concerns from media other than plain text. Text searching and analyzing approaches can not be applied to non-textual crosscutting concerns. Even if these figures, for example, are accompanied with description text, the information the text provides is much less than the information the figures express. The crosscutting nature of these figures may not be entirely clear from analyzing their accompanying descriptive text.

5.4.6 Broader and More Obscure Influence on Software Systems

Crosscutting concerns in requirements have much broader influence on software systems than crosscutting concerns in code. Crosscutting concerns manifest in requirements can propagate to the architecture stage and become architectural crosscutting concerns. These

crosscutting concerns thus have influence on architectural elements of a software system [6]. Crosscutting concerns in requirements sometimes can eventually manifest in code and thus have influence on implementation units of a software system. Moreover, crosscutting concerns in requirements can have influence on decision making in later development activities. Let us recap the crosscutting concern X203 that the message delivery from publishers to consumers uses a notification system similar to WS-Notification. This crosscutting concern influences the decision as to which communication mechanism is adopted between publishers and consumers.

Crosscutting concerns in code take effect on a software system only where their code fragments appear. Their influence on a software system is explicit and can be obtained by means of studying the entire source code of a software system, albeit at the cost of the effort of doing so. On the contrary, the influence of crosscutting concerns in requirements presents in a more implicit and more obscure manner [37]. For some crosscutting concerns in requirements, their influence on a software system can not be obtained until the requirements are fully developed and all the requirement documents are studied thoroughly over several rounds. And for some other crosscutting concerns in requirements, their influence on a software system can not be completely understood in the requirement engineering stage. Knowing their influence has to be postponed to later development stages.

5.5 Summary

In this chapter, we introduced the five levels of abstraction – the Vision Level, the Concept Level, the Architectural Level, the Operational Level and the Detailed Level – which we found in the INFOD base specification. We also discussed the different dominant decomposition structures of the INFOD base specification at these five levels of abstraction. Then, we proposed our classification of crosscutting concerns found in the INFOD base specification with respect to their crosscutting representation and discussed three categories of crosscutting concerns identified in the specification. Following this, we gave an in-depth analysis of sample crosscutting concerns identified in the INFOD base specification. We also proposed and applied a set of characteristics of crosscutting concerns in requirement specifications, based on our case study, to the INFOD base specification.

Chapter 6

Conclusions and Future Work

In this chapter, conclusions regarding identifying crosscutting concerns in requirement specifications will be drawn from our experiment. Then, limitations of our experiment as well as possible future work will be pointed out.

6.1 Conclusions

This thesis has contributed to furthering the understanding of crosscutting concerns in requirement documents, in particular how crosscutting concerns manifest themselves in software requirement specifications. Specially, this thesis has examined whether identifying crosscutting concerns from requirement documents may be an unproductive development practice. The conclusions of our experiment come from the following three observations:

1. Great difference exists between crosscutting concerns in requirements and crosscutting concerns in code.

2. Identifying crosscutting concerns from requirement documents is resource intensive and it requires more efforts than it is worth.

3. It is hard to predict crosscutting concerns in design and in code from crosscutting concerns in requirements.

4. Requirement documents may have multiple levels of abstraction. Dominant decomposition structures at each level can be different and concerns of interest at each level can be different as well. So, there may be different sets of crosscutting concerns at each level of abstraction.

We will discuss each of these observations in the following sections.

6.1.1 Differences

We have observed from our experiment that there is great difference between crosscutting concerns in requirements and crosscutting concerns in code.

First of all, how crosscutting concerns manifest in code and how crosscutting concerns present in requirements is different. Crosscutting concerns in code are scattered, which means code pieces implementing a crosscutting concern appear in multiple different code modules rather than being well encapsulated in one module. Crosscutting concerns in requirements, on the other hand, can be specified either in a scattered manner or in a localized manner. For example, as we have discussed in Chapter 5, the crosscutting concern X202 in the INFOD base specification is specified in a localized manner while the crosscutting concern X403 is specified in a scattered manner.

Secondly, crosscutting concerns in requirements also differ from crosscutting concerns in code in the way they influence the rest of the system. Crosscutting concerns in code influence the rest of the system in an explicit manner. They take effect only at where their code pieces appear. Crosscutting concerns in requirements, on the other hand, influence the system in a more implicit manner. They can have more far reaching influence than they seem to have on the surface. In requirements, locally specified crosscutting concerns can have influence on other concerns even if those concerns are not referenced. Locally specified crosscutting concerns can even have influence on the entire system (e.g. the crosscutting concern X206 in the INFOD base specification) and their system wide influence sometimes can not be fully understood at the requirement engineering stage.

Thirdly, crosscutting concerns in code must be presented as code pieces but crosscutting concerns in requirements can be presented in a non-textual format such as in a graphic format. In our experiment, crosscutting concerns X201, X205 and X311 are presented as figures.

In conclusion, crosscutting concerns in requirements and crosscutting concerns in code present differently and affect the system differently. Techniques used to identify and manage crosscutting concerns in requirements and in code therefore must be different. For example, searching for repeatedly presented code patterns can be efficient to identify crosscutting concerns in code [8]. However, similar techniques may be limited, or even

fail, on addressing crosscutting concerns in requirements, especially when crosscutting concerns in requirements are specified in localized and implicit ways.

6.1.2 Resources

In our experiment, we have observed that identifying crosscutting concerns from software requirement specifications is an intensively human-involved, laborious, and error-prone job. This is especially the case if a software requirement specification is developed without any consideration of crosscutting concerns. The software requirement specification has to be studied in several rounds with great care in order that a comprehensive understanding of the requirement specification can be acquired and thus a relatively complete set of crosscutting concerns can be obtained.

Keyword matching techniques, such as the Information-Retrieval (IR) based techniques proposed by L. Rosenhainer [37], are proposed to automate the job of identifying crosscutting concerns. However, such techniques are often inefficient and may not save much effort. This is because providing effective keywords for identifying crosscutting concerns itself is a resource intensive task. Developers have to read all requirement documents to input keywords. This can be time consuming and sometimes unrealistic for complex and unstructured requirement documents [39]. Although domain knowledge may help to recall keywords for some typical crosscutting concerns (e.g. transaction and authentication for banking systems), there are many atypical crosscutting concerns which are specific to particular requirement documents [6]. Providing keywords for atypical

crosscutting concerns requires a huge amount of work of studying requirement documents.

Moreover, due to the nature of human involvement, the result of crosscutting concerns identification is more or less biased by requirement engineers' subjective judgment, which varies with respect to different knowledge backgrounds, different level of experience, etc. It is often hard to reach consensus whether or not a concern is crosscutting. This situation may become even worse if software requirement specifications being studied are written in some informal and unstructured manner like the INFOD base specification.

6.1.3 Prediction

Our experiment shows that it would be imprudent to predict crosscutting concerns in later development stages from crosscutting concerns identified in requirement documents. This is because crosscutting is a structural relationship between concerns [3]. Discussing crosscutting concerns without a chosen dominant decomposition structure does not make any sense. The dominant decomposition structure of a system at the requirement engineering stage is in most cases different from dominant decomposition structures in later development stages such as the implementation stage – the requirements of a system are likely structured with respect to features and functions, while the code of a system often organized as classes or procedures. Different dominant decompositions lead to different crosscutting concerns. For example, fault-tolerance can be a crosscutting

concern in requirements because it can cross over several functional requirements. This crosscutting concern may or may not result in code level crosscutting concerns, depending on how the software system is structured at the implementation stage. In addition, some code level crosscutting concerns are implementation-specific and can not be traced back to requirements. Logging is a typical implementation-specific crosscutting concern that is not raised in requirements in most cases. As a common development practice, logging code is inserted in programs where exception situations are likely to happen so as to provide hints to locate errors in code.

6.1.4 Levels of Abstraction

In code, there is one set of crosscutting concerns because a program has just one dominant decomposition structure [21]. In requirement documents however, there can be multiple sets of crosscutting concerns since there can be multiple decomposition structures, one per level of abstraction. We have found in our experiment that there are three sets of crosscutting concerns in the INFOD base specification – one at the Architectural Level, one at the Operational Level and one at the Detailed Level. This is because there are multiple levels of abstraction in the INFOD base specification. As we have discussed in Chapter 5, the dominant decomposition structures at each level of abstraction are different and the concerns of interest at each level of abstraction are different as well. As a result, the sets of crosscutting concerns identified at each level of abstraction are different.

There can be a mapping between crosscutting concerns identified at different levels of abstraction in that crosscutting concerns at a more detailed level of abstraction can be derived from, or be a concrete instance of, a crosscutting concern at a higher level of abstraction. For example, X206 is a crosscutting concern at the Architectural Level and it specifies that a security mechanism should be applied to INFOD so that information dissemination can happen in a secure manner. X313 at the Detailed Level defines some secured contexts within an INFOD system and X314 at the Detailed Level specifies an authentication model in an INFOD system. These two crosscutting concerns are security related and can be seen as two concrete instances of X206.

A crosscutting concern at a higher level of abstraction may not propagate to more detailed levels of abstraction. For example, X203 is a crosscutting concern at the Architectural Level. It specifies that communication between publishers and consumers shall be made via a WS-Notification like mechanism. This crosscutting concern does not propagate to the Operation Level and the Detailed Level because the INFOD base specification at these two levels focuses on the INFOD registry, rather than the external participants outside the INFOD registry.

Also, a crosscutting concern at a more detailed level of abstraction may not trace back to a crosscutting concern at a higher level of abstraction. This is because this crosscutting

concern is only of interest at the level of abstraction it appears. For example, X420 is a crosscutting concern at the Detailed Level and it specifies that faults generated in an INFOD system must be compliant with WS-Base Fault specification. This crosscutting concern does not trace back to any crosscutting concern at a higher level of abstraction because fault handling is of interest only at the Detailed Level.

The existence of multiple levels of abstraction and multiple sets of crosscutting concerns makes identifying crosscutting concerns from requirement documents even harder. Requirement developers need to figure out the boundaries between levels of abstraction in a requirement document. Then for each level of abstraction, they need to figure out the decomposition structure at this level of abstraction. Not surprisingly, these are resource-intensive and error-prone tasks. Even after levels of abstraction are figured out and sets of crosscutting concerns at each level of abstraction are identified, matching these crosscutting concerns to form a whole picture of crosscutting concerns in a system is not an easy task – the entire requirement document and all identified crosscutting concerns have to be studied carefully in several rounds.

6.2 Limitations and Future Work

This thesis is based on our study of the INFOD base specification. We have noticed in our study that crosscutting concerns that are often referenced in AORE literature, such as security [29], transaction management [6] and correctness [36], are not well presented in

the INFOD base specification. We believe that this is because the INFOD base specification is still in an immature state and its developers have spent most of their efforts on eliciting and capturing functional requirements of the INFOD system, rather than the quality attributes of the INFOD system. The absence of these typical crosscutting concerns weakens our results and limits the application of our results.

The characteristics of crosscutting concerns in requirements are generalized from our study of crosscutting concerns in the INFOD base specification. We believe that there might be other characteristics of crosscutting concerns that do not present in the INFOD base specification but present in other requirement documents. In addition, the INFOD base specification is very informal and unstructured with some errors and inconsistencies. These factors undoubtedly affect the results of our experiment.

In the future, studying fully developed requirement documents with both functional requirements and quality attributes presented will provide important complements to our experiment. Studying semi-formal, well structured and more disciplined requirement documents are also helpful in verifying the results of our experiment.

We suspect that changing requirements may cause a huge amount of rework on the previously identified crosscutting concerns. As requirement documents are refined, elaborated and modified, new crosscutting concerns may appear; implicitly specified crosscutting concerns may become explicit; and localized crosscutting concerns may

become scattered or vice versa. Therefore, requirement documents need to be re-analyzed. However, we can not make any conclusion on this since we can not find any significant revision of the INFOD base specification. Future work of examining the impact of requirement evolution on identified crosscutting concerns is important.

Other important future work will be investigating how requirement documents should be built so that crosscutting concerns can be localized and their influence over other concerns can be specified explicitly. Instead of identifying crosscutting concerns from existing software requirement specifications, crosscutting concerns can be addressed upfront when requirements are being compiled and software requirement specifications are being formed. Non-crosscutting concerns and crosscutting concerns are probably best developed simultaneously in a parallel and iterative process. Non-crosscutting concerns originate from an initial set of functional concerns and crosscutting concerns originate from an initial set of quality attributes. Both sets of concerns are refined. Inter- and intra-set interactions of concerns are analyzed. For a crosscutting concern, it can be documented in the manner proposed in [6] – being specified in a modularized place accompanied with a complete list of concerns it cuts across. As higher level and more abstract concerns are refined to more detailed level and more concrete concerns, some non-crosscutting concerns may become crosscutting and are moved to the crosscutting concern set because they are found to be crosscutting. By doing so, a relatively complete picture of crosscutting concerns in a software system can be drawn. Tools support for the building of such requirement specifications is also worthy of investigation.

In addition, although hundreds of papers regarding AORE have been published, few of them address handling crosscutting concerns in Agile software development. Compared with highly disciplined development methods such as Waterfall Model [38], Agile methods emphasize time-boxed iterative and evolutionary development, adaptive planning, incremental and evolutionary delivery, and rapid and flexible response to changes [26]. In an Agile project, requirements are not completely developed upfront. Instead, they are developed in an incremental and evolutionary manner through development iterations. Whether or not the proposed AORE approaches are applicable to Agile projects is worthy of investigation.

6.3 Closing Words

In our experiment, we have seen that:

1. Crosscutting concerns in requirements differ greatly from crosscutting concerns in code and have distinct characteristics.

2. Because of the distinct characteristics of crosscutting concerns in requirements, techniques used for identifying crosscutting concerns in code are not immediately applicable to discover crosscutting concerns in requirements. Also since dominant decomposition structures of a system vary from one development stage to other

development stages, the sets of crosscutting concerns can be different. It does not make sense to predict crosscutting concerns in a later development stage from crosscutting concerns identified in requirements without considering the dominant decomposition structures adopted at this development stage.

3. Identifying crosscutting concerns from existing requirement documents is a manual, labor intensive, time consuming and error-prone task. Moreover, some crosscutting concerns can not be completely understood and thus be managed in the requirement stage. We believe that it will be better to manage crosscutting concerns upfront when requirements are elicited and compiled, rather than mining from existing requirement documents.

In conclusion, identifying crosscutting concerns in requirements is a resource intensive and time consuming task. The cost of identifying crosscutting concerns in requirements, in particular requirement specifications, is very likely to outweigh the benefits it brings about such as improving the comprehensibility and maintainability of requirement documents, promoting the traceability of concerns and providing insights to downstream development activities. We believe that identification of crosscutting concerns from requirement specifications requires more work than it is worth.

6.4 Summary

In this chapter, we drew conclusions that crosscutting concerns in requirements differ greatly from crosscutting concerns in code and identifying crosscutting concerns from requirement documents that are built without awareness of crosscutting concerns can be unproductive. Then, we briefly summarized the limit of our experiment and pointed out some possible future directions as the end of this thesis. Finally closing words of this thesis were given.

REFERENCES

[1] J. Araújo, E. Baniassad, P. Clements, A. Moreira, A. Rashid, and B. Tekinerdoğan, *Early Aspects: The Current Landscape*, Technical Notes, CMU/SEI and Lancaster University, 2005

[2] *Aspect C++ Home Page*, http://www.aspectc.org/, last accessed in July 2009

[3] *Aspect-Oriented Software Development Conferences Website*, http://www.aosd.net/, last accessed in July 2009

[4] *AspectJ Home Page*, http://www.eclipse.org/aspectj/, last accessed in July 2009

[5] E. Baniassad, S. Clarke, *Theme: An Approach for Aspect-Oriented Analysis and Design*, Proceedings of the 26th International Conference on Software Engineering, pp.158-167, Edinburgh, UK, May 23-28, 2004

[6] E. Baniassad, P. Clements, J. Araújo, A. Moreira, A. Rashid, B. Tekinerdoğan, *Discovering Early Aspects*, Software, IEEE, v.23 n.1, pp.61-70, January 2006

[7] I. Brito, A. Moreira. *Towards A Composition Process For Aspect-Oriented Requirements*, Proceedings of the Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, Boston, MA, March 17-21, 2003

[8] M. Bruntink, A. Deursen, R. Engelen, T. Tourwe, *On the Use of Clone Detection for Identifying Crosscutting Concern Code*, IEEE Transactions on Software Engineering, v.31 n.10, pp.804-818, October 2005

[9] P. Clements, F. Bachman, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, J. Stafford, *Documenting Software Architectures, Views and Beyond*, Addison Wesley, 2003

[10] S. Davey, V. Dialani, R. Fehling, S. Fisher, D. Gawlick, C. Kantarjiev, C. Madsen, S. Malaika, S. Mishra, M. Shankar, *Information Dissemination in the Grid Environment – Base Specifications*, revised on 5 May 2007, http://www.ogf.org/documents/GFD.110.pdf

[11] S. Davey, V. Dialani, R. Fehling, S. Fisher, D. Gawlick, C. Kantarjiev, C. Madsen, S. Malaika, S. Mishra, M. Shankar, *Information Dissemination in the Grid Environment – Base Specifications – V2 (draft)*, revised on 3 Jan 2008, http://forge.gridforum.org/sf/go/doc14987?nav=1

[12] *Early Aspects Website*, http://www.early-aspects.net/, last accessed in July 2009

[13] F. Hayes-Roth, *Model-Based Communication Networks and VIRT: Orders of Magnitude Better for Information Superiority*, Proceedings of the Military Communications Conference, Washington, 2006

[14] *IEEE Std. 830-1998, Recommended Practice for Software Requirements Specifications*, IEEE Press, Piscataway, NJ, 1998

[15] *INFOD-WG Project Website*, https://forge.gridforum.org/projects/infod-wg/, last accessed in July 2009

[16] I. Jacobson, *Object Oriented Software Engineering: A Use Case Driven Approach*, ACP Press, 1992

[17] I. Jacobson, PW. Ng, *Aspect-Oriented Software Development with Use Cases*, Addison Wesley Professional, 2005

[18] *Java Message Service Specification*, Sun Microsystems, Palo Alto, CA, 2001

[19] *JBoss AOP Project Website*, http://www.jboss.org/jbossaop, last accessed in July 2009

[20] M. Katara, S. Katz, *Architectural Views of Aspects*, Proceedings of the 2nd International Conference on Aspect-oriented Software Development, pp.1-10, Boston, MA, March 17-21, 2003

[21] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, J. Irwin, *Aspect-Oriented Programming*, Proceedings of the European Conference on Object-Oriented Programming (ECOOP97), v.1241, pp.220-242, Jyväskylä, Finland, June 9-13, 1997

[22] P. Kruchten, *The 4+1 View Model of Architecture*, IEEE Software, v.12 n.6, pp.42-50, November 1995

[23] P. Kruchten, *The Rational Unified Process: An Introduction 3rd Edition*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2003

[24] R. Laddad, *AspectJ in Action: Practical Aspect-Oriented Programming*, Manning Publications Co., Greenwich, CT, 2003

[25] C. Larman, *Agile and Iterative Development: A Manager's Guide*, Addison-Wesley, 2004

[26] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development 3rd Edition*, Prentice-Hall, 2004

[27] *Merriam-Webster's Online Dictionary*, http://www.merriam-webster.com/, last accessed in July 2009

[28] A. Moreira, J. Araújo, I. Brito, *Crosscutting Quality Attributes For Requirements Engineering*, Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02), ACM, v.27, pp.167-174, Ischia, Italy, July 15-19, 2002

[29] N. Niu, S. Easterbrook, Y. Yu, *A Taxonomy of Asymmetric Requirements Aspects*, Proceedings of the 10th International Workshop on Early Aspects, LNCS 4765, pp.1-18, Vancouver, Canada, March 12-16, 2007

[30] B. Nuseibeh, *Weaving Together Requirements and Architectures*, Computer, IEEE, v.34 n.3, pp.115-117, March 2001

[31] *Open Grid Forum Website*, http://www.ogf.org/, last accessed in July 2009

[32] D. Parnas, *On the Criteria to Be Used in Decomposing Systems into Modules*, Communications of the ACM, v.15 n.12, pp.1053-1058, 1972

[33] J. Pérez, N. Ali, J. Carsí, I. Ramos, *Designing Software Architectures with an Aspect-Oriented Architecture Description Language*, Proceedings of the 9th International Symposium on Component-Based Software Engineering, LNCS4063, pp.123-138, Västeras, Sweden, June 27-July 1, 2006

[34] P. Tarr, W. Harrison, H. Ossher, A. Finkelstein, B. Nuseibeh, D. Perry, *Workshop on Multi-Dimensional Separation of Concerns in Software Engineering*, ACM SIGSOFT, Software Engineering Notes, v.26 n.1, pp.78-81, 2001

[35] A. Rashid, P. Sawyer, A. Moreira, J. Araújo, *Early Aspects: A Model for Aspect-Oriented Requirements Engineering*, Proceedings of the 10th IEEE Joint International Conference on Requirements Engineering, pp.199-202, Essen, Germany, September 9-13, 2002

[36] A. Rashid, A. Moreira, J. Araújo, *Modularization and Composition of Aspectual Requirements*, Proceedings of the 2nd International Conference on Aspect-Oriented Software Development, ACM Press, pp.11-20, Boston, MA, March 17-21, 2003

[37] L. Rosenhainer, *Identifying Crosscutting Concerns in Requirements Specifications*, Proceedings of the Aspect-Oriented Requirements Engineering and Architecture Design Workshop, Vancouver, Canada, October 24-28, 2004

[38] W. Royce, *Managing the Development of Large Software Systems: Concepts and Techniques*, Proceedings of the 9th international Conference on Software Engineering, International Conference on Software Engineering, IEEE Computer Society Press, pp.328-338, Monterey, CA, March 30-April 2, 1987

[39] A. Sampaio, N. Loughran, A. Rashid, P. Rayson, *Mining Aspects in Requirements*, Aspect-Oriented Requirements Engineering and Architecture Design Workshop, AOSD 2005, Chicago, IL, March 14-18, 2005

[40] J. Sillito, C. Dutchyn, A. Eisenberg, K. DeVolder, *Use Case Level Pointcuts*, Proceedings of the 18th European Conference on Object-Oriented Programming (ECOOP'04), Oslo, Norway, June 14-18, 2004

[41] I. Sommerville, P. Sawyer, *Viewpoints: Principles, Problems and A Practical Approach to Requirements Engineering*, Annals of Software Engineering, v.3 n.1, pp.101-130, 1997

[42] B. Tekinerdogan, *ASAAM: Aspectual Software Architecture Analysis Method*, Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture (WICSA'04), IEEE CS Press, pp.5-14, Oslo, Norway, June 12-15, 2004

[43] K. Wiegers, *Software Requirements 2nd Edition*, Microsoft Press, 2003

[44] Y. Yu, J. Leite, J. Mylopoulos, *From Goals to Aspects: Discovering Aspects from Requirements Goal Models*, Proceedings of the Requirements Engineering Conference, 12th IEEE International (RE'04), pp.38-47, Kyoto, Japan, September 6-10, 2004

Appendix A

Crosscutting Concerns in the INFOD Base Specification

Crosscutting Concerns at the Architectural Level

- **X201**

X201 – Relationships of types of resources in the INFOD registry	
Description	
Figure 1 mainly defines the relationships of different types of resources in the INFOD registry. This concern has great influence on the definitions of types of resources in the INFOD registry.	
Where It Appears	
Figure 1	
Concerns Influenced	
Types of resources in the INFOD registry such as subscriptions and property vocabulary instances.	
Classification based on Crosscutting Representations	**XC1**
This crosscutting concern is specified in a localized manner with all influenced concerns specified explicitly.	

- **X202**

X202 – INFOD registry notifies publishers
Description
The INFOD registry notifies publishers about customers and content of information. This crosscutting concern has influence on the INFOD registry and publishers in that the INFOD registry need to send out notifications to publishers in certain forms and publishers also need to know how to process these notifications.
Where It Appears
Line 48

Concerns Influenced	
The INFOD registry and publishers.	
Classification based on Crosscutting Representations	**XC1**
This crosscutting concern is specified in a localized manner with all influenced concerns specified explicitly.	

- **X203**

X203 – Publishers use WS-Notification like mechanism to send messages	
Description	
Publishers use a WS-Notification like mechanism to send messages. This crosscutting concerns influences the behaviors of publishers and consumers because both sides of the message delivery must agree on a WS-Notification like notification mechanism.	
Where It Appears	
Line 51 – 52	
Concerns Influenced	
Publishers and consumers.	
Classification based on Crosscutting Representations	**XC1**
This crosscutting concern is specified in a localized manner with all influenced concerns specified explicitly.	

- **X204**

X204 – Each entry can have property constraints
Description
Each entry can have property constraints with reference to the properties of other types of entries. This crosscutting concern has influence on the definition of each type of entry.
Where It Appears
Line 55 – 56
Concerns Influenced

Definitions of types of entries including Publisher Entry (PE), Consumer Entry (CE), Subscriber Entry (SE) and Data Source Entry (DSE).
Classification based on Crosscutting Representations **XC1**
This crosscutting concern is specified in a localized manner. Although this crosscutting concern is specified with respect to a collective reference – "each entry", elements in this collection – PE, CE, SE and DSE – are known at this point. So the influenced concerns are explicitly specified.

- **X205**

X205 – Constraint relationships between types of entries	
Description	
Figure 2 specifies constraint relationships between different types of entries in the INFOD registry. This crosscutting concern has influence on the definition of the property constraint of each type of entry.	
Where It Appears	
Figure 2	
Concerns Influenced	
Definitions of property constraints of each type of entries e.g. DSE can not have property constraints to PE.	
Classification based on Crosscutting Representations	**XC1**
This crosscutting concern is specified in a localized manner with all influenced concerns specified explicitly.	

- **X206**

X206 – Security mechanism in INFOD
Description
Some existing security mechanism is applied to INFOD to ensure secure information dissemination. This crosscutting concern has broad influence on INFOD. The influence of this concern is so broadly spread that any other concerns in the specification may be affected. However, whether a given concern is affected by this concern or not can not be determined

immediately in the requirement stage.
Where It Appears
Line 175 – 176
Concerns Influenced
The influence of this crosscutting concern is too broad to know which concerns are influenced.
Classification based on Crosscutting Representations **XC2**
This crosscutting concern is specified in a localized manner. Its influence is so broad that which concerns it may impact can not be determined at the requirement stage.

- **X207**

X207 – Lifetime management in INFOD
Description
A mechanism need to be provided to keep extending the life period of resources, otherwise they will be destroyed automatically after a period of time. This crosscutting concern has system-wide influence on INFOD.
Where It Appears
Line 185 – 186
Concerns Influenced
The influence of this crosscutting concern is too broad to know which concerns are influenced.
Classification based on Crosscutting Representations **XC2**
This crosscutting concern is specified in a localized manner. It has broad influence and how it affects other concerns can not be determined at the requirement stage.

Crosscutting Concerns at Operational Level

- **X301**

X301 – Each resource type has calls to create and drop its instances	
Description	
Each resource type has calls to create and drop its instances from the INFOD registry. This crosscutting concern has influence on all types of resources in the INFOD registry.	
Where It Appears	
Line 94	
Concerns Influenced	
All types of resources in the INFOD registry.	
Classification based on Crosscutting Representations	**XC1**
This crosscutting concern is specified in a localized manner. All resource types in the INFOD registry are known at this point so they are considered being explicitly specified.	

- **X302**

X302 – Some resource type has a call to replace its instances	
Description	
Some resource type has a call to replace its instances from the INFOD registry. This crosscutting concern has influence on some types of resources in the INFOD registry.	
Where It Appears	
Line 95	
Concerns Influenced	
Some types of resources in the INFOD registry. These types of resources can not be enumerated immediately.	
Classification based on Crosscutting Representations	**XC2**
This crosscutting concern is specified in a localized manner. However, we don't know which resource type has such call until we know all types of resources appear.	

- **X303**

X303 – Each entry is identified by an EPR	
Description	
Each entry in the INFOD registry, such as PE, CE, SE and DSE, is uniquely identified by an EPR. This crosscutting concern has influence on the definitions of each type of entries.	
Where It Appears	
Line 98, Line 132	
Concerns Influenced	
Definitions of PE, CE, SE and DSE.	
Classification based on Crosscutting Representations	**XC3**
This crosscutting concern is specified in a scattered manner.	

- **X304**

X304 – Each entry has an optional name and description	
Description	
Each entry in the INFOD registry, such as PE, CE, SE and DSE, has an optional name and description. This crosscutting concern has influence on the definitions of each type of entries.	
Where It Appears	
Line 99 – 100, Line 133	
Concerns Influenced	
Definitions of PE, CE, SE and DSE.	
Classification based on Crosscutting Representations	**XC3**
This crosscutting concern is specified in a scattered manner.	

- **X305**

X305 – Creation operations of PE, CE and SE
Description
This crosscutting concern defines the common behavior of creation operations provided to PE,

CE and SE.
Where It Appears
Line 104 – 106
Concerns Influenced
Creation operations for PE, CE and SE.

Classification based on Crosscutting Representations	**XC1**

- **X306**

X306 – Replace operations of PE, CE and SE	
Description	
This crosscutting concern defines the common behavior of replace operations provided to PE, CE and SE.	
Where It Appears	
Line 107 – 111	
Concerns Influenced	
Replace operations for PE, CE and SE.	
Classification based on Crosscutting Representations	**XC1**
This crosscutting concern is specified in a localized manner with all influenced concerns explicitly specified.	

- **X307**

X307 – Removal operations of PE, CE and SE
Description
This crosscutting concern defines the common behavior of removal operations provided to PE, CE and SE.
Where It Appears
Line 112 – 118

Concerns Influenced	
Removal operations for PE, CE and SE.	
Classification based on Crosscutting Representations	**XC1**
This crosscutting concern is specified in a localized manner with all influenced concerns explicitly specified.	

- **X308**

X308 – No information flows until a subscription is created	
Description	
This crosscutting concern defines a pre-condition for information dissemination in INFOD. There is no information dissemination happens in INFOD until a subscription is created.	
Where It Appears	
Line 147	
Concerns Influenced	
Broad influence on INFOD.	
Classification based on Crosscutting Representations	**XC2**
This crosscutting concern is specified in a localized manner. Which concerns are influenced can not be determined immediately.	

- **X309**

X309 – Dynamic consumer constraint in subscriptions
Description
This crosscutting concern specifies that subscriptions can contain dynamic consumer constraints. It also specifies that publishers are responsible for evaluating dynamic consumer constraints. This concern has influence on the definition of subscription as well as the behavior of publishers.
Where It Appears
Line 156 – 157
Concerns Influenced

Subscription and Publishers.
Classification based on Crosscutting Representations **XC1**
This crosscutting concern is specified in a localized manner with its influenced concerns specified explicitly.

- **X310**

X310 – A basic dependency rule
Description
This crosscutting concern defines a basic dependency rule of types of resources managed in the INFOD registry. It constrains all creation, replace and drop operations.
Where It Appears
Line 161 – 162
Concerns Influenced
All operations of types of resources on the INFOD registry.
Classification based on Crosscutting Representations **XC1**
This crosscutting concern is specified in a localized manner with its influenced concerns specified explicitly.

- **X311**

X311 – Relation between INFOD resources
Description
Figure 5 dictates relations between types of resources in the INFOD registry. It has influence on all types of resources in the INFOD registry.
Where It Appears
Figure 5, Line 163 – 166
Concerns Influenced
Definitions of types of resources in the INFOD registry.
Classification based on Crosscutting Representations **XC1**
This crosscutting concern is specified in a localized manner with its influenced concerns

specified explicitly.

- **X312**

X312 – React to changes in the INFOD registry	
Description	
This concern specifies that INFOD objects, external participants of INFOD VIRT, need to react to changes in the INFOD registry. This concern influences the specification and behavior of all external participants, including publishers, consumers and subscribers.	
Where It Appears	
Line 172 – 173	
Concerns Influenced	
Behaviors of publishers, consumers and subscribers.	
Classification based on Crosscutting Representations	**XC1**
This crosscutting concern is specified in a localized manner with its influenced concerns specified explicitly.	

- **X313**

X313 – Creating security tokens
Description
This crosscutting concern specifies that to establish secured communication contexts between INFOD services the INFOD registry is responsible for creating security tokens for publishers, subscribers and consumers and publishers are responsible for creating security tokens for consumers. On the surface, this crosscutting concern affects the INFOD registry and publishers only. However, publishers, subscribers and consumers must be able to ask the INFOD registry for security tokens. Similarly, consumers should be able to ask publishers for security tokens.
Where It Appears
Line 1737 – 1743
Concerns Influenced
The INFOD registry, publishers, consumers and subscribers.

Classification based on Crosscutting Representations	**XC1**
This crosscutting concern is specified in a localized manner with its influenced concerns specified explicitly.	

- **X314**

X314 – Authentication model for accessing INFOD resources	
Description	
This crosscutting concern specifies that an authentication model needs to be applied to creation, deletion and other operations on service interfaces of types of resources in INFOD. This crosscutting concern affects service interfaces of types of resources.	
Where It Appears	
Line 1777 – 1782	
Concerns Influenced	
Services interfaces provided by the INFOD registry.	
Classification based on Crosscutting Representation	**XC1**
This crosscutting concern is specified in a localized manner. Because all service interfaces provided by the INFOD registry are known at the point, the influence of this crosscutting concern is considered as explicitly specified.	

Crosscutting Concerns at Detailed Level

- **X401**

X401 – Common actions of creating entry/resource in the INFOD registry	
Description	
This crosscutting concern specifies a series of common actions creation operations provided by the INFOD registry should perform.	
Where It Appears	
Line 203 – 204, 426 – 427, 617 – 618, 810 – 811, 1053 – 1054, 1151 – 1152, 1247 – 1248, 1355 – 1356	
Concerns Influenced	
Creation operations provided by all service interfaces of the INFOD registry.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of creation operations.	

- **X402**

X402 – Parameter WSReference follows the definition of WS-Addressing	
Description	
This crosscutting concern specifies that the parameter WSReference in the creation and replace operations of publisher entries, subscriber entries and consumer entries follows the definition of end point reference in the WS-Addressing specification.	
Where It Appears	
Line 224 – 226, 329 – 331, 447 – 449, 520 – 522, 638 – 640, 709 – 711	
Concerns Influenced	
Creation and replace operations of publisher entries, subscriber entries and consumer entries.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related creation and modification operations.	

- **X403**

X403 – Parameter PropertyConstraint in operations	
Description	
This crosscutting concern specifies that the parameter PropertyConstraint in operations needs to be formulated as an XQuery and needs to be encoded properly.	
Where It Appears	
Line 234 – 240, 342 – 348, 457 – 463, 533 – 539, 648 – 654, 722 – 728, 848 – 854, 937 – 941, 1387 – 1393	
Concerns Influenced	
The creation and replace operations of publisher entries, subscriber entries, consumer entries and subscriptions, and the creation operation of data source entries.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.	

- **X404**

X404 – Parameter Notification in creation and replace operations	
Description	
This crosscutting concern specifies that the parameter Notification in creation and replace operations of publisher entries, subscriber entries and consumer entries indicates if the corresponding external participants want to be notified by the INFOD registry.	
Where It Appears	
Line 242 – 243, 350 – 351, 465 – 466, 541 – 542, 656 – 657, 730 – 731, 1613, 1660, 1693	
Concerns Influenced	
The creation and replace operations of publisher entries, subscriber entries and consumer entries.	
Classification based on Crosscutting Representation	**XC3**

This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.

- **X405**

X405 – Response messages are structured according to the WS-Base Faults	
Description	
This crosscutting concern specifies that the response message of all operations of the INFOD registry must be structured according to the WS-Base Faults specification.	
Where It Appears	
Line 265 – 267, 374 – 375, 418 – 419, 488 – 489, 565 – 566, 609 – 610, 679 – 680, 754 – 755, 798 – 799, 886 – 887, 975 – 976, 1021 – 1022, 1098 – 1099, 1143 – 1144, 1198 – 1199, 1244 – 1245, 1301 – 1302, 1345 – 1346, 1416 – 1417, 1460 – 1461, 1513 – 1515	
Concerns Influenced	
All the operations provided by the INFOD registry.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.	

- **X406**

X406 – Create resource authentication fail
Description
This crosscutting concern defines a failed situation the operations to create resource in the INFOD registry must deal with.
Where It Appears
Line 261 – 262, 484 – 485, 675 – 676, 880 – 881, 1094 – 1095, 1192 – 1193, 1295 – 1296, 1410 – 1411
Concerns Influenced
All the operations that create resource in the INFOD registry.

Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.	

- **X407**

X407 – Missing required parameters fail	
Description	
This crosscutting concern defines a failed situation all operations provided by the INFOD registry must deal with.	
Where It Appears	
Line 263, 372, 416, 486, 563, 607, 677, 752, 796, 884, 973, 1019, 1096, 1141, 1196, 1242, 1299, 1343, 1414, 1458, 1511	
Concerns Influenced	
All operations provided by in the INFOD registry.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.	

- **X408**

X408 – Unsupported XQuery fail
Description
This crosscutting concern defines a failed situation where operations that take XQuery expressions as parameters must deal with.
Where It Appears
Line 264, 373, 487, 564, 678, 753, 885, 974, 1415, 1512
Concerns Influenced
The creation and replace operations of publisher entries, subscriber entries, consumer entries and subscriptions, the creation operation of data source entries and the GetMetaData operation.

Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.	

- **X409**

X409 – Replace resource authentication fail	
Description	
This crosscutting concern defines a failed situation that the operations to replace existing resource in the INFOD registry must deal with.	
Where It Appears	
Line 368 – 369, 559 – 560, 748 – 749, 969 – 970	
Concerns Influenced	
The replace operations of publisher entries, subscriber entries, consumer entries and subscriptions.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.	

- **X410**

X410 – Unknown resource reference fail
Description
This crosscutting concern defines a failed situation that the operations which take references to resources in the INFOD registry as parameters must deal with.
Where It Appears
Line 370 – 371, 414 – 415, 561 – 562, 605 – 606, 750 – 751, 794 – 795, 882 – 883, 971 – 972, 1139 – 1140, 1194 – 1195, 1240 – 1241, 1297 – 1298, 1341 – 1342, 1412 – 1413, 1456 – 1457
Concerns Influenced

The replace and drop operations of publisher entries, subscriber entries, consumer entries, data and subscriptions, the drop operations of property vocabulary, property vocabulary instance, data source entries and data vocabulary.
Classification based on Crosscutting Representation **XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.

- **X411**

X411 – Drop resource authentication fail	
Description	
This crosscutting concern defines a failed situation that the operations to drop resources from the INFOD registry must deal with.	
Where It Appears	
Line 412 – 413, 603 – 604, 792 – 793, 1015 – 1016, 1137 – 1138, 1238 – 1239, 1339 – 1340, 1454 – 1455	
Concerns Influenced	
All drop operations of types of resources in the INFOD registry.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.	

- **X412**

X412 – Execution mode fail
Description
This crosscutting concern defines a failed situation that all drop operations must deal with when applying the specified execution mode.
Where It Appears
Line 417, 608, 797, 1020, 1142, 1243, 1344, 1459

Concerns Influenced	
All drop operations of types of resources in the INFOD registry.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.	

- **X413**

X413 – Unsupported vocabulary language fail	
Description	
This crosscutting concern defines a failed situation where operations that takes vocabulary expression as parameters must deal with.	
Where It Appears	
Line 1097, 1197, 1300	
Concerns Influenced	
The creation operations of property vocabulary, property vocabulary instance and data vocabulary.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.	

- **X414**

X414 – Common actions of replacing entry/resource in the INFOD registry
Description
This crosscutting concern specifies a series of actions which are common to all replace operations provided by the INFOD registry.
Where It Appears
Line 303 – 306, 493 – 497, 682 – 686, 889 – 893
Concerns Influenced
Replace operations provided by all service interfaces of the INFOD registry.

Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of replace operations.	

- **X415**

X415 – Execution mode of removing entry/resource from the INFOD registry	
Description	
This crosscutting concern specifies a set of execution modes which are applied to all drop operations provided by the INFOD registry.	
Where It Appears	
Line 390 – 396, 581 – 587, 770 – 776, 992 – 999, 1115 – 1121, 1215 – 1221, 1317 – 1323, 1432 – 1438	
Concerns Influenced	
Drop operations provided by all service interfaces of the INFOD registry.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of drop operations.	

- **X417**

X417 – Publishers notify consumers using a WS-Notify like mechanism	
Description	
This crosscutting concern defines that publishers send notification to consumers using a WS-Notify like notification mechanism.	
Where It Appears	
Line 1522 – 1524	
Concerns Influenced	
Publishers and consumers.	
Classification based on Crosscutting Representation	**XC1**
This crosscutting concern is specified in a localized manner with its influence concerns	

specified explicitly.

- **X418**

X418 – Message headers of messages of operations	
Description	
This crosscutting concern specified that messages of all operations must construct message headers according to the WS-Addressing specification.	
Where It Appears	
Line 245 – 246, 353 – 354, 397 – 398, 468 – 469, 544 – 545, 588 – 589, 659 – 660, 733 – 734, 777 – 778, 864 – 865, 953 – 954, 1000 – 1001, 1077 – 1078, 1122 – 1123, 1173 – 1175, 1222 – 1223, 1278 – 1279, 1324 – 1325, 1394 – 1395, 1439 – 1440, 1497 – 1498	
Concerns Influenced	
All operations provided by the INFOD registry.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications of the related operations.	

- **X419**

X419 – Message headers of notifications in INFOD	
Description	
This crosscutting concern specified that messages of all notifications must construct message headers according to the WS-Addressing specification.	
Where It Appears	
Line 1564 – 1565, 1655 – 1656, 1688 – 1689, 1718 – 1719	
Concerns Influenced	
All notifications in INFOD.	
Classification based on Crosscutting Representation	**XC3**
This crosscutting concern is specified in a scattered manner and is mixed with the specifications	

of the related operations.

- **X420**

X420 – Faults must be complaint to WS-Base Fault	
Description	
This crosscutting concern specified that faults generated by NotificationProducer or SubscriptionManager must be complaint to the WS-Base specification.	
Where It Appears	
Line 188 – 190	
Concerns Influenced	
NotificationProducer and SubscriptionManager	
Classification based on Crosscutting Representation	**XC1**
This crosscutting concern is specified in a localized manner with its influenced concerns specified explicitly.	

- **X421**

X421 – All faults must be identified with the given WS-Addressing URI	
Description	
This crosscutting concern specified that all faults in INFOD must be identified with the given WS-Addressing URI.	
Where It Appears	
Line 191 – 192	
Concerns Influenced	
All components in INFOD with generating faults	
Classification based on Crosscutting Representation	**XC2**
This crosscutting concern is specified in a localized manner. Which components in INFOD are responsible for generating faults are unknown.	

Appendix B

Annotated Version of the INFOD Base Specification

The annotated version of the INFOD base specification is created based on the only official release of the INFOD base specification which was revised in May 2007 and released in July 2007. The annotated version is provided separately from this thesis.

Appendix B

Annotated Version of the INFOD Base Specification

GWD-R
Category: Recommendation
INFOD-WG

Stephen Davey, National e-Science Centre
Vijay Dialani, IBM Corporation
Ronny Fehling, Oracle Corporation
Steve Fisher, Rutherford Appleton Laboratory
Dieter Gawlick, Oracle Corporation
Christopher Kantarjiev, Oracle Corporation
Cecile Madsen, IBM Corporation
Susan Malaika, IBM Corporation
Shailendra Mishra, Oracle Corporation
Mallikarjun Shankar, Oak Ridge National Laboratory
Created: 2 Oct 2003
Revised Date: 5 May 2007

Information Dissemination in the Grid Environment – Base Specifications

Status of This Memo

This memo provides a recommendation to the Grid communities. The intention is to define a standard. Distribution is unlimited.

Copyright Notice

Abstract

INFOD (Information Dissemination) provides a general means to determine which messages are to be sent from which publishers to which consumers based upon information kept in a registry. To support this, INFOD specifies interfaces that allow the characterization (in the registry) of publishers, consumers and various other components using vocabularies that are meaningful to members of the communities they belong to. INFOD makes use of a notify operation similar to that defined by the WS-Notification specification to send information between publishers and consumers.

INFOD also extends the publish/subscribe paradigm by allowing consumers to be determined dynamically based on the message content. Additionally, INFOD allows subscribers to determine what defines an event and which messages should be created in response to these events.

Table of Contents

1 Introduction

Having the most up-to-date information available is becoming increasingly important. Rick Hayes-Roth uses the term VIRT (Valuable Information at the Right Time) to capture this requirement[1, 2]. The core idea of VIRT is that consumers of information should receive the information that is relevant to them as soon as it is available or whenever it is needed. COI (Conditions Of Interest) determine which information is needed when and by whom. Information Dissemination[3] (INFOD) provides core technology to support the VIRT objectives for a wide range of applications.

Technology to support basic aspects of VIRT has been well established; JMS, the Java Messaging System, is a good example. JMS supports publishers as information providers and consumers as information recipients. The selection of the information is driven through subscriptions, which represent the COI.

This basic model has been extended by INFOD with:

- **Subscribers**: Subscriptions are typically specified by consumers. By assigning this task to subscribers they can determine a fitting subset of potential consumers based on some properties associated with them; e.g., notify the two security agents closest to an incident. Furthermore, consumers can get information that they did not subscribe to; a chemical spill ahead of me (the consumer) is an example.

- **Data Sources**: Publishers may be able to publish a wide variety of information. This information is organized as data sources. Examples of data sources are queues, (RSS) streams, files, (temporal) databases and applications. The structure, and to some extent the meaning, of the information of each data source is defined by one or more data vocabularies.

- **Data Vocabularies**: Data vocabularies are used to define the structure of information independent of the publishers and the data sources. Data vocabularies can be specified using SQL, XML, RDF or any other method as long as this method supports at least one query/filter language.

- **Property Vocabularies**: Property vocabularies are used to specify XML schemas that can be used to describe a class of publishers, consumers, subscribers and data sources in a way that is meaningful to a community that intends to share information. For example: all the publishers of the car dealer and the consumers of car buyer communities share property vocabularies.

- **Property Vocabulary Instances**: Property vocabulary instances are used to describe specific publishers, consumers, subscribers and data sources; e.g., a publisher who is characterized as a car dealer and further described by its location, its business rating and any other information that may be of interest.

---

[1] Model Based Communication Network and VIRT: Orders of magnitude better for Information Superiority (http://www.w2cog.org/revamp/files/MICOM2006-RHR-VIRT-final-1751.pdf?PHPSESSID=7f4fae6171ea8ef9204bcb000a6d2b67)

[2] Event Processing in the Global Information Grid (GIG) (http://complexevents.com/wp-content/uploads/2007/01/Event_Processing_GIG_RHR.ppt)

[3] The INFOD Base Use Case Scenarios (see http://forge.gridforum.org/sf/go/doc13626?nav=1) provide helpful background information. It describes INFOD patterns and their implementation as well as INFOD Use Cases

Using this extended model, the effort to establish and maintain the desired information flow, i.e., the effort to define and maintain subscriptions, can be significantly reduced. Without the extended model a subscriber needs to determine explicitly which publishers and data sources are of interest. With the extended model subscribers specify the type of information of interest along with the required properties of the publishers and data sources.

INFOD captures information about publishers, consumers, subscriptions, subscribers, data sources, data vocabularies and property vocabularies in a registry, called the INFOD registry. The information in this registry is organized as resources. Some of the resources in the INFOD registry capture information about objects that exist outside of INFOD; e.g., a publisher and consumer are typically web services. Resources that capture information about an external object are called entries.

<x-concern id=X201>

Figure 1 shows the INFOD resources.

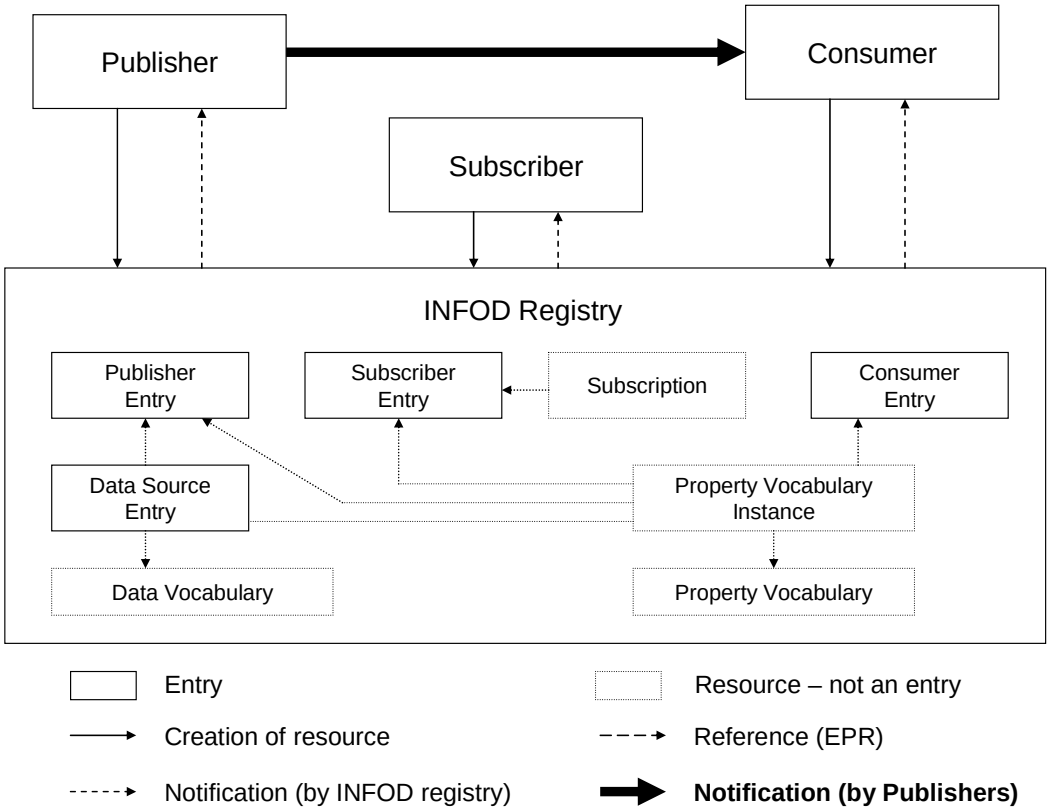

Figure 1: INFOD Resources

</x-concern id=X201>

The main objective of the INFOD registry is to match publishers and consumers

<x-concern id=X202>

and to notify publishers which information has to be delivered to which consumer.

</x-concern id=X202>

The registry is used to manage the information that is required to determine which information (messages) has to flow from which publishers to which consumers.

<x-concern id=X203>

The messages flow directly from the publishers to the consumers making use of a notification system similar to WS-Notification.

</x-concern id=X203>

Here is a list of the contributions of the INFOD model:

- **Property Constraints and Mutual Filtering**:

<x-concern id=X204>

Each entry can specify a set of property constraints referencing information related to other entries;

</x-concern id=X204>

a property constraint is an XQuery or XPath expression referencing entries and property vocabulary instances. Property constraints are used to specify which other entries are eligible to interact with a given entry. Examples of interactions are sending or receiving a message or reacting to a subscription. Property constraints can reference properties of other entries as well as properties captured in property vocabulary instances.

<x-concern id=X205>

Figure 2 shows all property constraints that can be specified between entries. The absence of constraints shows that the interaction is unrestricted.

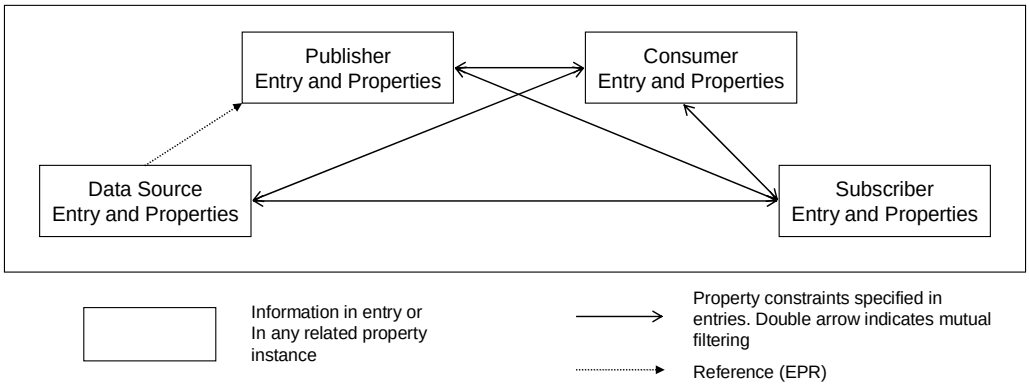

Figure 2: Property Constraints

</x-concern id=X205>

- **Property constraints (in Subscriptions)**: Property constraints can be used in subscriptions to define publishers and consumers instead of identifying publisher and consumers explicitly.

Figure 3 shows all property constraints that can be specified by subscriptions. EPRs can be used to identify entries explicitly. The absence of property constraints shows that there is no limitation in the selection of publishers, data sources and consumers.

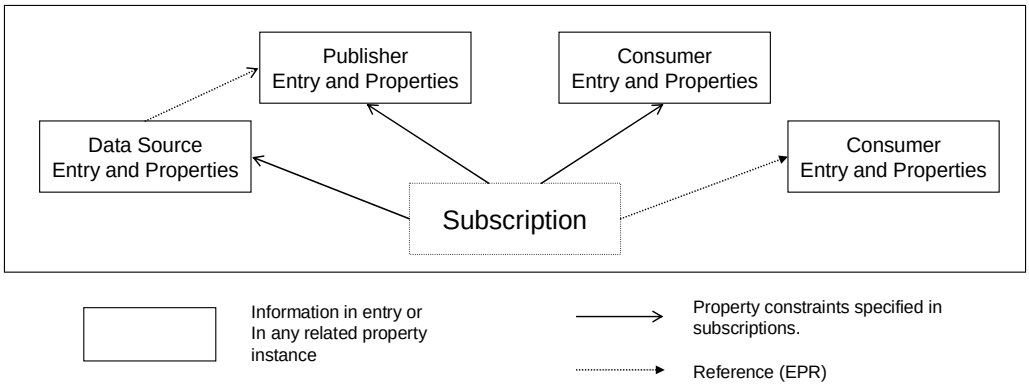

Figure 3: Property Constraints in Subscriptions

The INFOD registry will determine which publishers and consumers conform to the constraints. Any limitation imposed through mutual filtering will be taken into account. The INFOD registry also adapts the information flow to changes of resources; e.g., the INFOD registry will react to new, modified or deleted publisher entries as soon as they become available.

- **Data Constraints (in Subscription)**: A data constraint is a query supported by a data vocabulary. Data constraints are used to specify which information is of interest. To make the information *as valuable as possible* subscribers can specify what an event is by defining conditions or patterns on (temporal) data sources. Additionally, subscribers can specify which information or message should be disseminated in response to an event. Data constraints can only be specified for subscriptions.

  Figure 4 shows those data constraints which data sources eligible. The absence of a data constraint indicates interest in all data.

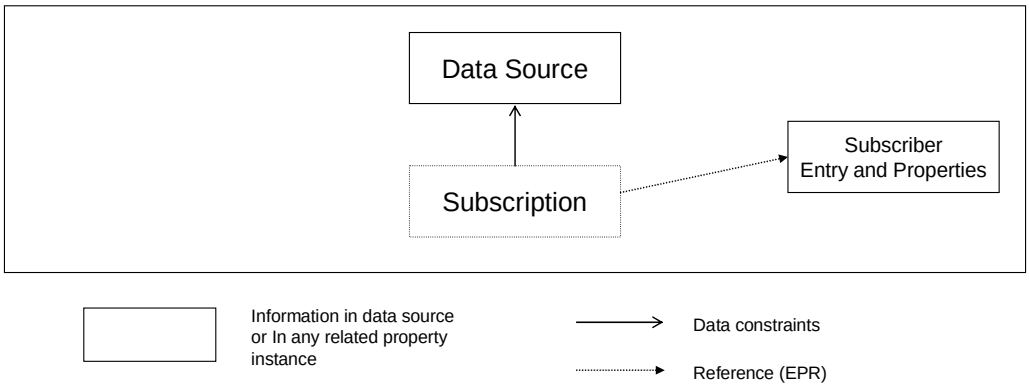

Figure 4: Data Constraints in Subscriptions

Property and data constraints in subscriptions represent the COI in VIRT. Data and property constraints specified in subscription are complemented by property constraints specified in entries.

92 # 1.1 The Registry

93 ## 1.1.1 Resources

94 The registry manages various resources as listed below. A resource is used here as meaning
95 something which is held in the registry.

<x-concern id=X301>

96 Each resource type has calls to create and drop it from the registry.

</x-concern id=X301>

<x-concern id=X302>

97 Some resources have a call to replace them.

</x-concern id=X302>

98 ### 1.1.1.1 Publisher Entry, Consumer Entry and Subscriber Entry

99 As already explained, an entry is the information stored in the registry about an external object.

<x-concern id=X303 part=1/2>

100 Each is identified in the registry by a unique EPR (endpoint reference).

</x-concern id=X303>

<x-concern id=X304 part=1/2>

101 Each entry has a name and description, both of which are optional, not necessarily unique and have
102 string values. They are also both expected to be meaningful to humans.

</x-concern id=X304>

103 Operations are provided to create, replace and drop these entries. Note that these verbs are with
104 respect to the entries in the registry and not the external object, so we talk about creating a publisher
105 entry rather than registering a publisher.

<x-concern id=X305>

106 The act of creation involves storing information and returning the EPR of the entry. The creation
107 operation will often store the EPR of the external object. This is the only place the external EPR,
108 identifying the external object, is stored. All other references to EPRs are to EPRs of resources.

</x-concern id=X305>

<x-concern id=X306>

109 The replace operation (for example ReplacePublisher in Section 2.1.2) takes the EPR that was
110 returned by the create operation as an additional parameter and keeps only the identity of the entry:
111 all the data associated with it by the create operation is replaced by new data however all relations
112 established after the original entry was created are preserved as the identity of the entry remains
113 unchanged.

</x-concern id=X306>

<x-concern id=X307>

The drop operation (for example DropPublisher in Section 2.1.3) takes the EPR of the entry and makes the stored entry unavailable and so makes subsequent use of the EPR invalid. The drop operation is not allowed to make the system inconsistent (see Section 1.1.2) so, by default, an error will be reported if an attempt is made to drop an entry which is still referenced. There is an optional flag which can be set to “DISABLE NEW REFERENCES” which results in the entry being dropped when the last reference to the entry has been removed and “CASCADE”, which also drops (recursively) all entries referencing that entry.

</x-concern id=X307>

1.1.1.2 Data Vocabularies and Data Source Entries

Data are only useful if there is a shared understanding of these data by publishers, consumers and subscribers. For this purpose INFOD uses vocabularies, which are maintained within the registry. Data vocabularies describe the structure of the data that is available from publishers. It is the responsibility of a community of users with a common interest to define a data vocabulary and register it as the first step in using INFOD. For flexibility, data vocabularies can be specified using SQL, XML, RDF or any other data model. The INFOD registry will *not* manage instances of user data. A data vocabulary is used by the registry to carry out vocabulary specific operations. Vocabularies are managed, with operations such as CreateDataVocabulary (Section 2.5.5) to store information about the data vocabulary in the registry.

A data source entry is created by an operation called CreateDataSourceEntry. This represents an association between data vocabularies and entries – specifically publisher entries thereby identifying the publisher as a source of some specific type of information.

<x-concern id=X303 part=2/2>

Data Source Entries, like other entries have their own EPR

</x-concern id=X303>

<x-concern id=X304 part=2/2>

and an optional name and description.

</x-concern id=X304>

In addition they have the EPR of the two things they are relating.

1.1.1.3 Property Vocabularies and Property Vocabulary Instances

A user community may also define property vocabularies to allow property constraints to be defined. These vocabularies, which are optional, are expressed by an XML schema.

The CreatePropertyVocabularyInstance call (Section 2.5.2) is then used to create a The Property Vocabulary Instance which holds actual values for a particular Publisher, Consumer or Subscriber entry. The Property Vocabulary Instance references a Property Vocabulary.

Constraints identifying which other resources are of interest or unacceptable may be expressed using these properties.

Property vocabularies can be used as an extension mechanism to define notions such as quality of service. In a future version of the document this extension mechanism may be used to formalize properties such as operational characteristics.

1.1.1.4 Subscriptions and Constraints

<x-concern id=X308>

No information starts flowing in an INFOD system until a subscription is created.

</x-concern id=X308>

A subscription normally defines various constraints. In the absence of all constraints a subscription will cause all messages to be sent from all publishers to all consumers. In practice producers have constraints to indicate who they will send messages to, consumers have constraints to say who they will get messages from and a subscription will normally have at least a data constraint indicating what kind of messages are wanted. The registry acts on subscriptions by finding matching publishers and consumers using the property constraints of publisher entries, consumer entries, subscriber entries, subscriptions and data source entries along with data constraints of subscriptions expressed in terms of a data vocabulary.

<x-concern id=X309>

In addition a subscription may include dynamic consumer constraints. These are constraints which are evaluated by the consumer rather than the registry by looking at the contents of a potential message.

</x-concern id=X309>

As already mentioned the subscription is not an entry as it has no counterpart outside the registry. The create operation returns an EPR.

1.1.2 Dependencies

<x-concern id=X310>

A basic dependency rule governs the creation, modification and removal of resources within the INFOD registry: only resources that are registered in the INFOD registry can be referenced.

</x-concern id=X310>

<x-concern id=X311>

Figure 5 shows the relations between the various INFOD resources. The arrows show the direction of reference. In addition the P or D in the box shows which resources may hold property or data constraints respectively.

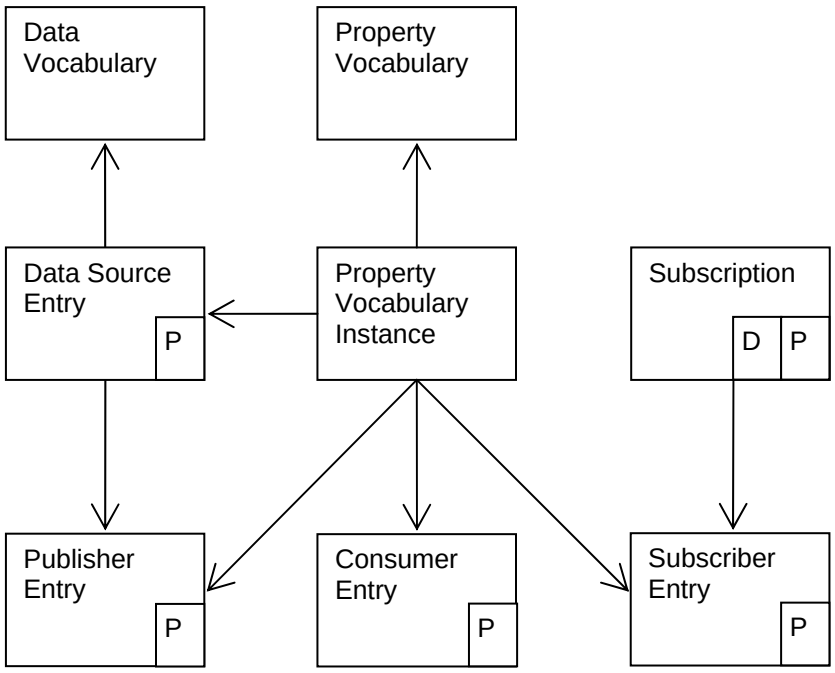

Figure 5: Relation between INFOD resources

</x-concern id=X311>

1.1.3 Matching Publishers to Subscriptions

Discovery of publishers to match a specific subscription requires the registry to examine the vocabularies and all the constraints so that it can generate correct notifications. Instead of using notifications the GetMetaData operation (see section 2.7) may be used to query the information in the INFOD registry and, most importantly, to look up matching subscriptions and publishers.

<x-concern id=X312>

INFOD objects, especially publishers need to react immediately to changes in the INFO registry. They may register to be notified of any changes that are significant for them.

</x-concern id=X312>

1.2 Security

<x-concern id=X206>

INFOD uses existing security mechanisms to ensure that the dissemination of information happens according to security policies.

</x-concern id=X206>

The specification of communities can be used to complement and enhance security policies.

1.3 Lifetime Management

The INFOD specification does not contain any specific resource lifetime management other than the facilities to remove INFOD resources, for example *DropSubscription* etc. However, to ensure that in cases where a client becomes disconnected from the INFOD Registry and is unable or unwilling to destroy obsolete INFOD resources, some form of lifetime management should be employed such as WS-ResourceLifetime (see http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-03.pdf).

<x-concern id=X207>

This should provide a mechanism by which resources may be destroyed after a period of time unless the scheduled termination time is extended.

</x-concern id=X207>

1.4 Fault Definitions

<x-concern id=X420>

All faults generated by a NotificationProducer or SubscriptionManager should be compliant with the WS-BaseFaults (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf) specification.

</x-concern id=X420>

<x-concern id=X421>

All faults defined by this specification MUST use the following URI for the WS-Addressing [action]: http://www.ogf.org/infod/fault.

</x-concern id=X421>

2 The Base INFOD Registry Interface

<<The following table defines the dominant decomposition structure of this chapter>>

The tables below list the operations of the base INFOD registry interface and the section that describes them in detail.

The Base INFOD Registry Interface:

Operation		Description	Section
Managing Publisher Entries	CreatePublisherEntry	This operation defines how to create a new Publisher entry in an INFOD registry.	2.1.1
	ReplacePublisherEntry	This operation defines how to replace a particular Publisher entry in an INFOD registry.	2.1.2
	DropPublisherEntry	This operation defines how to drop an existing Publisher entry from an INFOD registry.	2.1.3
Managing Subscriber Entries	CreateSubscriberEntry	This operation defines how to create a new Subscriber entry in an INFOD registry.	2.2.1
	ReplaceSubscriberEntry	This operation defines how to replace a particular Subscriber entry in an INFOD registry.	2.2.2
	DropSubscriberEntry	This operation defines how to drop an existing Subscriber entry from an INFOD registry.	2.2.3
Managing Consumer Entries	CreateConsumerEntry	This operation defines how to create a new Consumer entry in an INFOD registry.	2.3.1
	ReplaceConsumerEntry	This operation defines how to replace a particular Consumer entry in an INFOD registry.	2.3.2
	DropConsumerEntry	This operation defines how to drop an existing Consumer entry from an INFOD registry.	2.3.3
Managing Subscriptions	CreateSubscription	This operation defines how to create a new Subscription in an INFOD registry.	2.4.1
	ReplaceSubscription	This operation defines how to replace a particular Subscription in an INFOD registry.	2.4.2
	DropSubscription	This operation defines how to drop an existing Subscription from an INFOD registry.	2.4.3

Managing Vocabularies and Property Vocabulary Instances	CreatePropertyVocabulary	This operation defines how to create a property vocabulary to an INFOD registry	2.5.1
	DropPropertyVocabulary	This operation defines how to drop a property vocabulary from an INFOD registry.	2.5.2
	CreatePropertyVocabularyInstance	This operation creates a new instance of a property vocabulary that is already registered in an INFOD registry.	2.5.3
	DropPropertyVocabularyInstance	This operation drops an existing instance of a particular property vocabulary registered in an INFOD registry.	2.5.4
	CreateDataVocabulary	This operation defines how to create a data vocabulary to an INFOD registry	2.5.5
	DropDataVocabulary	This operation defines how to drop a data vocabulary from an INFOD registry.	2.5.6
Managing Data Sources	CreateDataSource	This operation defines how to create a data source in an INFOD registry.	2.6.1
	DropDataSource	This operation defines how to drop a data source from a particular Publisher entry in an INFOD registry.	2.6.2
	GetMetaData	This operation queries the metadata of resources defined in a particular INFOD registry.	2.7

2.1 Managing Publisher Entries

These operations are used to manage publishers

- CreatePublisherEntry (section 2.1.1)
- ReplacePublisherEntry (section 2.1.2)
- DropPublisherEntry (section 2.1.3)

2.1.1 CreatePublisherEntry

<x-concern id=X401 part=1/8>

As part of the processing of a CreatePublisherEntry request message, the INFOD registry MUST create an INFOD entry and an EPR representing the publisher entry.

</x-concern id=X401>

The format of the request message for the CreatePublisherEntry operation is based on the schema provided in Appendix 1 definition for an INFOD entry. Details are as follows:

```
<infod:CreatePublisherEntry>
  <infod:WSReference>
    wsa:EndPointReferenceType
  </infod:WSReference> ?
  <infod:PublisherName> xsd:string </infod:PublisherName> ?
```

```
  <infod:PublisherDescription>
    xsd:string
  </infod:PublisherDescription> ?
  <infod:PropertyConstraint>
    xsd:any
  </infod:PropertyConstraint> *
  <infod:Notification>
    xsd:boolean default ”FALSE”
  </infod:Notification> ?
</infod:CreatePublisherEntry>
```

The elements of the CreatePublisherEntry message are further described as follows:

/infod:WSReference

<x-concern id=X402 part=1/6>

An endpoint reference element, as defined by WS-Addressing, used to identify the WS endpoint for the entry. Note that this MAY be the WS EPR of the requesting service, but does not have to be.

</x-concern id=X402>

/infod:PublisherName

A string representing the name of the publisher. This name MAY NOT be unique.

/infod:PublisherDescription

A string representing a description of the publisher.

/infod:PropertyConstraint

Property contraints are used to specify which conditions must be satisfied by other entries (consumers, data sources and subscribers) to be eligible for interaction with this publisher.

<x-concern id=X403 part=1/9>

A property constraint MUST be formulated as an XQuery. The INFOD Base Use Case Scenarios (see http://forge.gridforum.org/sf/go/doc13626?nav=1) provide examples of XQueries.

For example, a publisher identifies the set of consumers that are eligible to receive data by formulating property constraints.

Note that the XQuery statement MUST be encoded correctly, i.e. characters such as “>” would be represented as “>”

</x-concern id=X403>

/infod:Notification

<x-concern id=X404 part=1/6>

When used, the registry MUST notify the publisher about changes relevant in the registry. A fault MUST be returned if infod:WSReference is not specified.

</x-concern id=X404>

For further details see section 3.2.1

<x-concern id=X418 part=1/21>

A WS-Addressing Action header with the value
http://www.ogf.org/infod/INFODRegistry/CreatePublisherEntry MUST accompany the message.

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the CreatePublisherEntry message, it MUST respond to the WS endpoint specified in the request message with a CreatePublisherEntryResponse message. The CreatePublisherEntryResponse message is a message of the following form:

```
<infod:CreatePublisherEntryResponse>
    <infod:PublisherEntryReference>
       wsa:EndPointReferenceType
    </infod:PublisherEntryReference>
</infod:CreatePublisherEntryResponse>
```

The elements of the CreatePublisherEntryResponse message are further described as follows:

/infod:PublisherEntryReference

An endpoint reference element, as defined by WS-Addressing, used to identify the newly created publisher entry in the INFOD registry.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X406 part=1/8>

- CreateResourceAuthorizationFault: User not authorized to create the INFOD resource at this INFOD registry

</x-concern id=X406>

<x-concern id=X407 part=1/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X408 part=1/10>

- UnsupportedXQueryFault: The XQuery specified could not be parsed correctly

</x-concern id=X408>

<x-concern id=X405 part=1/21>

The message MUST be structured according to the WS-Base Faults specification. For examples using SOAP, see the SOAP v1.2 Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

Example SOAP Encoding of the Create Publisher Message

The following is a non-normative example of a CreatePublisherEntry request message using SOAP:

```
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
      http://www.ogf.org/infod/INFODRegistry/CreatePublisherEntry
    </wsa:Action>
    ...
```

```
  </s:Header>
  <s:Body>
    <infod:CreatePublisherEntry>
      <infod:WSReference>
        <wsa:Address>
          http://www.example.org/SomePublisher
        </wsa:Address>
      </infod:WSReference>
      <infod:PublisherName>
        SomePublisher
      </infod:PublisherName>
      <infod:PublisherDescription>
        This publisher can publish some information
      </infod:PublisherDescription>
      <infod:PropertyConstraints>
       fn:doc(”INFODRegistry.xml”)/Consumers/infodConsumer
         [fn:contains(ConsumerName,”Ronny”}]
      </infod:PropertyConstraints>
      <infod:Notification>
        TRUE
      </infod:Notification>
    </infod:CreatePublisherEntry>
  </s:Body>
</s:Envelope>
```

2.1.2 ReplacePublisherEntry

The ReplacePublisherEntry operation replaces an INFOD publisher entry’s metadata information at a given INFOD registry.

<x-concern id=X414 part=1/4>

As part of the processing of a ReplacePublisherEntry message, the INFOD registry MUST replace the entire INFOD metadata for the entry representing the publisher. All previously defined values MUST be deleted. The ReplacePublisherEntry differs from the CreatePublisherEntry interface in that it replaces an existing publisher entry and assigns the original EPR to the replaced publisher.

</x-concern id=X414>

The format of the request message for a ReplacePublisherEntry operation is also based on the schema definition provided in Appendix 1 for an INFOD entry. Details are as follows:

```
<infod:ReplacePublisherEntry>
  <infod:WSReference>
    wsa:EndPointReferenceType
  </infod:WSReference> ?
  <infod:PublisherEntryReference>
    wsa:EndPointReferenceType
  </infod:PublisherEntryReference>
  <infod:PublisherName> xsd:string </infod:PublisherName> ?
  <infod:PublisherDescription>
    xsd:string
  </infod:PublisherDescription> ?
  <infod:PropertyConstraint>
    xsd:any
  </infod:PropertyConstraint> *
  <infod:Notification>
    xsd:boolean “FALSE”
  </infod:Notification> ?
</infod:ReplacePublisherEntry>
```

The elements of the ReplacePublisherEntry message are further described as follows:

/infod:WSReference

<x-concern id=X402 part=2/6>

An endpoint reference element, as defined by WS-Addressing, used to identify the WS endpoint for the entry. Note that this MAY be the WS EPR of the requesting service, but does not have to be. The request MAY be made 'on behalf' of the actual service.

</x-concern id=X402>

/infod:PublisherEntryReference

An endpoint reference element, as defined by WS-Addressing, used to identify the publisher entry in the INFOD registry that will be replaced.

/infod:PublisherName

A string representing the name of the publisher. This name MAY NOT be unique.

/infod:PublisherDescription

A string representing a description of the publisher.

/infod:PropertyConstraint

Property contraints are used to specify which conditions must be safisfied by other entries (consumers, data sources and subscribers) to be eligible for interaction with this publisher.

<x-concern id=X403 part=2/9>

A property constraint MUST be formulated as an XQuery. The INFOD Base Use Case Scenarios (see http://forge.gridforum.org/sf/go/doc13626?nav=1) provide examples of XQueries.

For example, a publisher identifies the set of consumers that are eligible to receive data by formulating property constraints.

Note that the XQuery statement MUST be encoded correctly, i.e. characters such as ">" would be represented as ">"

</x-concern id=X403>

/infod:Notification

<x-concern id=X404 part=2/6>

When used, the registry MUST notify the publisher about changes relevant in the registry. A fault MUST be returned if infod:WSReference is not specified.

</x-concern id=X404>

For further details see section 3.2.1

<x-concern id=X418 part=2/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/ReplacePublisherEntry MUST accompany the message.

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the ReplacePublisherEntry message, it MUST respond to the WS endpoint specified in the request message with a ReplacePublisherEntryResponse message. The ReplacePublisherEntryResponse message is a message of the following form:

```
<infod:ReplacePublisherEntryResponse>
  <infod:Status>
 xsd:string default ”COMPLETED”
  </infod:Status>
</infod:ReplacePublisherEntryResponse>
```

The elements of the ReplacePublisherEntryResponse message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X409 part=1/4>

- ReplaceResourceAuthorizationFault: User not authorized to replace the INFOD resource at this INFOD registry

</x-concern id=X409>

<x-concern id=X410 part=1/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=2/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X408 part=2/10>

- UnsupportedXQueryFault: The XQuery specified could not be parsed correctly

</x-concern id=X408>

<x-concern id=X405 part=2/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.1.3 DropPublisherEntry

The DropPublisherEntry operation removes an INFOD publisher entry from an INFOD registry.

The format of the request message for a DropPublisherEntry operation is:

```
<infod:DropPublisherEntry>
  <infod:PublisherEntryReference>
    wsa:EndPointReferenceType
  </infod:PublisherEntryReference>
  <infod:ExecutionMode> xsd:string </infod:ExecutionMode> ?
```

```
</infod:DropPublisherEntry>
```

The elements of the DropPublisherEntry message are further described as follows:

/infod:PublisherEntryReference

An endpoint reference element, as defined by WS-Addressing, used to identify the INFOD resource in the registry to drop.

/infod:ExecutionMode

<x-concern id=X415 part=1/8>

A parameter indicating the mode of execution of the drop request. Possible values are:

"IF UNUSED"	The drop request will execute only if the resource is unreferenced
"DISABLE NEW"	No new references are possible for the resource. The resource will be dropped when the last reference to this resource is gone
"CASCADE"	The drop request will execute immediately and all references to the resource will be removed recursively

If this parameter is not specified, the default value "IF UNUSED" MUST be used.

</x-concern id=X415>

<x-concern id=X418 part=3/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/DropPublisherEntry MUST accompany the message.

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the DropPublisherEntry message, it MUST respond to the WS endpoint specified in the request message with a DropPublisherEntryResponse message. The DropPublisherEntryResponse message is a message of the following form:

```
<infod:DropPublisherEntryResponse>
  <infod:Status>
 xsd:string default "COMPLETED"
  </infod:Status>
</infod:DropPublisherEntryResponse>
```

The elements of the DropPublisherEntryResponse message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X411 part=1/8>

- DropResourceAuthorizationFailure: User not authorized to drop the INFOD resource at this INFOD registry

</x-concern id=X411>

<x-concern id=X410 part=2/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=3/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X412 part=1/8>

- ExecutionModeFault: Cannot use ExecutionMode provided

</x-concern id=X412>

<x-concern id=X405 part=3/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.2 Managing Subscriber Entries

The following operations are used to manage subscribers:

- CreateSubscriberEntry (section 2.2.1)
- ReplaceSubscriberEntry (section 2.2.2)
- DropSubscriberEntry (section 2.2.3)

2.2.1 CreateSubscriberEntry

<x-concern id=X401 part=2/8>

As part of the processing of a CreateSubscriberEntry request message, the INFOD registry MUST create an INFOD entry and an EPR representing the subscriber.

</x-concern id=X401>

The format of the request message for CreateSubscriberEntry operation is based on the schema provided in Appendix 1 for an INFOD entry. Details are as follows:

```
<infod:CreateSubscriberEntry>
  <infod:WSReference>
    wsa:EndPointReferenceType
  </infod:WSReference> ?
  <infod:SubscriberName> xsd:string </infod:SubscriberName> ?
  <infod:SubscriberDescription>
    xsd:string
  </infod:SubscriberDescription> ?
  <infod:PropertyConstraint>
    xsd:any
</infod:PropertyConstraint> *
<infod:Notification>
  xsd:Boolean default “FALSE”
</infod:Notification> ?
</infod:CreateSubscriber>
```

The elements of the CreateSubscriberEntry message are further described as follows:

/infod:WSReference

<x-concern id=X402 part=3/6>

An endpoint reference element, as defined by WS-Addressing, used to identify the WS endpoint for the entry. Note that this MAY be the WS EPR of the requesting service, but does not have to be. The request MAY be made 'on behalf' of the actual service.

</x-concern id=X402>

/infod:SubscriberName

A string representing the name of the subscriber name, this name MAY NOT be unique.

/infod:SubscriberDescription

A string representing a description of the subscriber.

/infod:PropertyConstraint

Property contraints are used to specify which conditions must be safisfied by other entries (publishers, data sources, and consumers) to be eligible for interaction with this subscriber.

<x-concern id=X403 part=3/9>

A property constraint MUST be formulated as an XQuery. The INFOD Base Use Case Scenarios (see http://forge.gridforum.org/sf/go/doc13626?nav=1) provide examples of XQueries.

For example, a subscriber identifies the set of publishers that are eligible to react to subscriptions specified by this subscriber.

Note that the XQuery statement MUST be encoded correctly, i.e. characters such as ">" would be represented as ">"

</x-concern id=X403>

infod:Notification

<x-concern id=X404 part=3/6>

When used, the registry MUST notify the subscriber about relevant changes in the INFOD registry. A fault MUST be returned if infod:WSReference is not specified.

</x-concern id=X404>

For further details see section 3.2.2.

<x-concern id=X418 part=4/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/CreateSubscriberEntry MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the CreateSubscriberEntry message, it MUST respond to the WS endpoint specified in the request message with a CreateSubscriberEntryResponse message. The CreateSubscriberEntry response message is a message of the following form:

```
<infod:CreateSubscriberEntryResponse>
    <infod:SubscriberEntityReference>
       wsa:EndPointReferenceType
    </infod:SubscriberEntityReference>
</infod:CreateSubscriberEntryResponse>
```

The elements of the CreateSubscriberEntryResponse message are further described as follows:

/infod:SubscriberEntityReference

An endpoint reference element, as defined by WS-Addressing, used to identify the newly created subscriber entry in the INFOD registry.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X406 part=2/8>

- CreateResourceAuthorizationFault: User not authorized to create the INFOD resource at this INFOD registry

</x-concern id=X406>

<x-concern id=X407 part=4/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X408 part=3/10>

- UnsupportedXQueryFault: The XQuery specified could not be parsed correctly

</x-concern id=X408>

<x-concern id=X405 part=4/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.2.2 ReplaceSubscriberEntry

The ReplaceSubscriberEntry operation replaces an INFOD subscriber entry's metadata information at a given INFOD registry.

<x-concern id=X414 part=2/4>

As part of the processing of a ReplaceSubscriberEntry request message, the INFOD Registry MUST replace the entire INFOD metadata for the entry representing the subscriber. All previously defined values MUST be deleted. The ReplaceSubscriberEntry differs from the CreateSubscriberEntry interface in that it replaces an existing subscriber entry and assigns the original EPR to the replaced subscriber.

</x-concern id=X414>

The format of the request message for a ReplaceSubscriberEntry operation is also based on the schema definition provided in Appendix 1 for an INFOD entry. Details are as follows:

```
<infod:ReplaceSubscriberEntryEntry>
  <infod:WSReference>
    wsa:EndPointReferenceType
```

```
  </infod:WSReference> ?
  <infod:SubscriberEntryReference>
    wsa:EndPointReferenceType
  </infod:SubscriberEntryReference>
  <infod:SubscriberName> xsd:string </infod:SubscriberName> ?
  <infod:SubscriberDescription>
    xsd:string
  </infod:SubscriberDescription> ?
  <infod:PropertyConstraint>
    xsd:any
</infod:PropertyConstraint> *
 <infod:Notification>
  xsd:Boolean default "FALSE"
</infod:Notification> ?
</infod:ReplaceSubscriberEntry>
```

The elements of the ReplaceSubscriberEntry message are further described as follows:

/infod:WSReference

<x-concern id=X402 part=4/6>

An endpoint reference element, as defined by WS-Addressing, used to identify the WS endpoint for the entry. Note that this MAY be the WS EPR of the requesting service, but does not have to be. The request MAY be made 'on behalf' of the actual service.

</x-concern id=X402>

/infod:SubscriberEntryReference

An endpoint reference element, as defined by WS-Addressing, used to identify the subscriber entry in the INFOD registry that will be replaced.

/infod:SubscriberName

A string representing the name of the subscriber. This name MAY NOT be unique.

/infod:SubscriberDescription

A string representing a description of the subscriber.

/infod:PropertyConstraint

Property contraints are used to specify which conditions must be safisfied by other entries (publishers, data sources, and consumers) to be eligible for interaction with this subscriber.

<x-concern id=X403 part=4/9>

A property constraint MUST be formulated as an XQuery. The INFOD Base Use Case Scenarios (see http://forge.gridforum.org/sf/go/doc13626?nav=1) provide examples of XQueries.

For example, a subscriber identifies the set of publishers that are eligible to react to subscriptions specified by this subscriber.

Note that the XQuery statement MUST be encoded correctly, i.e. characters such as ">" would be represented as ">"

</x-concern id=X403>

infod:Notification

<x-concern id=X404 part=4/6>

When used, the registry MUST notify the subscriber about relevant changes in the INFOD registry. A fault MUST be returned if infod:WSReference is not specified.

</x-concern id=X404>

For further details see section 3.2.2.

<x-concern id=X418 part=5/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/ReplaceSubscriberEntry MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the ReplaceSubscriberEntry message, it MUST respond to the WS endpoint specified in the request message with a ReplaceSubscriberEntryResponse message. The ReplaceEntrySubscriber response message is a message of the following form:

```
<infod:ReplaceSubscriberEntryResponse>
  <infod:Status>
 xsd:string default ”COMPLETED”
  </infod:Status>
</infod:ReplaceSubscriberEntryResponse>
```

The elements of the ReplaceSubscriberEntryResponse message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X409 part=2/4>

- ReplaceResourceAuthorizationFault: User not authorized to replace the INFOD resource at this INFOD registry

</x-concern id=X409>

<x-concern id=X410 part=3/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=5/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X408 part=4/10>

- UnsupportedXQueryFault: The XQuery specified could not be parsed correctly

</x-concern id=X408>

<x-concern id=X405 part=5/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.2.3 DropSubscriberEntry

The DropSubscriberEntry operation removes an INFOD subscriber entry from an INFOD registry.

The format of the request message for a DropSubscriberEntry operation is:

```
<infod:DropSubscriberEntry>
  <infod:SubscriberEntryReference>
    wsa:EndPointReferenceType
  </infod:SubscriberEntryReference>
  <infod:ExecutionMode> xsd:string </infod:ExecutionMode>
</infod:DropSubscriberEntry>
```

The elements of the DropSubscriberEntry message are further described as follows:

/infod:ResourceReference

An endpoint reference element, as defined by WS-Addressing, used to identify the INFOD resource in the registry to drop.

/infod:ExecutionMode

<x-concern id=X415 part=2/8>

A parameter indicating the mode of execution of the drop request. Possible values are:

"IF UNUSED"	The drop request will execute only if the resource is unreferenced
"DISABLE NEW"	No new references are possible for the resource. The resource will be dropped when the last reference to this resource is gone
"CASCADE"	The drop request will execute immediately and all references to the resource will be removed recursively

If this parameter is not specified, the default value "IF UNUSED" MUST be used.

</x-concern id=X415>

<x-concern id=X418 part=6/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/DropSubscriberEntry MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the DropSubscriberEntry message, it MUST respond to the WS endpoint specified in the request message with a DropSubscriberEntryResponse message. The DropSubscriberEntry response message is a message of the following form:

```
<infod:DropSubscriberEntryResponse>
  <infod:Status>
 xsd:string default "COMPLETED"
  </infod:Status>
</infod:DropSubscriberEntryResponse>
```

The elements of the DropSubscriberEntryResponse message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X411 part=2/8>

- DropResourceAuthorizationFailure: User not authorized to drop the INFOD resource at this INFOD registry

</x-concern id=X411>

<x-concern id=X410 part=4/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=6/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X412 part=2/8>

- ExecutionModeFault: Cannot use ExecutionMode provided

</x-concern id=X412>

<x-concern id=X405 part=6/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.3 Managing Consumer Entries

The following operations are used to manage consumers:

- CreateConsumerEntry (section 2.3.1)
- ReplaceConsumerEntry (section 2.3.2)
- DropConsumerEntry (section 2.3.3)

2.3.1 CreateConsumerEntry

<x-concern id=X401 part=3/8>

As part of the processing of a CreateConsumerEntry request message, the INFOD registry MUST create an INFOD entry and an EPR representing the consumer.

</x-concern id=X401>

The format of the request message for CreateConsumerEntry operation is based on the schema provided in Appendix 1 for an INFOD entry. Details are as follows:

```
<infod:CreateConsumerEntry>
  <infod:WSReference>
    wsa:EndPointReferenceType
  </infod:WSReference>
  <infod:ConsumerName> xsd:string </infod:ConsumerName> ?
  <infod:ConsumerDescription>
    xsd:string
  </infod:ConsumerDescription> ?
  <infod:PropertyConstraint>
    xsd:any
</infod:PropertyConstraint> *
  <infod:Notification>
  xsd:Boolean default “FALSE”
</infod:Notification> ?
</infod:CreateConsumerEntry>
```

The elements of the CreateConsumerEntry message are further described as follows:

/infod:WSReference

<x-concern id=X402 part=5/6>

An endpoint reference element, as defined by WS-Addressing, used to identify the WS endpoint for the entry. Note that this MAY be the WS EPR of the requesting service, but does not have to be. The request MAY be made ‘on behalf’ of the actual service.

</x-concern id=X402>

/infod:ConsumerName

A string representing the name of the consumer. This name MAY NOT be unique.

/infod:ConsumerDescription

A string representing a description of the consumer

/infod:PropertyConstraint

Property contraints are used to specify which conditions must be safisfied by other entries (publishers, data sources, and subscribers) to be eligible for interaction with this consumer.

<x-concern id=X403 part=5/9>

A property constraint MUST be formulated as an XQuery. The INFOD Base Use Case Scenarios (see http://forge.gridforum.org/sf/go/doc13626?nav=1) provide examples of XQueries.

For example, a consumer identifies the set of publishers that are eligible to react to subscriptions.

Note that the XQuery statement MUST be encoded correctly, i.e. characters such as “>” would be represented as “>”

</x-concern id=X403>

infod:Notification

<x-concern id=X404 part=5/6>

When used, the registry MUST notify the consumer about relevant changes in the INFOD registry. A fault MUST be returned if infod:WSReference is not specified.

</x-concern id=X404>

For further details see section 3.2.3.

<x-concern id=X418 part=7/21>

A WS-Addressing Action header with the value
http://www.ogf.org/infod/INFODRegistry/CreateConsumerEntry MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the CreateConsumerEntry message, it MUST respond to the WS endpoint specified in the request message with a CreateConsumerEntryResponse message. The CreateConsumerEntry response message is a message of the following form:

```
<infod:CreateConsumerEntryResponse>
  <infod:ConsumerEntryReference>
    wsa:EndPointReferenceType
  </infod:ConsumerEntryReference>
</infod:CreateConsumerEntryResponse>
```

The elements of the CreateConsumerEntryResponse message are further described as follows:

/infod:ConsumerEntryReference

An endpoint reference element, as defined by WS-Addressing, used to identify the newly created consumer entry in the INFOD registry.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X406 part=3/8>

- CreateResourceAuthorizationFault: User not authorized to create the INFOD resource at this INFOD registry

</x-concern id=X406>

<x-concern id=X407 part=7/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X408 part=5/10>

- UnsupportedXQueryFault: The XQuery specified could not be parsed correctly

</x-concern id=X408>

<x-concern id=X405 part=7/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.3.2 ReplaceConsumerEntry

<x-concern id=X414 part=3/4>

As part of the processing of a ReplaceConsumerEntry request message, the INFOD registry MUST replace the entire INFOD metadata for the entry representing the consumer. All previously defined values MUST be deleted. The ReplaceConsumerEntry differs from the CreateConsumerEntry interface in that it replaces an existing consumer entry and assigns the original EPR to the replaced consumer.

</x-concern id=X414>

The format of the request message for a ReplaceConsumer operation is also based on the schema definition provided in Appendix 1 for an INFOD entry. Details are as follows:

```
<infod:ReplaceConsumerEntry>
  <infod:WSReference>
    wsa:EndPointReferenceType
  </infod:WSReference>
  <infod:ConsumerEntryReference>
    wsa:EndPointReferenceType
  </infod:ConsumerEntryReference>
  <infod:ConsumerName> xsd:string </infod:ConsumerName> ?
  <infod:ConsumerDescription>
    xsd:string
  </infod:ConsumerDescription> ?
  <infod:PropertyConstraint>
    xsd:any
</infod:PropertyConstraint> *
  <infod:Notification>
  xsd:Boolean default “FALSE”
</infod:Notification> ?
</infod:ReplaceConsumerEntry>
```

The elements of the ReplaceConsumerEntry message are further described as follows:

/infod:WSReference

<x-concern id=X402 part=6/6>

A REQUIRED endpoint reference element, as defined by WS-Addressing, used to identify the WS endpoint for the entry. Note that this MAY be the WS EPR of the requesting service, but does not have to be. The request MAY be made ‘on behalf’ of the actual service.

</x-concern id=X402>

/infod:ConsumerEntryReference

A REQUIRED endpoint reference element, as defined by WS-Addressing, used to identify the resource in the INFOD registry that will be replaced.

/infod:ConsumerName

A string representing the name of the consumer. This name MAY NOT be unique.

/infod:ConsumerDescription

A string representing a description of the consumer

/infod:PropertyConstraint

Property contraints are used to specify which conditions must be safisfied by other entries (publishers, data sources, and subscribers) to be eligible for interaction with this consumer.

<x-concern id=X403 part=6/9>

A property constraint MUST be formulated as an XQuery. The INFOD Base Use Case Scenarios (see http://forge.gridforum.org/sf/go/doc13626?nav=1) provide examples of XQueries.

For example, a consumer identifies the set of publishers that are eligible to react to subscriptions.

Note that the XQuery statement MUST be encoded correctly, i.e. characters such as ">" would be represented as ">"

</x-concern id=X403>

infod:Notification

<x-concern id=X404 part=6/6>

When used, the registry MUST notify the consumer about relevant changes in the INFOD registry. A fault MUST be returned if infod:WSReference is not specified.

</x-concern id=X404>

For further details see section 3.2.3.

<x-concern id=X418 part=8/21>

A WS-Addressing Action header with the value
http://www.ogf.org/infod/INFODRegistry/ReplaceConsumerEntry MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the ReplaceConsumerEntry message, it MUST respond to the WS endpoint specified in the request message with a ReplaceConsumerEntryResponse message. The ReplaceConsumerEntry response message is a message of the following form:

```
<infod:ReplaceConsumerEntryResponse>
  <infod:Status>
 xsd:string default ”COMPLETED”
  </infod:Status>
</infod:ReplaceConsumerEntryResponse>
```

The elements of the ReplaceConsumerEntryResponse message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X409 part=3/4>

- ReplaceResourceAuthorizationFault: User not authorized to replace the INFOD resource at this INFOD registry

</x-concern id=X409>

<x-concern id=X410 part=5/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=8/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X408 part=6/10>

- UnsupportedXQueryFault: The XQuery specified could not be parsed correctly

</x-concern id=X408>

<x-concern id=X405 part=8/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.3.3 DropConsumerEntry

The DropConsumerEntry operation removes an INFOD consumer entry from an INFOD registry.

The format of the request message for a DropConsumerEntry operation is:

```
<infod:DropConsumerEntry>
  <infod:ConsumerEntryReference>
    wsa:EndPointReferenceType
  </infod:ConsumerEntryReference>
  <infod:ExecutionMode> xsd:string </infod:ExecutionMode>
</infod:DropConsumerEntry>
```

The elements of the DropConsumerEntry message are further described as follows:

/infod:ConsumerEntryReference

An endpoint reference element, as defined by WS-Addressing, used to identify the INFOD resource in the registry to drop.

/infod:ExecutionMode

<x-concern id=X415 part=3/8>

A parameter indicating the mode of execution of the drop request. Possible values are:

"IF UNUSED"	The drop request will execute only if the resource is unreferenced
"DISABLE NEW"	No new references are possible for the resource. The resource will be dropped when the last reference to this resource is gone
"CASCADE"	The drop request will execute immediately and all references to the resource will be removed recursively

If this parameter is not specified, the default value "IF UNUSED" MUST be used.

</x-concern id=X415>

<x-concern id=X418 part=9/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/DropConsumerEntry MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the DropConsumerEntry message, it MUST respond to the WS endpoint specified in the request message with a DropConsumerResponseEntry message. The DropConsumerEntry response message is a message of the following form:

```
<infod:DropConsumerEntryResponse>
  <infod:Status>
 xsd:string default ”COMPLETED”
  </infod:Status>
</infod:DropConsumerEntryResponse>
```

The elements of the DropConsumerResponseEntry message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X411 part=3/8>

- DropResourceAuthorizationFailure: User not authorized to drop the INFOD resource at this INFOD registry

</x-concern id=X411>

<x-concern id=X410 part=6/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=9/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X412 part=3/8>

- ExecutionModeFault: Cannot use ExecutionMode provided

</x-concern id=X412>

<x-concern id=X405 part=9/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.4 Managing Subscriptions

The following operations are used to manage subscriptions:

- CreateSubscription (section 2.4.1)
- ReplaceSubscription (section 2.4.2)

- DropSubscription (section 2.4.3)

2.4.1 CreateSubscription

The CreateSubscription operation is used by a subscriber, to create an INFOD subscription in an INFOD registry.

This subscription resource is responsible to describe the conditions of interest of potential consumers for potential publishers.

<x-concern id=X401 part=4/8>

As part of the processing of a CreateSubscription request message, the INFOD registry MUST create an INFOD resource representing the subscription.

</x-concern id=X401>

The format of the request message for CreateSubscription operation is based on the schema provided in Appendix 1 for an INFOD resource. Details are as follows:

```
<infod:CreateSubscription>
  <infod:SubscriptionName> xsd:string </infod:SubscriptionName> ?
  <infod:SubscriptionDescription>
    xsd:string
  </infod:SubscriptionDescription> ?
  <infod:SubscriberEntryReference>
    wsa:EndPointReferenceType
  </infod:SubscriberEntryReference>
  <infod:DataConstraint >
    xsd:anyType
  </infod:DataConstraint> *
  <infod:PropertyConstraint>
    xsd:any
  </infod:PropertyConstraint> *
  <infod:DynamicConsumerConstraint>
    xsd:anyType
  </infod:DynamicConsumerConstraint> *
</infod:CreateSubscription>
```

The elements of the CreateSubscription message are further described as follows:

/infod:SubscriptionName

A string representing the name for the subscription. This name MAY NOT be unique.

/infod:SubscriptionDescription

A string representing a description of the subscription.

/infod:SubscriberEntryReference

An endpoint reference element to the INFOD EPR, as defined by WS-Addressing, used to identify the subscriber entry responsible for the subscription.

/infod:DataConstraint

DataConstraint specifies which information is of interest to consumers. The constraint(s) language(s) is/are implicitly defined through the reference of the vocabulary EPR. Data Constraints are not applied by the INFOD registry but by the publishers.

See 2.5 for more details on how to define a vocabulary referenced by such constraints.

/infod:PropertyConstraint

Property contraints are used to specify which conditions must be safisfied by entries (publishers, data sources, and consumers) to be eligible for this subsription.

<x-concern id=X403 part=7/9>

A property constraint MUST be formulated as an XQuery. The INFOD Base Use Case Scenarios (see http://forge.gridforum.org/sf/go/doc13626?nav=1) provide examples of XQueries.

For example, a subscription identifies the set of publishers that are eligible to react to this subscription.

Note that the XQuery statement MUST be encoded correctly, i.e. characters such as ">" would be represented as ">"

</x-concern id=X403>

/infod:DynamicConsumerConstraint

An element specifying which consumers receive a specific message. The constraint(s) language(s) is/are implicitly defined through the reference of the vocabulary EPR.

These Constraints are designed to determine the consumers of each message based on its content; i.e., a Dynamic Consumer Constraint cannot be applied by the INFOD registry and is processed by the publishers.

infod:PropertyConstraint should be used to specify consumer constraints if all messages created in response to the subscription are published to the same set of consumers.

For example, a message representing a bill should be *published* to the payee.

<x-concern id=X418 part=10/21>

A WS-Addressing Action header with the value
http://www.ogf.org/infod/INFODRegistry/CreateSubscription MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the CreateSubscription message, it MUST respond to the WS endpoint specified in the request message with a CreateSubscriptionResponse message. The CreateSubscription response message is a message of the following form:

```
<infod:CreateSubscriptionResponse>
  <infod:SubscriptionReference>
    wsa:EndPointReferenceType
  </infod:SubscriptionReference>
</infod:CreateSubscriptionResponse>
```

The elements of the CreateSubscriptionResponse message are further described as follows:

/infod:SubscriptionReference

An endpoint reference element, as defined by WS-Addressing, used to identify the newly created subscription in the INFOD registry.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X406 part=4/8>

- CreateResourceAuthorizationFault: User not authorized to create the INFOD resource at this INFOD registry

</x-concern id=X406>

<x-concern id=X410 part=7/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=10/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X408 part=7/10>

- UnsupportedXQueryFault: The XQuery specified could not be parsed correctly

</x-concern id=X408>

<x-concern id=X405 part=10/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.4.2 ReplaceSubscription

<x-concern id=X414 part=4/4>

As part of the processing of a ReplaceSubscription request message, the INFOD registry MUST replace the entire INFOD metadata for the resource representing the subscription. All previously defined values MUST be deleted. The ReplaceSubscription differs from the CreateSubscription interface in that it replaces an existing subscription resource and assigns the original EPR to the replaced subscription.

</x-concern id=X414>

The format of the request message for a ReplaceSubscription operation is also based on the schema definition provided in Appendix 1 for an INFOD resource. Details are as follows:

```
<infod:ReplaceSubscription>
  <infod:SubscriptionReference>
    wsa:EndPointReferenceType
  </infod:SubscriptionReference>
  <infod:SubscriptionName> xsd:string </infod:SubscriptionName> ?
  <infod:SubscriptionDescription>
    xsd:string
  </infod:SubscriptionDescription> ?
  <infod:SubscriberReference>
    wsa:EndPointReferenceType
  </infod:SubscriberReference>
  <infod:DataConstraint>
    xsd:anyType
  </infod:DataConstraint> *
  <infod:PropertyConstraint>
    xsd:any
```

```
  </infod:PropertyConstraint> *
  <infod:DynamicConsumerConstraint>
    xsd:anyType
  </infod:DynamicConsumerConstraint> *
</infod:ReplaceSubscription>
```

The elements of the ReplaceSubscription message are further described as follows:

/infod:SubscriptionReference

An endpoint reference element, as defined by WS-Addressing, used to identify the subscription resource in the INFOD registry that will be replaced.

/infod:SubscriptionName

A string representing the name of the subscription. This name MAY NOT be unique.

/infod:SubscriptionDescription

A string representing a description of the subscription.

/infod:SubscriberEntryReference

An endpoint reference element to the INFOD EPR, as defined by WS-Addressing, used to identify the subscriber entry responsible for the subscription.

/infod:DataConstraint

DataConstraint specifies which information is of interest to consumers. The constraint(s) language(s) is/are implicitly defined through the reference of the vocabulary EPR. Data Constraints are not applied by the INFOD registry but by the publishers.

See 2.5 for more details on how to define a vocabulary referenced by such constraints.

Note: If no data constraint is specified all messages published by publishers are of interest.

/infod:PropertyConstraint

Property contraints are used to specify which conditions must be safisfied by entries (publishers, data sources, and consumers) to be eligible for this subsription.

<x-concern id=X403 part=8/9>

A property constraint MUST be formulated as an XQuery. The INFOD Base Use Case Scenarios (see http://forge.gridforum.org/sf/go/doc13626?nav=1) provide examples of XQueries.

For example, a subscription identifies the set of publishers that are eligible to react to this subscription.

Note that the XQuery statement MUST be encoded correctly, i.e. characters such as “>” would be represented as “>”

</x-concern id=X403>

/infod:DynamicConsumerConstraint

An element specifying which consumers receive a specific message. The constraint(s) language(s) is/are implicitly defined through the reference of the vocabulary EPR.

These Constraints are designed to determine the consumers of each message based on its content; i.e., a Dynamic Consumer Constraint cannot be applied by the INFOD registry and is processed by the publishers.

infod:PropertyConstraint should be used to specify consumer constraints if all messages created in response to the subscription are disseminated to the same set of consumers.

For example, a message representing a bill should be disseminated to the payee.

<x-concern id=X418 part=11/21>

A WS-Addressing Action header with the value
http://www.ogf.org/infod/INFODRegistry/ReplaceSubscription MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the ReplaceSubscriptionRequest, it MUST respond to the WS endpoint specified in the request message with a ReplaceSubscription message. The ReplaceSubscription response message is a message of the following form:

```
<infod:ReplaceSubscriptionResponse>
  <infod:Status>
 xsd:string default ”COMPLETED”
  </infod:Status>
</infod:ReplaceSubscriptionResponse>
```

The elements of the ReplaceSubscriptionResponse message are further described as follows:

/infod:SubscriptionReference

An endpoint reference element, as defined by WS-Addressing, used to identify the subscription resource in the INFOD registry to replace.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X409 part=4/4>

- ReplaceResourceAuthorizationFault: User not authorized to replace the INFOD resource at this INFOD registry

</x-concern id=X409>

<x-concern id=X410 part=8/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=11/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X408 part=8/10>

- UnsupportedXQueryFault: The XQuery specified could not be parsed correctly

</x-concern id=X408>

<x-concern id=X405 part=11/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.4.3 DropSubscription

The DropSubscription operation MUST be used to remove an INFOD subscription resource from an INFOD registry.

The format of the request message for a DropSubscription operation is:

```
<infod:DropSubscription>
  <infod:SubscriptionReference>
     wsa:EndPointReferenceType
  </infod:SubscriptionReference>
  <infod:ExecutionMode> xsd:string </infod:ExecutionMode>
</infod:DropSubscription>
```

The elements of the DropSubscription message are further described as follows:

/infod:SubscriptionReference

An endpoint reference element, as defined by WS-Addressing, used to identify the INFOD subscription resource in the registry to drop.

/infod:ExecutionMode

<x-concern id=X415 part=4/4>

An optional parameter indicating the mode of execution of the drop request. Possible values are:

"IF UNUSED"	The drop request will execute only if the resource is unreferenced
"DISABLE NEW"	No new references are possible for the resource. The resource will be dropped when the last reference to this resource is gone
"CASCADE"	The drop request will execute immediately and all references to the resource will be removed recursively

If this parameter is not specified, the default value "IF UNUSED" MUST be used.

</x-concern id=X415>

<x-concern id=X418 part=12/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/DropSubscription MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the DropSubscription request, it MUST respond to the WS endpoint specified in the request message with a DropSubscriptionResponse message. The DropSubscriptionResponse message is a message of the following form:

```
<infod:DropSubscriptionResponse>
  <infod:Status>
```

```
 xsd:string default ”COMPLETED”
  </infod:Status>
</infod:DropSubscriptionResponse>
```

The elements of the ReplaceSubscriptionResponse message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X411 part=4/8>

- DropResourceAuthorizationFailure: User not authorized to drop the INFOD resource at this INFOD registry

</x-concern id=X411>

- UnknownElementReferenceFault: An element has been referenced that is unknown to the INFOD registry

<x-concern id=X407 part=12/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X412 part=4/8>

- ExecutionModeFault: Cannot use ExecutionMode provided

</x-concern id=X412>

<x-concern id=X405 part=12/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.5 Managing Vocabularies

INFOD has a set of predefined vocabularies. These are REQUIRED vocabularies for the INFOD registry:

- INFOD PublisherEntry Vocabulary
- INFOD SubscriberEntry Vocabulary
- INFOD ConsumerEntry Vocabulary
- INFOD Subscription Vocabulary
- INFOD DataSourceEntry Vocabulary

These vocabularies are used by the INFOD registry to match publishers with consumers through subscriptions and ensure that property constraints and data constraints are validated. All of these vocabularies are described in xml and detailed in section **Error! Reference source not found.**

Users MAY also define two additional types of vocabularies:

Property Vocabularies: Entries may specify properties that define their characteristics. They do that using a property vocabulary that may be queried. If two or more entries share the same property vocabulary, they can specify constraints on each other. The INFOD registry MAY manage constraints on these property vocabularies in addition to constraints formulated in the INFOD vocabularies. Property Vocabularies MUST be defined in xml.

Data Vocabularies**:** In order to tell publishers which messages a subscription is interested in, they MUST agree on the data vocabulary. The data vocabulary is referenced in the *DataConstraints* component of a subscription resource, which allows INFOD subscribers to describe the structure of the published data/data of interest to them.

Data constraints' definitions MUST point to an existing data vocabulary and thus are simply equivalent to defining operations on top of an existing vocabulary (i.e. selection criteria, etc. on top of published data). Data Vocabularies are not limited to xml.

This section describes how these two types of vocabulary are created and dropped from an INFOD registry. It also includes operations for creating and dropping instances of a registered property vocabulary.

2.5.1 CreatePropertyVocabulary

The CreatePropertyVocabulary creates a property vocabulary in an INFOD registry. The Property Vocabulary is an XML schema.

<x-concern id=X401 part=5/8>

As part of the processing of a CreatePropertyVocabulary request message, the INFOD registry MUST create a new resource for that vocabulary.

</x-concern id=X401>

The format of the request message for CreatePropertyVocabulary operation is as follows:

```
<infod:CreatePropertyVocabulary>
  <infod:PropertyVocabularyName>
    xsd:string
  </infod:PropertyVocabularyName> ?
  <infod:PropertyVocabularyDescription>
    xsd:string
  </infod:PropertyVocabularyDescription> ?
  <infod:PropertyVocabularyBody>
    xsd:schema
  </infod:PropertyVocabularyBody>
</infod:CreatePropertyVocabulary>
```

The elements of the CreatePropertyVocabulary message are further described as follows:

/infod:PropertyVocabularyName

A string representing a name that is local to the INFOD registry where the CreatePropertyVocabulary operation takes place. This name MAY NOT be unique.

Names MUST NOT start with $$infod.

/infod:PropertyVocabularyDescription

A string representing a description of the vocabulary.

/infod:PropertyVocabularyBody

1077 An element defining an XML Schema. This is an extensibility mechanism to allow XML
1078 elements to be specified for the defined property vocabulary.

<x-concern id=X418 part=13/21>

1079 A WS-Addressing Action header with the value
1080 http://www.ogf.org/infod/INFODRegistry/CreatePropertyVocabulary MUST accompany the message.

</x-concern id=X418>

1081 **INFOD Registry Response**

1082 If the INFOD registry accepts the CreatePropertyVocabulary request, it MUST respond to the WS
1083 endpoint specified in the request message with a CreateVocabularyResponse message.

1084 In case of a successful registration, the CreateVocabularyResponse message is a message of the
1085 following form:

```
1086 <infod:CreatePropertyVocabularyResponse>
1087     <infod:PropertyVocabularyReference>
1088        wsa:EndPointReferenceType
1089     </infod:PropertyVocabularyReference>
1090 </infod:CreateVocabularyResponse>
```

1091 The elements of the CreateVocabularyResponse message are further described as follows:

1092 /infod:PropertyVocabularyReference

1093 An endpoint reference element, as defined by WS-Addressing, used to identify the newly
1094 created property vocabulary.

1095 One of the following faults MUST be sent if the operation fails::

<x-concern id=X406 part=5/8>

1096 • CreateResourceAuthorizationFault: User not authorized to create a resource at this
1097 INFOD registry

</x-concern id=X406>

<x-concern id=X407 part=13/21>

1098 • MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X413 part=1/3>

1099 • UnSupportedVocabularyFault: Vocabulary Language not supported

</x-concern id=X413>

<x-concern id=X405 part=13/21>

1100 The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP
1101 v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.5.2 DropPropertyVocabulary

The DropPropertyVocabulary operation drops a particular property vocabulary from an INFOD registry.

The format of the request message for an DropPropertyVocabulary operation is:

```
<infod:DropPropertyVocabulary>
  <infod:PropertyVocabularyReference>
    wsa:EndPointReferenceType
  </infod:PropertyVocabularyReference
  <infod:ExecutionMode> xsd:string </infod:ExecutionMode>
</infod:DropPropertyVocabulary>
```

The elements of the DropPropertyVocabulary message are further described as follows:

/infod: PropertyVocabularyReference

An endpoint reference element, as defined by WS-Addressing, used to identify the vocabulary to drop from the Registry.

/infod:ExecutionMode

<x-concern id=X415 part=5/8>

A parameter indicating the mode of execution of the drop request. Possible values are:

"IF UNUSED"	The drop request will execute only if the resource is unreferenced
"DISABLE NEW"	No new references are possible for the resource. The resource will be dropped when the last reference to this resource is gone
"CASCADE"	The drop request will execute immediately and all references to the resource will be removed recursively

If this parameter is not specified, the default value "IF UNUSED" MUST be used.

</x-concern id=X415>

<x-concern id=X418 part=14/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/DropPropertyVocabulary MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the DropPropertyVocabulary request, it MUST respond to the WS endpoint specified in the request message with an DropPropertyVocabularyResponse message. The DropPropertyVocabulary response message is a message of the following form:

```
<infod:DropPropertyVocabularyResponse>
  <infod:Status>
 xsd:string default "COMPLETED"
  </infod:Status>
</infod:DropPropertyVocabularyResponse>
```

The elements of the DropPropertyVocabularyResponse message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

1138 One of the following faults MUST be sent if the operation fails:

<x-concern id=X411 part=5/8>

1139 • DropResourceAuthorizationFailure: User not authorized to drop the resource at this
1140 INFOD registry

</x-concern id=X411>

<x-concern id=X410 part=9/15>

1141 • UnknownResourceReferenceFault: An element has been referenced that is unknown to
1142 the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=14/21>

1143 • MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X412 part=5/8>

1144 • ExecutionModeFault: Cannot use ExecutionMode provided

</x-concern id=X412>

<x-concern id=X405 part=14/21>

1145 The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP
1146 v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

1147 2.5.3 CreatePropertyVocabularyInstance

1148 The CreatePropertyVocabularyInstance operation creates a new instance of a particular property
1149 vocabulary previously created in the INFOD registry. An instance of a property vocabulary fills in
1150 values into the vocabulary structure defined by the Property Vocabulary (section 2.5.1) and relates a
1151 particular INFOD entry to the instance. The referenced entry is now identified to use the property
1152 vocabulary.

<x-concern id=X401 part=6/8>

1153 As part of the processing of a CreatePropertyVocabularyInstance request message, the INFOD
1154 registry MUST create a new instance for that vocabulary.

</x-concern id=X401>

1155 The format of the request message for CreatePropertyVocabularyInstance operation is as follows:

```
1156 <infod:CreatePropertyVocabularyInstance>
1157   <infod:EntryReference>
1158     wsa:EndPointReferenceType
1159   </infod:EntryReference>
1160   <infod:PropertyVocabularyReference>
1161     wsa:EndPointReferenceType
1162   </infod:PropertyVocabularyReference>
1163   <infod:PropertyVocabularyInstanceBody>
1164     {xsd:anyType} ?
1165   </infod:PropertyVocabularyInstanceBody>
```

```
</infod:CreatePropertyVocabularyInstance>
```

The elements of the CreatePropertyVocabularyInstance message are further described as follows:

/infod:EntryReference

EPR of the INFOD entry that the instance of the property vocabulary will be identified with.

/infod:PropertyVocabularyReference

EPR of a vocabulary that will be referenced to the INFOD resource.

/infod:PropertyVocabularyInstanceBody

An element that contains specific instance information that needs to match the structure of the vocabulary defined in VocabularyReference.

<x-concern id=X418 part=15/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/CreatePropertyVocabularyInstance MUST accompany the message.

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the CreatePropertyVocabularyInstance request, it MUST respond to the WS endpoint specified in the request message with a CreatePropertyVocabularyInstance response message.

The CreatePropertyVocabularyInstanceResponse message is a message of the following form:

```
<infod:CreatePropertyVocabularyInstanceResponse>
  <infod:PropertyVocabularyInstanceReference>
    wsa:EndPointReferenceType
  </infod:PropertyVocabularyInstanceReference>
</infod:CreatePropertyVocabularyInstanceResponse>
```

The elements of the CreatePropertyVocabularyInstanceResponse message are further described as follows:

/infod:PropertyVocabularyInstanceReference

An endpoint reference element, as defined by WS-Addressing, used to identify the newly created vocabulary instance.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X406 part=6/8>

- CreateResourceAuthorizationFault: User not authorized to create the INFOD resource at this INFOD registry

</x-concern id=X406>

<x-concern id=X410 part=10/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

<x-concern id=X410>

<x-concern id=X407 part=15/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X413 part=2/3>

- UnSupportedVocabularyFault: Vocabulary Language not supported

</x-concern id=X413>

<x-concern id=X405 part=15/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf) .

</x-concern id=X405>

2.5.4 DropPropertyVocabularyInstance

The DropPropertyVocabularyInstance operation drops an existing instance of a particular property vocabulary previously created in the INFOD registry.

The format of the request message for a DropPropertyVocabularyInstance operation is:

```
<infod:DropPropertyVocabularyInstance>
  <infod:PropertyVocabularyInstanceReference>
     wsa:EndPointReferenceType
  </infod:PropertyVocabularyInstanceReference>
  <infod:ExecutionMode> xsd:string </infod:ExecutionMode>
</infod:DropPropertyVocabularyInstance>
```

The elements of the DropPropertyVocabularyInstance message are further described as follows:

/infod:PropertyVocabularyInstanceReference

An endpoint reference element, as defined by WS-Addressing, used to identify the property vocabulary instance to drop from the Registry.

/infod:ExecutionMode

<x-concern id=X415 part=6/8>

A parameter indicating the mode of execution of the drop request. Possible values are:

"IF UNUSED"	The drop request will execute only if the resource is unreferenced
"DISABLE NEW"	No new references are possible for the resource. The resource will be dropped when the last reference to this resource is gone
"CASCADE"	The drop request will execute immediately and all references to the resource will be removed recursively

If this parameter is not specified, the default value "IF UNUSED" MUST be used.

</x-concern id=X415>

<x-concern id=X418 part=16/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/DropVocabularyInstance MUST accompany the message.

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the DropPropertyVocabularyInstance request, it MUST respond to the WS endpoint specified in the request message with a DropPropertyVocabularyInstanceResponse message in the following form:

```
<infod:DropPropertyVocabularyInstanceResponse>
  <infod:Status>
 xsd:string default ”COMPLETED”
  </infod:Status>
</infod:DropPropertyVocabularyInstanceResponse>
```

The elements of the DropPropertyVocabularyInstanceResponse message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X411 part=6/8>

- DropResourceAuthorizationFailure: User not authorized to drop the resource at this INFOD registry

</x-concern id=X411>

<x-concern id=X410 part=11/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=16/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X412 part=6/8>

- ExecutionModeFault: Cannot use ExecutionMode provided

</x-concern id=X412>

<x-concern id=X405 part=16/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.5.5 CreateDataVocabulary

<x-concern id=X401 part=7/8>

As part of the processing of a CreateDataVocabulary request message, the INFOD registry MUST create a new resource for that vocabulary.

</x-concern id=X401>

The format of the request message for CreateDataVocabulary operation is:

```
<infod:CreateDataVocabulary>
  <infod:DataVocabularyName> xsd:string </infod:DataVocabularyName> ?
  <infod:DataVocabularyDescription>
    xsd:string
  </infod:DataVocabularyDescription> ?
  <infod:DataVocabularyLanguage>
    {anyURI} (Namespace/URI of DataFormat)
  </infod:DataVocabularyLanguage>
  <infod:LanguageUsageDescription>
    xsd:string
  </infod:LanguageUsageDescription> ?
  <infod:DataVocabularyBody>
    xsd:anyType
  </infod:DataVocabularyBody>
</infod:CreateDataVocabulary>
```

The elements of the CreateDataVocabulary message are further described as follows:

/infod:DataVocabularyName

A string.representing a name in the INFOD registry where the CreateDataVocabulary operation takes place. This name MAY NOT be unique.

Names MUST NOT start with $$infod.

/infod:DataVocabularyDescription

A string representing a description of the vocabulary.

/infod:DataVocabularyLanguage

A URI defining the format of the data vocabulary.

/infod:DataVocabularyBody

A string representing a data vocabulary.

This embedded string represents the vocabulary and MUST be encoded correctly as defined through the DataVocabularyLanguage definition (escape characters etc.)

<x-concern id=X418 part=17/21>

A WS-Addressing Action header with the value
http://www.ogf.org/infod/INFODRegistry/CreateDataVocabulary MUST accompany the message.

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the CreateDataVocabulary request, it MUST respond to the WS endpoint specified in the request message with a CreateVocabularyResponse message.

In case of a successful registration, the CreateVocabularyResponse message is a message of the following form:

```
<infod:CreateDataVocabularyResponse>
  <infod:DataVocabularyReference>
    wsa:EndPointReferenceType
  </infod:DataVocabularyReference>
```

```
</infod:CreateDataVocabularyResponse>
```

The elements of the CreateVocabularyResponse message are further described as follows:

/infod:DataVocabularyReference

An endpoint reference element, as defined by WS-Addressing, used to identify the newly created vocabulary.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X406 part=7/8>

- CreateResourceAuthorizationFault: User not authorized to create a resource at this INFOD registry

</x-concern id=X406>

<x-concern id=X410 part=12/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=17/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X413 part=3/3>

- UnSupportedVocabularyFault: Vocabulary Language not supported

</x-concern id=X413>

<x-concern id=X405 part=17/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.5.6 DropDataVocabulary

The DropDataVocabulary operation drops a particular data vocabulary from an INFOD registry.

The format of the request message for an DropDataVocabulary operation is:

```
<infod:DropDataVocabulary>
  <infod:DataVocabularyReference>
    wsa:EndPointReferenceType
  </infod:DataVocabularyReference
  <infod:ExecutionMode> xsd:string </infod:ExecutionMode>
</infod:DropDataVocabulary>
```

The elements of the DropDataVocabulary message are further described as follows:

/infod: DataVocabularyReference

An endpoint reference element, as defined by WS-Addressing, used to identify the vocabulary to drop from the Registry.

/infod:ExecutionMode

<x-concern id=X415 part=7/8>

A parameter indicating the mode of execution of the drop request. Possible values are:

"IF UNUSED"	The drop request will execute only if the resource is unreferenced
"DISABLE NEW"	No new references are possible for the resource. The resource will be dropped when the last reference to this resource is gone
"CASCADE"	The drop request will execute immediately and all references to the resource will be removed recursively

If this parameter is not specified, the default value "IF UNUSED" MUST be used.

</x-concern id=X415>

<x-concern id=X418 part=18/21>

A WS-Addressing Action header with the value
http://www.ogf.org/infod/INFODRegistry/DropDataVocabulary MUST accompany the message

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the DropDataVocabulary request, it MUST respond to the WS endpoint specified in the request message with a DropDataVocabularyResponse message. The DropDataVocabulary response message is a message of the following form:

```
<infod:DropDataVocabularyResponse>
  <infod:Status>
 xsd:string default "COMPLETED"
  </infod:Status>
</infod:DropDataVocabularyResponse>
```

The elements of the DropDataVocabularyResponse message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X411 part=7/8>

- DropResourceAuthorizationFailure: User not authorized to drop the resource at this INFOD registry

</x-concern id=X411>

<x-concern id=X410 part=13/15>

- UnknownResourceReferenceFault: An element has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=18/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X412 part=7/8>

- ExecutionModeFault: Cannot use ExecutionMode provided

</x-concern id=X412>

<x-concern id=X405 part=18/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.6 Data Source Entries

A Data Source Entry relates a publisher entry with a data vocabulary.

The following operations are used to manage data sources:

- CreateDataSourceEntry (section 2.6.1)
- DropDataSourceEntry (section 2.6.2)

2.6.1 CreateDataSourceEntry

The CreateDataSourceEntry operation creates a relation between an INFOD publisher entry and a data vocabulary at the INFOD registry.

<x-concern id=X401 part=8/8>

As part of the processing of an CreateDataSourceEntry operation message, the INFOD registry MUST create an INFOD vocabulary association resource.

</x-concern id=X401>

The format of the request message for an CreateDataSourceEntry operation is:

```
<infod:CreateDataSourceEntry>
  <infod:DataSourceEntryName> ?
    xsd:string
  </infod:DataSourceEntryName>
  <infod:DataSourceEntryDescription>
     xsd:string
  </infod:DataSourceEntryDescription> ?
  <infod:PublisherEntryReference>
     wsa:EndPointReferenceType
  </infod:PublisherEntryReference>
  <infod:DataVocabularyReference>
     wsa:EndPointReferenceType
  </infod:DataVocabularyReference> +
  <infod:PropertyConstraint>
    xsd:any
  </infod:PropertyConstraint> *
</infod:CreateDataSourceEntry>
```

The elements of the CreateDataSourceEntry message are further described as follows:

/infod:DataSourceEntryName

A string representing the name of the data source entry. This name MAY NOT be unique.

/infod:DataSourceEntryDescription

A string representing a description of the data source entry.

/infod:PublisherEntryReference

The EPR of the publisher entry for which a data source entry is created.

/infod:DataVocabularyReference

The EPR(s) of a vocabulary with which to associate the publisher entry.

/infod:PropertyConstraint

Property contraints are used to specify which conditions must be safisfied by entries (subscribers and consumers) to be eligible to receive data from this data source.

<x-concern id=X403 part=9/9>

A property constraint MUST be formulated as an XQuery. The INFOD Base Use Case Scenarios (see http://forge.gridforum.org/sf/go/doc13626?nav=1) provide examples of XQueries.

For example, a data sources identifies the set of consumers that are eligible to receive data from this data source.

Note that the XQuery statement MUST be encoded correctly, i.e. characters such as “>” would be represented as “>”

</x-concern id=X403>

<x-concern id=X418 part=19/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/CreateDataSourceEntry MUST accompany the message.

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the CreateDataSourceEntry request, it MUST respond to the WS endpoint specified in the request message with a CreateDataSourceEntryResponse message.

The CreateDataSourceEntryResponse message is a message of the following form:

```
<infod:CreateDataSourceEntryResponse>
  <infod:DataSourceEntryReference>
    wsa:EndPointReferenceType
  </infod:DataSourceEntryReference>
</infod:CreateDataSourceEntryResponse>
```

The elements of the response message are further described as follows:

/infod:DataSourceEntryReference

An endpoint reference element, as defined by WS-Addressing, used to identify the newly created vocabulary association.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X406 part=8/8>

- CreateResourceAuthorizationFault: User not authorized to create the resource at this INFOD registry

</x-concern id=X406>

<x-concern id=X410 part=14/15>

- UnknownResourceReferenceFault: An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=19/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X408 part=9/10>

- UnsupportedXQueryFault: The XQuery specified could not be parsed correctly

</x-concern id=X408>

<x-concern id=X405 part=19/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.6.2 DropDataSourceEntry

The DropDataSourceEntry operation drops a data source entry from an INFOD registry.

The format of the request message for a DropDataSourceEntry operation is:

```
<infod:DropDataSourceEntry>
  <infod:DataSourceEntryReference>
    wsa:EndPointReferenceType
  </infod:DataSourceEntryReference>
  <infod:ExecutionMode> xsd:string </infod:ExecutionMode>
</infod:DropDataSourceEntry>
```

The elements of the DropDataSourceEntry message are further described as follows:

/infod: DataSourceEntryReference

An endpoint reference element, as defined by WS-Addressing, used to identify the association to drop from the Registry.

/infod:ExecutionMode

<x-concern id=X415 part=8/8>

A parameter indicating the mode of execution of the drop request. Possible values are:

“IF UNUSED”	The drop request will execute only if the resource is unreferenced
“DISABLE NEW”	No new references are possible for the resource. The resource will be dropped when the last reference to this resource is gone

“CASCADE”	The drop request will execute immediately and all references to the resource will be removed recursively

If this parameter is not specified, the default value “IF UNUSED” MUST be used.

</x-concern id=X415>

<x-concern id=X418 part=20/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/DropDataSourceEntry MUST accompany the message.

</x-concern id=X418>

INFOD Registry Response

If the INFOD registry accepts the DropDataSourceEntry request, it MUST respond to the WS endpoint specified in the request message with a DropDataSourceEntryResponse message. The DisCreateDataSourceEntryResponse message is a message of the following form:

```
<infod:DropDataSourceEntryResponse>
  <infod:Status>
 xsd:string default ”COMPLETED”
  </infod:Status>
</infod:DropDataSourceEntryResponse>
```

The elements of the DropDataSourceEntryResponse message are further described as follows:

/infod:Status

An indication that the request has been successfully executed.

One of the following faults MUST be sent if the operation fails:

<x-concern id=X411 part=8/8>

• DropResourceAuthorizationFailure:	User not authorized to drop the resource at this INFOD registry

</x-concern id=X411>

<x-concern id=X410 part=15/15>

• UnknownResourceReferenceFault:	An resource has been referenced that is unknown to the INFOD registry

</x-concern id=X410>

<x-concern id=X407 part=20/21>

• MissingRequiredParameterFault:	A required parameter was not specified

</x-concern id=X407>

<x-concern id=X412 part=8/8>

• ExecutionModeFault:	Cannot use ExecutionMode provided

</x-concern id=X412>

<x-concern id=X405 part=20/21>

The message MUST be sent using the WS-Base Faults. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

2.7 The GetMetaData Operation

The Base Meta Data Access interface provides access to data contained in an INFOD registry. The request is formulated as an XQuery and the result is returned according to the specification in the return clause of the XQuery.

The format of the request message for a GetMetadata operation is:

```
<infod:GetMetaData>
  <infod:MetaDataQueryExpression>
 {xsd:anyType}
  </infod:MetaDataQueryExpression>
</infod:GetMetadata>
```

The elements of the GetMetadata message are further described as follows:

/infod:MetaDataQueryExpression

The element MUST be a valid XQuery or an XPath expression for the INFOD registry.

The INFOD registry is fully qualified by an INFOD registry service name appended to the string “INFODRegistry.xml”. A fully qualified name allows the registry instance to be referenced uniquely.

An example for a fully qualified INFOD registry is: http://www.w3c.org/OGF/INFOD/Instance/INFODRegistry.xml.

The INFOD registry service name need not be hard coded into the XQuery fn:doc but could be specified by setting the base-URI to be the service name e.g. declare base-URI "http://www.w3c.org/OGF/INFOD/Instance/INFODRegistry.xml". This indirection allows us to specify specific a registry amongst many in a given environment.

Default XPath expresssions:

In addition to supporting user defined Xpath/XQuery expressions, INFOD reserves the following paths and mandates their implementation.

- **All publishers** - fn:doc('INFODRegistry.xml')/publishers/$$infodPublisher
- **All subscribers** - fn:doc('INFODRegistry.xml')/subscribers/$$infodSubscriber
- **All consumers** - fn:doc('INFODRegistry.xml')/consumers/$$infodConsumer
- **All subscriptions** - fn:doc('INFODRegistry.xml')/subscriptions/$$infodSubscription
- **All property vocabularies** - fn:doc('INFODRegistry.xml')/propertyvocabularies/$$infodPropertyVocabulary
- **All property vocabulary instances** - fn:doc('INFODRegistry.xml')/propertyvocabularyinstances/$$infodPropertyVocabularyInstance
- **All data vocabulary** - fn:doc('INFODRegistry.xml')/datavocabularies/$$infodDataVocabulary

<x-concern id=X418 part=21/21>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODRegistry/GetMetaData MUST accompany the message.

</x-concern id=X418>

INFOD Registry Response

The response of the INFOD registry is:

```
<infod:GetMetaDataQueryResponse>
  <infod:MetaDataQueryResult>
    {xsd:anyType}
  <infod:GetMetaDataQueryResult>
<infod:MetaDataQueryResponse>
```

The content of infod:GetMetaDataQueryResult MUST be structured according to the return specification in the GetMetaData request.

One of the following faults MUST be sent if the operation fails:

- GetMetaDataAuthorizationFailure: User not authorized to use the operation at this INFOD registry

<x-concern id=X407 part=21/21>

- MissingRequiredParameterFault: A required parameter was not specified

</x-concern id=X407>

<x-concern id=X408 part=10/10>

- UnsupportedXQueryFault: The XQuery specified could not be parsed correctly

</x-concern id=X408>

<x-concern id=X405 part=21/21>

The message MUST be structured according to the WS-Base Faults specification. For examples using SOAP, see the SOAP v1.2. Base Fault Spec (see http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf).

</x-concern id=X405>

3 Base INFOD Notification Interfaces

We divide notifications between INFOD components into two major categories: notifications from publishers to consumers which carry the actual data, and notifications from the registry to publishers, subscribers, and consumers which contain information about relevant state changes in the registry. INFOD does not use the WSN notify interface due to different header requirements.

3.1 Notifications from Publishers to Consumers

<x-concern id=X417>

An INFOD publisher uses a Notify operation similar to that defined by WS-Notification to send messages to an INFOD consumer (see http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.3-draft-01.pdf).

</x-concern id=X417>

The following xml describes the format of an INFOD Notify message:

```
<infod:Notify>
  <infod:NotificationMessage>
    <infod:SubscriptionReference>
      wsa:EndpointReferenceType
    </infod:SubscriptionReference> ?
    <infod:Topic Dialect="xsd:anyURI">
      {any} ?
    </infod:Topic>?
    <infod:PublisherReference>
      wsa:EndpointReferenceType
    </infod:PublisherReference> ?
    <infod:Message>
      {any}
    </infod:Message>
  </infod:NotificationMessage> +
  {any} *
</infod:Notify>
```

The components of the Notify message are further described as follows:

/infod:Notify

Contains a collection of one or more Notifications.

/infod:NotificationMessage

Contains a Notification payload.

/infod:SubscriptionReference

An endpoint reference to the Subscription that is associated with the Notify message.

/infod:Topic

An endpoint reference to the VocabularyAssociation respresenting the source of the payload.

/infod:Topic/@Dialect

An endpoint reference to the vocabulary that was used to structure the payload.

/infod:ProducerReference

An endpoint reference to the Publisher that produced the Notification.

/infod:Message

The actual Notification payload.

/infod:Notify/{any}

The Notify message also allows for open content, in order to accommodate elements that may be needed by extensions built on the WSN BaseNotification (see http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.3-draft-01.pdf), including those providing additional filtering mechanisms.

<x-concern id=X419 part=1/4>

A WS-Addressing Action header with the value http://www.ogf.org/infod/INFODNotify/Notify MUST accompany the message

</x-concern id=X419>

INFOD Registry Response

No response is expected from the INFOD consumer upon receipt of this message.

Example SOAP Encoding of the Notify Message

The following is a non-normative example of a Notify request message using SOAP:

```
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
      http://www.ogf.org/infod/INFODNotify/Notify
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <infod:Notify>
      <infod:NotificationMessage>
        <infod:SubscriptionReference>
          <wsa:Address>
            http://www.example.org/SomeSubscripiton
          </wsa:Address>
        </infod:SubscriptionReference>
        <infod:Topic Dialect=
     "http://www.myinfodregistry.com/infod/MyDataVocabularyEPR">
          infod:DatavocabularyEPR
        </infod:Topic>
        <infod:ProducerReference>
          <wsa:Address>
            http://www.example.org/Publisher
          </wsa:Address>
        </infod:ProducerReference>
        <infod:Message>
         <MyDataVocabulary:MessageContent>MessageDataContent</MyDataVocabul
ary:MessageContent>
        </infod:Message>
```

```
      </infod:NotificationMessage>
    </infod:Notify>

  </s:Body>
</s:Envelope>
```

3.2 Notification from the Registry

The registry sends notifications to those publishers, subscribers and consumers that have registered for them. Changes of state within the registry lead to generation of events. The specifics of the payload and the condition under which a notification MUST be sent are described in the following section:

- Notification of publishers (section 3.2.1)
- Notification of subscribers (section 3.2.2)
- Notification of consumers (section 3.2.3)

3.2.1 Notification of Publishers

The INFOD registry will inform publishers that need to react to changes in the INFOD registry.

<x-concern id=X404 part=7/9>

The notification is conditional on the information in the publisher entry.

</x-concern id=X406>

Publishers SHOULD react immediately to these notifications.

A new publisher MUST be informed about each subscription that requires[4] this publisher to send messages; there will be one notification per subscription.

For existing publishers notifications MUST be sent about those subscriptions that mandate different messages or mandate messages to be sent to different consumers. An empty list of static and dynamic consumers indicates that a publisher MUST stop publishing for the referenced subscription

Notifications are determined by processing the property constraints and the vocabulary reference in the data constraints.

The notification contains the following message body:

```
<infod:PublisherNotification>
  <infod:SubscriptionReference>
    wsa:EndPointReferenceType
  </infod:SubscriptionReference>
  <infod:ConsumerEntryReference>
    wsa:EndPointReferenceType
  <infod:ConsumerEntryReference> *
  <infod:DynamicConsumerConstraint>
    {xsd:anyType}
  <infod:DynamicConsumerConstraint> *
  <infod:DataConstraint>
    {xsd:anyType}
```

[4] Static and dynamic constraints are evaluated to determine if and whether the event notification should be propagated to the recipient.

```
  <infod:DataConstraint> *
<infod:PublisherNotification>
```

The message content is further described as follows:

/infod:SubscriptionReference

This is the EPR of the subscription for which the information is provided.

If all other parameters are omitted the publisher does not need to process this subscription any longer. This EPR is not valid after the subscription is dropped. However, the no longer valid EPR is propagated, as some of the publishers may be using the EPR for their internal references.

/infod:ConsumerEntryReference

This is a list of 0 to n EPR references of consumer entries. The list of consumers is computed by the INFOD Registry and given to each publisher.

/infod:DynamicConsumerConstraint

This is an expression that directs the publisher to determine the consumer(s) based on the listed expressions. Each expression references data that are created by the publishers, e.g. messages to be published, and references properties of INFOD Registry resources.

The subscription should be discarded if there is no entry for StaticConsumers and for DynamicConsumerConstraint.

/infod:DataConstraint

These are the data constraints as specified in the referenced subscription.

<x-concern id=X419 part=2/4>

WS-Addressing of the action MUST contain the URI
http://www.ogf.org/infod/INFODNotify/SubscriptionNotification.

</x-concern id=X419>

3.2.2 Notification of Subscribers

The INFOD registry MUST inform subscriber that need to know the impact of changes in the INFOD registry on their subscriptions; e.g., subscription with an EPR pointing to them.

<x-concern id=X404 part=8/9>

The notification is conditional on the information in the subscriber entry.

</x-concern id=X404>

In reaction to a newly created or replaced subscription the subscriber MUST be informed which publishers send and consumers receive messages based on that subscription.

In reaction to any other change in the INFOD registry the subscriber MUST be informed about those subscription for which the list of publishers or consumers has changed.

Notifications are determined by processing the property constraints and the vocabulary reference in the data constraints.

The notification contains the following message body:

```
<infod:SubscriberNotification.
```

```
<infodSubscriptionReference>
  wsa:endPointReferenceType
</infodSubscriptionReference>
<infod:PublisherEntryReference>
  wsa:endPointReferenceType
</infod:PublisherEntryReference> *
<infod:ConsumerEntryReference>
  wsa:endPointReferenceType
</infod:ConsumerEntryReference> *
<infod:SubscriberNotification
```

The message content is further described as follows:

/infod:SubscriptionReference

This is the EPR of the subscription for which the information is provided

/infod:PublisherEntryReference

This is a list of 0 to n EPR references of publisher entries. The list of publisher entries is computed by the INFOD registry.

Infod:ConsumerEntryReference

This is a list of 0 to n references to static consumers. The list of consumer entries is computed by the INFOD Registry.

<x-concern id=X419 part=3/4>

WS-Addressing of the action MUST contain the URI
http://www.ogf.org/infod/INFODNotify/SubscriptionNotification.

</x-concern id=X419>

3.2.3 Notification of Consumers

The INFOD registry will inform consumers that need to know about changes in the INFOD registry that result in different messages being received or different publishers sending messages.

<x-concern id=X404 part=9/9>

The notification is conditional on the information in the consumer entry.

</x-concern id=X404>

A new consumer MUST be informed about those subscriptions that result in messages being send to this consumer.

An existing consumer MUST be informed about any change in the INFOD registry that adds or removes subscriptions applying to this consumer. The consumer MUST also be notified if the list of publishers of a subscription, already referenced in previous notification to that consumer, has changed.

Notifications are determined by processing the property constraints and the vocabulary reference in the data constraints.

The notification will not be send to dynamic consumers.

The notification contains the following message body:

```
<infod:ConsumerNotification.
<infodSubscriptionReference>
```

```
  wsa:endPointReferenceType
</infodSubscriptionReference>
<infod:PublisherEntryReference>
  wsa:endPointReferenceType
</infod:PublisherEntryReference> *
<infod:ConsumerNotification
```

The message content is further described as follows:

/infod:SubscriptionReference

This is the EPR of the subscription for which the information is provided

/infod:PublisherEntryReference

This is a list of 0 to n EPR references of publisher entries. The list of publisher entries is computed by the INFOD registry.

<x-concern id=X419 part=4/4>

WS-Addressing of the action MUST contain the URI
http://www.ogf.org/infod/INFODNotify/SubscriptionNotification.

</x-concern id=X419>

4 Security Considerations

An INFOD operating environment consists of a set of publishers, consumers and registries. All the above service components operate in different security domains and require “long-term” secure communication of messages. Additionally, as the INFOD services operate in a web services environment, SOAP may be used as the base communication protocol. SOAP based communication between services can be secured by using the mechanisms described by the *WS security* specification (see http://www.oasis-open.org/committees/download.php/5531/oasis-200401-wsssoap-message-security-1.0.pdf). Although, the use of WS-Security provides the mechanisms to accommodate multiple security tokens and encryption technologies, it remains limited to providing a secured point-to-point communication mechanism on a message level. However, INFOD services need to build upon this security mechanism to describe the security context under which they could sustain long running exchanges of messages. A communication session between the two parties such as publisher and consumer serves as the basis for establishing the security context. Establishing a security context between system entries allows secured messaging on the session level and reduces the synchronization overheads required to obtain it on per-message basis. *WS-Secure Conversation* (see ftp://www6.software.ibm.com/software/developer/library/ws-secureconversation.pdf) provides the mechanism for maintaining such long-term contexts for message exchange.

<x-concern id=X313>

The INFOD model **RECOMMENDS** the establishment of the following contexts:

- Publisher – Registry secured context, with Registry as the context security token creator.
- Consumer – Registry secured context, with Registry as the context security token creator.
- Subscriber – Registry secured context, with Registry as the context security token creator.
- Publisher – Consumer secured context, with Publisher as the context security token creator. It may be possible to support registry mediated delegation, where the registry mediates the establishment of trust between producer and consumer.

</x-concern id=X313>

Authentication remains a crucial aspect of formation of a secured conversation. Hence, the specification identifies the objects that create the secured context. It is envisaged that an INFOD-Registry will provide services to multiple publishers/consumers/subscriptions and controls the access to this shared state. Hence, it is imperative to have the INFOD-Registry act as the authenticator for other services. Similarly, a publisher controls the dissemination of the messages and hence is deemed responsible for establishing the context with the consumers. In the future, it is envisaged that in later versions INFOD may introduce mechanisms for mutual authentication based on trust mechanisms. An example, is that future authentication of consumers by the publishers could be mediated by the registry.

4.1 Message Encryption and Data Privacy Requirements

INFOD advocates the use of mutual filtering techniques to provide smart dissemination of the messages. Mutual filtering requires the publishers and consumers to be able to interpret the contents of the messages being routed. As INFOD isolates a publisher from a consumer and does not require either the publishers or the consumers to authenticate each other, secured point-to-point communication becomes a non-issue for the base specification. It is assumed that publishers are able to authenticate the consumers based on their EPR references.

INFOD system provides non-repudiation of transmitted messages. It is recommended that the publisher signs its message and also provides its public key for subsequent verification by the recipients. It is suggested that the public key of each publisher is registered with the INFOD registry for retrieval by the network entities, such a public key should be registered with the PropertyVocabulary.

In some cases, INFOD publishers can determine the list of consumers and can provide messages for consumption by a single consumer or a group of consumers. No present security mechanism supports such communication pattern without the establishment of a shared key between the group of consumers and the publisher.

4.2 Integration with Authorization Model

Access control mechanisms for management of resources rely on the authentication mechanisms to authorize the access to the resources. Only authorized principals are allowed to register the publishers publish messages, create and manage the subscription and manage the consumers. It is recommended that the authorization model should provide a fine-grained control, preferably at the level of the evaluation context/ topics. Authorization models can be divided into two categories:

- Access model for INFOD resources
- Access model for INFOD messages

<x-concern id=X314>

Access models for the INFOD resources enforce the policies to allow restricted access to creation, deletion, and invocation of methods on service interfaces. Access models for resources can be maintained individually by each of the INFOD services as they are directly associated with the state maintained by the service. For example, an access model of INFOD registry resources controls the process of registering a publication and remains solely responsible for enforcing the related access policies.

</x-concern id=X314>

Access model for INFOD messages allows association of the dynamic authorization policies that control the access to the contents and the routing of the messages. Candidate examples include a publisher restricting dissemination of messages to a restricted list of consumers. Dynamic authorization policies may be propagated as a part of the secured conversation context and will need to be enforced by each participant that shares the context.