The Service Responsibility and Interaction Design Method: Using an Agile approach for Web Service Design

David E. Millard
Hugh C. Davis

Yvonne Howard
Lester Gilbert
Robert J. Walters

Noura Abbas
Gary B. Wills

School of Electronics and Computer Science
University of Southampton, Southampton, UK
+44 (0)23 8059 5749
{dem, ymh, na06r, hcd, lg3, gbw, rjw1} @ecs.soton.ac.uk

Abstract

Service-Oriented Architectures (SOAs) are increasingly deployed to achieve distributed systems that are modular, flexible and extensible. Designing for a SOA can be difficult, however. There are issues involving the granularity of the cooperating services, and there are no currently accepted conventions for describing a service or its interactions at an abstract level. This paper presents the Service Responsibility and Interaction Design Method (SRI-DM), an agile approach for engineering a Web Service design, based on capturing a scenario as a use-case, factoring this into a set of Service Responsibility and Collaboration Cards, and constructing a Sequence diagram illustrating their interactions in fulfilling the scenario. The paper presents the notation for each step and describes with the aid of an example how this process is used to create a service design within the domain of e-assessment.

1. Introduction

Engineering widely distributed systems has long been a challenge for the software engineering community. In the last few years a trend has emerged towards Service-Oriented Architectures (SOA) that aims at simplifying this problem. SOAs are an attempt to modularize systems in such a way that they are composed of independent software components that offer services to one another through well-defined interfaces. Such modularization typically is most useful for large and/or complex systems, but may also be used for other systems where a service orientation offers particular benefits, such as minimizing new building effort and maximizing the use of existing services. The service approach is ideally suited to more loosely coupled systems, where individual parts may be developed by different people or organizations. Wilson et al. describe the three main advantages of such a system as Modularity (dynamic coupling), Interoperability (standard interfaces), and Extensibility (encapsulation) [15].

Service-orientation is a philosophical approach to creating distributed systems, but there are a number of standards and approaches to providing them at an implementation level (including Web Services based on SOAP, GRID Services based on OGSI, and REST services based on HTTP and XML). Because of the difference in these approaches, and due to a lack of common notation and engineering experience, developing a service-oriented system can be difficult. Decisions must be made about how to divide a problem into logical services, how those logical services should be interfaced to maximize reuse, how they should be gathered together to create composite services, and what service-oriented implementation is best suited to each service, or to the design as a whole.

Agile methods are a number of software development methods which were proposed in the mid 1990s as a reaction to the limitations of traditional software development methodologies. Although these methods vary in practice, they share common principles such as [16]:

- deliver working software frequently within a short timescale
- close communication
- simplicity
- programming over documenting
- customer involvement
- encourage rapid and flexible response to change.

In this paper we present the Service Responsibility and Interaction Design Method (SRI-DM), an agile approach for the modelling of services at an abstract level that is independent of implementation. SRI-DM is agile as it enables a team of developers to quickly define a scenario and generate a number of services that will fulfill it. It is lightweight in that the documentation is minimal, and serves to drive the development forward as well as record it for others. SRI-DM:

- Defines a scenario with a use case diagram.
- Factors a set of services based on individual use cases.
- Represents these services at a high level using Service Responsibility and Collaboration cards (SRCs).
- Refactors these SRCs as necessary.
- Defines how Services might interact to fulfill the scenario using a Sequence diagram.

In this paper we present the SRI-DM and its notation, and present an example of SRI-DM being used to create a set of services in the domain of e-assessment.

2. Background

Service-orientation is an approach to creating stand alone components such that their potential for reuse is maximized. A number of standards, infrastructures and protocols have emerged which provide for this at an implementation level.

Web services have received a great of recent attention, and are defined around a set of standards (such as SOAP, WSDL, UDDI) developed by the W3C to make functionality available over the Web as simply as data [4]. Originally Web services had little support for security, which made them good for non-sensitive information and ad-hoc systems, but meant that they were not easily capable of supporting a virtual organisation (a tightly integrated secure system that is distributed) without additional non-standard security layers.

Grid services on the other hand assume a highly secure environment, and rely on certificates and authentication bodies to operate [7]. This heavyweight approach to security makes it possible to build virtual organisations that exchange and manipulate sensitive information, but might be prohibitive for developers wishing to build simpler services and applications.

These two technologies are becoming more closely defined and a new generation of Web Service standards (such as WS_Security) is now being introduced to add a standard layer of authentication and security to Web Services. This will make Web Services attractive for systems builders as it will become possible to build virtual organisations using relatively lightweight middleware.

A third approach to service provision is represented by Representational State Transfer (REST) [6], the name for a methodology rather than a set of standards, where HTTP and XML are used to send and retrieve data to a remote script or application residing on a Web server. REST services are popular, but are not secure enough to build virtual organisations and therefore will not be able to support the growing number of sophisticated service-based systems.

We believe that each approach is applicable in different situations, and that an agile methodology for service design should be agnostic about the service technology itself.

2.1 Establishing SOA

The take-up of Web services within enterprises may be problematic. Weatherley suggests that in the educational domain there are a number of barriers that prevent the widespread use of Web services for delivering Web-based educational materials [14]. These barriers relate to the need for understanding Web service protocols and the dynamic nature of the communication with Web browsers. In addition, in many institutions, developers are prevented from installing or running dynamic application software on their servers. Mukhi et al. believe that an increase in the adoption of SOA requires improvement to some of the non-functional features such as security transactionality and reliability [10]. They have developed a framework that supports and uses transactional and reliable services, achieved by using a policy model based on WS_Policy.

SOA specifications are progressing toward standardization in a variety of ways, including small groups of vendors and chartered technical committees. For example, an SOA Reference Model Technical Committee has been formed by OASIS members to encourage the continued growth of different and specialized SOA implementations whilst preserving a common layer of understanding about SOAs themselves. The e-Framework is an initiative by the UK's Joint Information Systems Committee (JISC) and Australia's Department of Education, Science and Training (DEST) to systematise a SOA for Education and Research [17]. We believe that substantive barriers to the establishment of SOAs include little shared understanding about how services should be developed, what granularity is appropriate for different problems, and no common notation to enable developers to share designs.

2.2. Modelling Services

Dijkman and Dumas explain the need for particular Service Oriented Design strategies [5], based on a number of characteristics that differentiate Service from Component-based design: High Autonomy (of designers and developers), Coarse Granularity (of service interfaces), and Process Awareness (close relationship with business processes). Enterprise level service development is most affected by the latter two characteristics. For example, Quartel et al describe the use of design milestones to help develop Web services from business practices [12], and Benatallah and Dumas have created environments to ease the creation of composite services [3]. Martin et al. suggest that the best way to implement Web Services in an enterprise is to start with a component-based architecture that exposes business process level services as Web services [9]. Wada et al have taken a model driven approach to this problem, building a model of the domain and then using this to derive an object design [13]; this kind of modelling has also been used with SOAs to validate a design [1].

In more loosely coupled community efforts, such as the JISC e-Framework [11, 17], the first characteristic of SOA design, High Autonomy, becomes the dominant problem, as services for the framework are being developed by a wide variety of institutions for a number of purposes. What is required is not just a common repository for services, but a community wide understanding of the domain, and how independently authored services fit within.

Wilson et al. present Reference Models as a potential solution [15]. Broadly speaking a Reference Model can be thought of as a description of how a set of services within a Framework collaborate to provide the necessary functionality for a particular domain. Reference models are a way to help architects and software vendors make consistent logical divisions in their architectures and products. However, they require a method for describing services and their interactions at an abstract, logical level.

We believe that the model-driven approach to service-design, while worthwhile in many domains where there is a consistent/constrained understanding of the processes, may be too heavyweight for situations where the domain is broader and the service model will need to respond to rapid changes. In these situations an agile approach seems more appropriate.

3. SRI-DM

The Service Responsibility and Interaction Design Method (SRI-DM) separates abstract representations of Services from their implementation. It uses a collection of logical descriptions (Service Profiles) to describe how a number of services, regardless of implementation, might be combined to solve a particular problem defined as a Use Case scenario. Our approach is based on the following principles:

- To facilitate and record a clear design path from a problem scenario to a software implementation.
- To be informed by agile principles and practices:
  - Start with scenarios that are useful and simple.
  - Enable developers to build the simplest service architecture with quality attributes of cohesion and loose coupling.
  - Draw on close relationship with domain experts to define scenarios and re-factor the SRCs.
  - Produce design documentation as part of the design process rather than using a document oriented process.
- Use UML 2.0 as a modelling method where possible, to enable understandability and promote links to CASE tools.
- Work at an abstract level that is non-prescriptive at implementation level.

There is a tension when designing services between ensuring that services are atomic (to encourage reuse) and ensuring that they are appropriate building blocks for a higher purpose, enabling the services to be combined to create a larger system. Services are always created within a context, and yet must be described independently from that context to be fully reusable. SRI-DM achieves this by treating individual Service Profiles as atomic, and placing the description of how they might be combined in a separate sequence diagram that is tied to a particular scenario. Therefore the method produces a design that has the following parts:

- **A Scenario**: presented as a Use Case Diagram and narrative that describes a problem for which a set of services can provide a solution.
- **Service Profiles**: a set of profiles that describe a number of services at an abstract logical level. These suggest granularity, and describe the individual capabilities of each service. They promote reuse and understanding of the design, while retaining flexibility in the implementation.

- **Sequence Diagram**: This describes one example of how the services can interoperate to fulfil the scenario.

Service Profiles are not concrete interfaces and so cannot be described using interface definition languages (such as WSDL). Instead they set the granularity of the model, and describe in a semi-formal way the role of each service and the potential way in which they might rely on one another.

In the rest of this section we will look at each part of the SRI-DM - Scenario, Service Profiles, and Sequences - and describe their formal notation.

3.1. Scenarios

Our method takes as its starting point a scenario that describes a problem that is to be solved using a set of interacting services. We have chosen Use Case diagrams as our method of modeling because they are high level and implementation independent. From an agile point of view they are also useful in that they relatively informal, simple, and help to define and structure a problem space without too much detail about the activities within that space. A brief narrative description is held alongside the diagram as a whole, as well as for each individual use case. These descriptions help disambiguate the use cases, explain the roles of the different actors associated with the use cases, and focus at a high level on what each use case involves.

Scenarios are developed in a community or user focused manner in line with agile principles to ensure that they are relevant. These use case diagrams capture the practice of an existing user community.

3.2 Service Profiles

Service Profiles are abstract descriptions of services that may be fulfilled by several different Service Implementations that may each expose different concrete interfaces. Service Profiles are thus modelled in an abstract way that does not prescribe a data model or dictate explicit methods. To do this we created Service Responsibility and Collaboration cards (SRCs) based on Class Responsibilities/ Collaborations, a modelling technique first described by Beck and Cunningham for eXtreme Programming (XP) [2].

Our SRCs model the capability of a service to realise a specific use case (a single bubble from a larger use case diagram). The aim of the cards is to help articulate a design, to suggest granularity, to guide refinement of that design, and to model for understandability. The SRCs do not show how services may be combined in a wider scenario, but do model possible collaborations with other services that might occur for this service to fulfill its own specific responsibilities.

An SRC card is a small card (we use A5 address cards in our design sessions).

- The name of the service appears at the top of the card.
- Down the left hand side of the card, we list the responsibilities of the service.
- On the right hand side we list and group other services which collaborate to fulfill the responsibilities listed on the left hand side.

The responsibilities of a service describe at a high level: what is it for, what does it do, and what can it provide to other components.

The guidance for CRC design is that a class should not have more than 3-4 responsibilities, as too many responsibilities corresponds to low cohesion in a class (a measure of a poor quality design). This guideline seems appropriate for the SRCs we have developed in our e-assessment domain cases.

The use case from which the service was derived can help indicate where collaborations will be required. In particular, *include* relationships are a strong indicator that a collaboration should be used, although as too many collaborations create a tightly coupled design they should be suggested sparingly. In particular, use cases connected through a use case *actor* do not necessarily collaborate.

The Service Profile is atomic in that any connection with other services is described in terms of how that connection might help this service fulfil its own role. This is different from describing how a set of services might be used together for some purpose that is greater than any individual service.

3.3 Service Sequence Diagrams

At the scenario level, services represented by SRCs must interact with each other to fulfil a wider purpose. These interactions are complex and include transactions, sequences and state. We looked at a number of UML 2.0 diagrams for representing a dynamic model, including state and activity diagrams. We decided that if the scenario modelling was to maintain the high level of abstraction necessary for agile development then it would be inappropriate to declare a detailed data model, or to specify the logic of the communicating services. So we use Sequence Diagrams to represent the interactions, showing which services should communicate and in which order, and containing enough description to show how the individual services are responsible for moving and

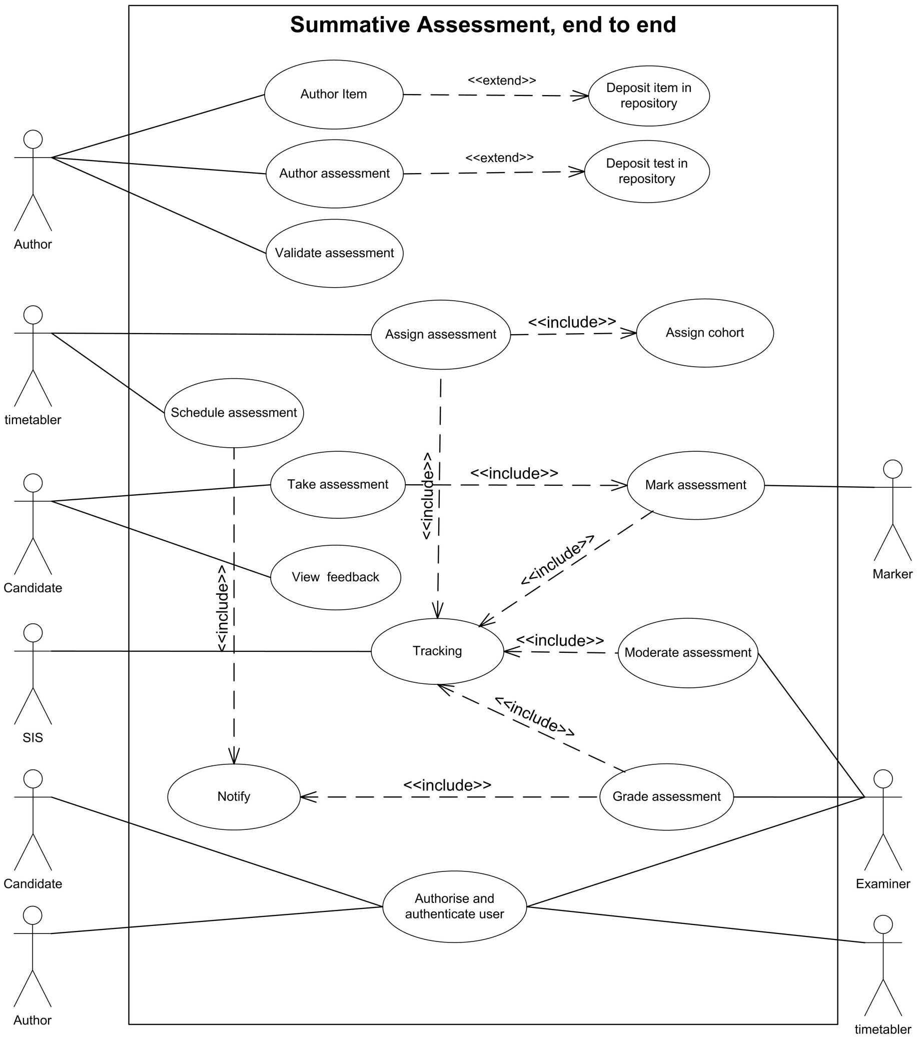

Figure 1: Use Case for Summative End-to-End CAA

processing data, without having to specify the detail of the data model or the decision making logic.

4. An Example Factoring

We call the process of deriving a set of services for a given scenario *Service Factoring*. The philosophy behind our method is that this whole process is transparent and fully audited. It begins with a community consultation exercise that produces a number of scenarios. These are then formalised as Use Case Diagrams, and from each Use Case a SRC (or set of SRCs) is created. Referring back to the Use Case Diagram of the Scenario allows us to specify a Sequence Diagram that describes how these services interact to fulfil the goal of the scenario.

The authors have been involved in a project called FREMA (the Framework Reference Model for Assessment) which has examined how a number of scenarios from the e-assessment domain might be supported via services. The Assessment Domain can be described as a brown-field site for service developers due to the many existing tools and standards in the area. Therefore what is required is not just a common repository for services, but a community wide understanding of the domain, and how services fit within it. FREMA has developed a Community Model, based on a Semantic Wiki that should help the

Assessment Community develop Web Services in this context.

4.1 Developing the Use Cases

The first part of the FREMA project was to elicit practice from a number of members of the e-assessment community via workshops and semi-formal interviews, including the UK Centre for Educational Technology Interoperability Standards (CETIS), Qualification agencies such as SQA and Edexcel, and a number of Higher Education Institutions. While the resulting view of assessment was very broad, the most common scenario was one of Computer Aided Assessment (CAA); this concerns a lecturer or teacher who can set summative assessments to be taken digitally. We call this the End-to-end Summative Assessment Scenario. Figure 1 shows a part of the Use Case diagram constructed for this scenario. The granularity of the Use Cases translates directly to the granularity of the Service Profiles (although there is not necessarily a one-to-one mapping of Use Case to Service Profile). Broadly speaking it has three parts: The first models the authoring of the assignment (and potentially of the items within the assignment). The second represents the run-time system, including the assessment event itself. The last part models the post-assessment process of marking and grading. There is no clear distinction between the parts. For example, scheduling is part of authoring and the run-time, and feedback is part of the run-time and the marking/grading

4.2 Constructing the SRCs

Deriving SRCs from Use Cases is a complex process:

1. Work through each use case. A traditional noun and verb analysis is a useful technique; verbs can indicate the responsibilities of the services that fulfill the use case, and nouns imply a data model and inform the narrative. From the verb analysis write down all of the operations needed for a use case.
2. Consider which operations might be common with other SRCs and move them from the responsibilities column to collaborations.
3. Group the operations into responsibilities and collaborations.
4. Identify which responsibilities would benefit from which collaboration.
5. Test the completeness/accuracy of the design by working various scenarios.
6. Re-visit the SRC and re-factor as necessary as other SRCs are developed, and as common collaborations become apparent.

Figure 2 shows this process applied to the "Take Assessment" Use Case (the numbers above each card refer to the stages described above). The Use Case description is used to derive the initial list of operations, which are consequently factored into a set of responsibilities and collaborations. Sometimes the operations that are moved to collaborations also remain as responsibilities (for example, Choose Assessment spawns a collaboration called Schedule, but remains as one of the responsibilities of the Service), because the service still has a responsibility to allow users to choose an assessment, even if this is done via a collaboration. On the other hand, Tracking is removed as a responsibility because it is not something that this service offers to others.

4.3 Building a Sequence Diagram

A sequence diagram is the way to which a group of services can interact to fulfill the original scenario. It cannot be a definitive representation of service interaction in general, as services are asynchronous, and some of the communication can be reordered without affecting the performance of the system as a whole. So a sequence diagram acts as a demonstration and validation of the SRCs, rather than as a definitive template for service interaction. Figure 3 shows a sequence diagram for the part of the scenario related to "Take Assessment", and in particular the interactions around the candidate actor. Collaborations are modelled, although in this diagram they are grouped together into one column to aid clarity.

The sequence diagram describes how the core services interact so no collaborations are shown; this is because at this level the services do not need to know how other services are implemented, merely that they fulfill their responsibilities. The diagram shows which services interact, and in what order, in order to make the scenario happen

State is not shown, because that is an implementation detail, and the data passed around is described verbally, but not formally, for the same reason. The SRCs and Sequence diagrams are not intended to provide a complete description of interacting services; it is a reference model, and not an interface description or detailed process model. However, we would expect systems builders to be able to use them to describe their particular implementations and to aid the construction of interoperable interfaces. Developers can use the SRCs to decide what

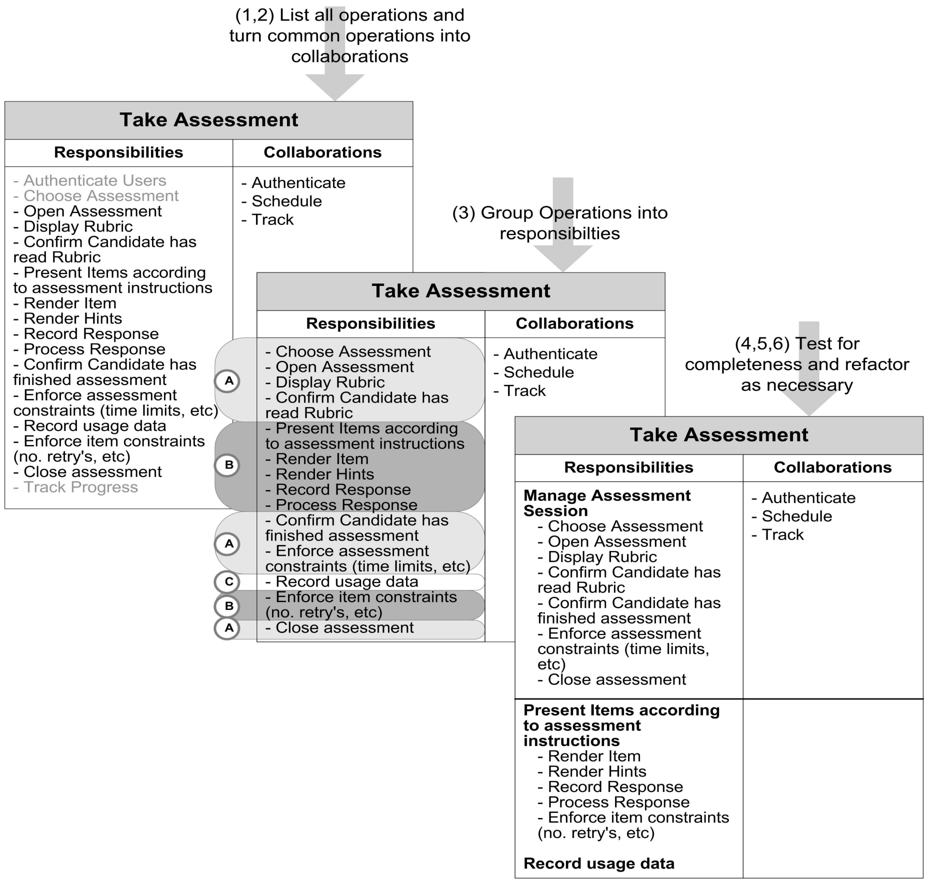
(1,2) List all operations and turn common operations into collaborations
Take Assessment
Responsibilities
Collaborations
- Authenticate Users
- Choose Assessment
- Open Assessment
- Display Rubric
- Confirm Candidate has read Rubric
- Present Items according to assessment instructions
- Render Item
- Render Hints
- Record Response
- Process Response
- Confirm Candidate has finished assessment
- Enforce assessment constraints (time limits, etc)
- Record usage data
- Enforce item constraints (no. retry's, etc)
- Close assessment
- Track Progress
- Authenticate
- Schedule
- Track
(3) Group Operations into responsibilties
Take Assessment
Responsibilities
Collaborations
A
B
C
Manage Assessment Session
Present Items according to assessment instructions
Record usage data
(4,5,6) Test for completeness and refactor as necessary

Figure 2: The factoring of the "Take Assessment" Use Case into a SRC

responsibilities their service implementations will take, and the sequence diagrams to see what consequences this will mean for interfaces to other services.

4.4 Presenting the Design

Figure 4 shows the final set of SRCs that we derived after several iterations of the factoring and the re-factoring process (note that this overview diagram does not contain the full details for each SRC). The core services that we believe are needed to support this activity are shown within the large Summative End-to-end (FREMA) bubble, with services that may be used via collaborations around the outside. The core services are divided into the three parts identified within the use case earlier (authoring, run-time, and post-assessment), although this is purely to add clarity to the diagram and has no engineering consequences.

In the re-factoring process we identified a number of core services that seemed to be involved in many collaborations: these were Notify, Track and Metadata Tagger, these are shown in a separate layer at the bottom of the bubble. The other collaborations that lie around the outside of the main bubble seemed less important, but may well be core services for another scenario. We have tried to group these into likely areas, such as Grading and Previewing, but again this grouping is purely to add clarity.

5. Validation and Discussion

Our validation strategy has been to ensure that the designs produced via SRI-DM are sensible, accessible and intuitive. To this end we have undertaken a formal evaluation of our scenarios and our methods of presenting them. For our formal evaluation we presented versions of our e-Assessment research and resulting scenarios at the CETIS Assessment Special Interest Group (SIG). This is a self-selecting group which includes early adopters, developers, and representatives of standards bodies. The reaction of the group was encouraging, they believed that the

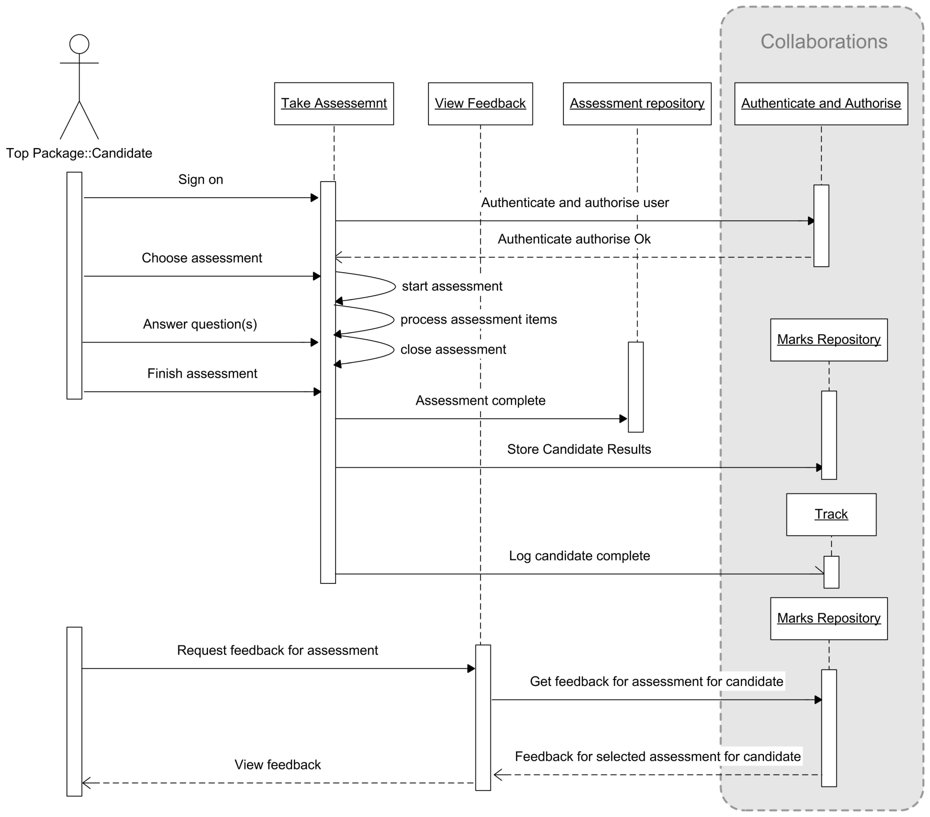

Figure 3: Sequence Diagram from the Take Assessment Use Case

scenarios that we had developed were accurate and important to the e-Assessment domain, and the use case diagrams that we presented captured the scenarios well.

We have also presented the CETIS group with the SRC and interaction diagrams for the CAA scenario. Reaction to the cards and the interaction diagram was good. All delegates agreed that it was a sensible granularity at which to model services, and that the SRC and interaction diagrams were suitably expressive. Many believed that this lightweight modelling would be useful in their existing service design practice. Based on this we are now in a position to engage more directly with community members, in particular with a group of developers at Kingston University, to undertake a more formal evaluation of the SRI-DM, both in terms of its representation (via a formal design review, evaluating the effectiveness of the model compared to its aspirations), and as a service design process (using the SRI-DM to guide the development of a mini-project at Kingston with a formal evaluation at the project's close).

We already use SRI-DM ourselves to create a set of services for several core assessment scenarios (including CAA), and as a result have a number of personal reflections on the design method.

We believe that one of the most difficult challenges with service design is choosing an appropriate granularity at which to define services. With SRI-DM we have chosen a top-down approach that is firmly built on a starting scenario and use-case diagram. These are typically high level views of a problem space, and translating them almost directly into service profiles produces a high level design But because SRI-DM does not capture business logic or interfaces in a detailed way, it becomes easy to re-factor services in order to break down that high level design to a level at which the designers are comfortable

This approach is agile, as it requires only a little modelling overhead, and produces a stable service design before the expensive process of agreeing data models and interfaces is undertaken. It also produces design documentation as part of the design process, rather than adding a separate task of recording an external design process.

In addition, the feedback from the domain experts assures the relevance of the scenarios and the re-factored service profiles.

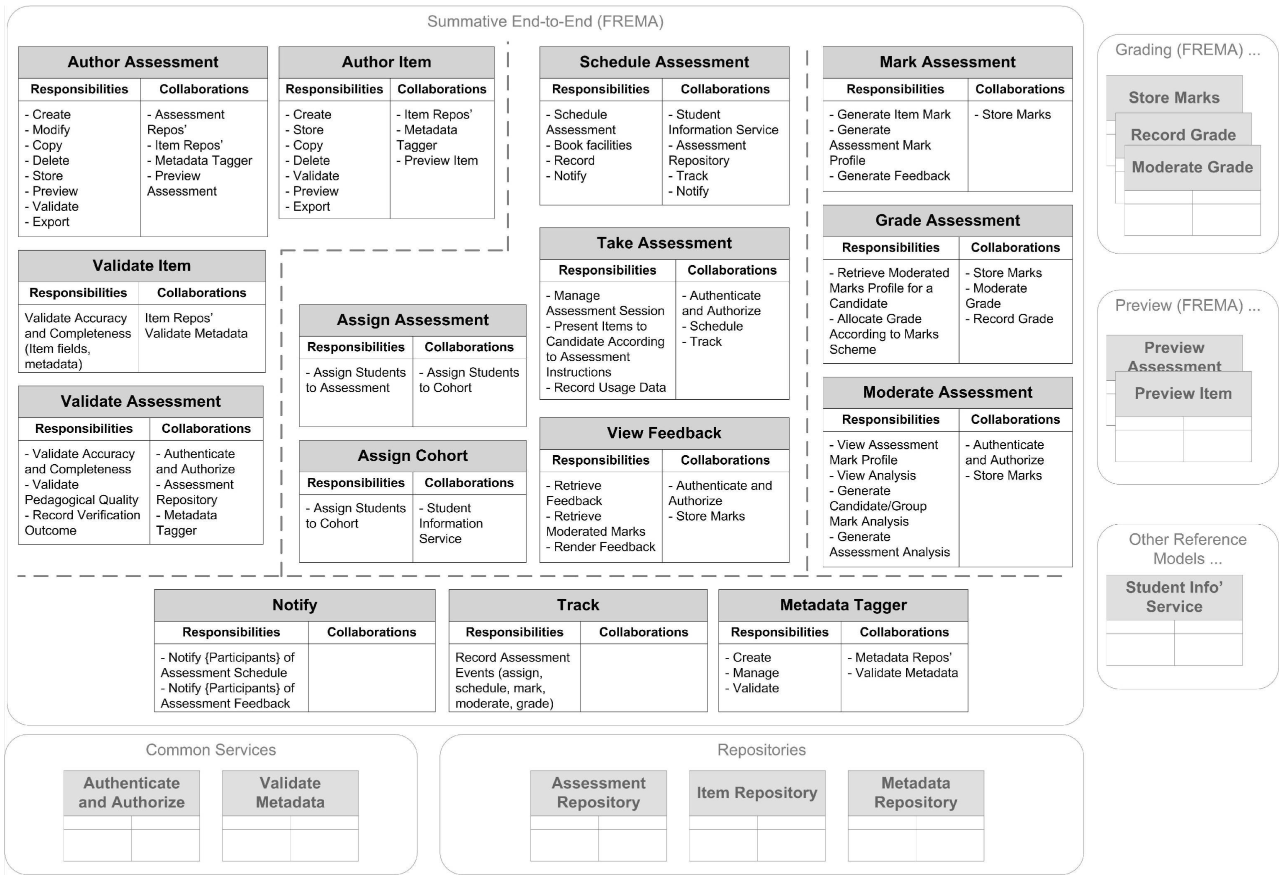

Figure 4: SRCs for Summative End-to-End CAA

Another challenge with service design is agreeing on a service workflow. SRI-DM does not attempt to define the full rules of interaction (causal relations, points of synchronization, critical paths, etc). This is another way in which SRI-DM is an agile approach; a full model of all the ways in which services can interact is not needed to produce a working system of services, and so SRI-DM does not make designers create this. Instead it demonstrates the validity of a service design by showing *one example* of how a set of services could interact to fulfill the scenario. Our major observation while developing services with SRI-DM is the paucity of traditional flat-file documentation for linking evidence with decision making. This inflexibility in justifying design decisions may be a real problem with SOAs due to the distributed way in which services are often created. To cope with this we have been developing the notion of a *Community Reference Model* alongside SRI-DM, this is a community Web site, where the scenarios and their evidential resources can be described, linked and discussed[1]. We hope that by explicitly supporting the use-cases, service profiles and interaction diagrams of SRI-DM we can also encourage the community to start building common models of how services can interact to fulfill scenarios, leading eventually to common services themselves. We are currently developing this idea using a *Semantic Wiki*, and plan to hand this resource over to the e-assessment community through the CETIS SIG later this year.

6. Conclusions

In this paper we have presented the Service Responsibility and Interaction Design Method (SRI-DM), an agile approach to designing Web Services. The SRI-DM is a process of factoring abstract service profiles from formal domain scenarios. In the method

[1] For an example see the FREMA Web site: www.frema.ecs.soton.ac.uk

the scenarios are modelled as use-case diagrams, and the profiles as Service Responsibility and Collaboration cards (SRCs). SRCs capture the granularity of a service by defining the set of responsibilities that it holds, and the collaborations that it uses to fulfill those responsibilities. Because SRCs only model atomic service profiles the SRI-DM also uses a UML 2.0 sequence diagram to show how the SRCs interact to fulfill the original scenario. The sequence diagram is not intended as a full model of all possible interactions, but as an example of one interaction that demonstrates the validity of the service design.

The SRI-DM focuses on the rapid factoring of a set of services given a well-understood scenario. We are currently evaluating SRI-DM through an independent project, and plan to take the method forward to the e-assessment development community through a Web-based Community Reference Model.

As SOAs become more reliable, and the standards underlying them more stable, it seems inevitable that they will form the basis of many distributed systems. If these systems are to be created as quickly and as flexibly as current software deployments then we must use design methodologies that are agile enough to cope with rapid turnaround, yet create designs that are fit-for-purpose, and leave a documentation trail strong enough to support software throughout its lifetime.

7. References

1. Baresi, L., Heckel, R., Thöne, S., and Varró, D. (2003). Modeling and validation of service-oriented architectures: application vs. style. In Proceedings of the 9th European Software Engineering Conference held jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering (Helsinki, Finland, September 01 - 05, 2003). ESEC/FSE-11.
2. Beck, K. and Cunningham, W. (1989). A laboratory for teaching object oriented thinking. ACM SIGPLAN, Notices, 24(10):1-6, October 1989.
3. Benatallah B., Sheng Q., and Dumas M. (2003). The Self-Serv environment for Web services composition. IEEE Internet Computing, 7(1):40-48, Jan/Feb. 2003.
4. Curbera, F.; Duftler, M.; Khalaf, R.; Nagy, W.; Mukhi, N.; Weerawarana, S. (2002). "Unraveling the Web services Web: an introduction to SOAP, WSDL, and UDDI," Internet Computing, IEEE , vol.6, no.2, pp.86-93, Mar/Apr 2002.
5. Dijkman, R. and Dumas, M. (2004). Service-oriented Design: A Multi-viewpoint Approach. International
6. Journal of Cooperative Information Systems 13(4), December 2004.
7. Fielding, R. T. and Taylor, R. N. 2002. Principled design of the modern Web architecture. ACM Trans. Inter. Tech. 2, 2 (May. 2002), 115-150.
8. Foster, I., Kesselman, C., and Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Int. J. High Perform. Comput. Appl. 15, 3 (Aug. 2001), 200-222.
9. Highsmith, J. and Cockburn, A. (2001). "Agile software development: the business of innovation". Computer, Sep 2001, Volume: 34, Issue: 9, pg 120-127, ISSN: 0018-9162.
10. Martin J., Arsanjani A., Tarr P., and Hailpern B. (2003). "Web Services: Promises and Compromises," Queue vol. 1, pp. 48-58, 2003.
11. Mukhi N. K. and Plebani P. (2004). "Supporting policy-driven behaviors in Web services: experiences and issues" in proceedings of the 2nd international Conference on Service Oriented Computing ICSOC '04, (New York, NY, USA, 2004).
12. Olivier B., Roberts T., and Blinco K., (2005). "The e-Framework for Education and Research:An Overview". DEST (Australia). Downloaded 10 March 2007 from http://www.e-framework.org/Portals/9/Resources/eframeworkrV1.pdf
13. Quartel D.A.C., Dijkman R.M., and van Sinderen M.J. (2004). Methodological Support for Service-oriented Design with ISDL. In: Proceedings of the 2nd ACM International Conference on Service Oriented Computing (ICSOC), New York City, NY, USA, pp. 1-10, 2004.
14. Wada, H., Suzuki, J., and Oba, K. (2005). Modeling turnpike: a model-driven framework for domain-specific software development. In Companion to the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (San Diego, CA, USA, October 16 - 20, 2005). OOPSLA '05. ACM Press, New York, NY, 128-129.
15. Weatherley J. (2005). "A Web service framework for embedding discovery services in distributed library interfaces," in proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries JCDL '05, Denver, CO, USA, 2005.
16. Wilson, S., Blinco, K. and Rehak, D. (2004). Service-Oriented Frameworks: Modeling the infrastructure for the next generation of e-Learning Systems. A Paper prepared on behalf of DEST (Australia), JISC-CETIS (UK), and Industry Canada. Downloaded 10 March 2007 from http://www.jisc.ac.uk/uploaded_documents/AltilabServiceOrientedFrameworks.pdf
17. Larman, C. (2004). Agile and Iterative Development: A manager's guide. Pearson Education.
18. JISC (2007). http://www.e-framework.org/.