I3-D8

Feedback report on requirements from other working groups

Project title:	Reasoning on the Web with Rules and Semantics
Project acronym:	REWERSE
Project number:	IST-2004-506779
Project instrument:	EU FP6 Network of Excellence (NoE)
Project thematic priority:	Priority 2: Information Society Technologies (IST)
Document type:	D (deliverable)
Nature of document:	R (report)
Dissemination level:	PU (public)
Document number:	IST506779/Dresden/I3-D8/D/PU/b1
Responsible editors:	Jakob Henriksson
Reviewers:	Charlie Abela (int.), Sergey Lukichev (ext.)
Contributing participants:	Dresden, Linköping, Naples, Cottbus
Contributing workpackages:	I1,I3,I4,I5,A2,A3
Contractual date of deliverable:	3 September 2006
Actual submission date:	5 September 2006

Abstract
For the composition ideas and technologies of the REWERSE working group I3 to be successful, it is vital to understand how the collaborating groups within the network view its work. It is important to understand, and take into consideration, the requirements of the collaborating working groups wrt. their languages and applications, which I3 is to support with composition technology. To this end, through a survey, we have collected such requirements and in this report present the collected result.

Keyword List
composition, invasive software composition, component based development, survey, semantic web

Project co-funded by the European Commission and the Swiss Federal Office for Education and Science within the Sixth Framework Programme.

Feedback report on requirements from other working groups

Uwe Aßmann[1] **and Jakob Henriksson**[1] **and Kristian Sandahl**[2]

[1] Fakultät Informatik, Technische Universität Dresden
Email: {uwe.assmann|jakob.henriksson}@tu-dresden.de
[2] Institutionen för datavetenskap, Linköpings universitet
Email: krs@ida.liu.se

5 September 2006

Abstract
For the composition ideas and technologies of the REWERSE working group I3 to be successful, it is vital to understand how the collaborating groups within the network view its work. It is important to understand, and take into consideration, the requirements of the collaborating working groups wrt. their languages and applications, which I3 is to support with composition technology. To this end, through a survey, we have collected such requirements and in this report present the collected result.

Keyword List
composition, invasive software composition, component based development, survey, semantic web

Contents

1 Introduction

The purpose of the REWERSE working group I3 (Composition and Typing) is to provide composition and typing technology for the languages and applications developed by the other working groups within the REWERSE network. Thus, the success of I3 relies heavily on how well its technology is being adopted by its collaborating partners and their work.

The degree to which the technology of I3 is adopted depends to a large extent on how well its work is understood by its partners. If the work performed by I3 is properly understood, it enables a closer cooperation between I3 and its partners. This close symbiosis is essential for the success of I3.

Here we focus specifically on the composition part of the working group I3. In order to find out how well the composition work is understood by our partners, we constructed a survey to find out their level of understanding and their requirements of composition. The survey can be found in Section 2. The results from the survey and a description of the requirements defined by the other working groups is summarized in Section 3.

2 The survey

The following is the survey that was used to find out the level of understanding and the requirements among the collaborating working groups of I3.

2.1 General

Component-based software engineering has proven invaluable in many areas of software development. The major reasons for this are: saving time and increasing quality. Time is saved since complicated coding will not be performed every time a certain function is needed. Quality is increased since the component provider has long experience of many customer's needs.

Q1 Will, for whatever reason, component based development never be needed for your REWERSE-related language or applications?

Q2 If you can, describe one or several components in your REWERSE-related language or applications in terms of name, semantics, externally visible properties.

Q3 How many components do you guess will be needed in practical use of your REWERSE-related language or applications?

There are several techniques used to compose software out of components, such as mediating between interface languages, integrating data and control flow, weaving of cross-cutting aspects etc.

Q4 Describe your future need for support of component based development for your REWERSE-related language?

2.2 Type of support

Q5 Do you wish to see support for composition of the concrete syntax (if applicable) of your REWERSE-related language?

Q6 Do you wish to see composition support for your REWERSE-related language in a development tool such as Eclipse?

Q7 If you answered yes to both Q5 and Q6, please describe pros and cons with the type of support.

2.3 Type of composition

Composition can be done at compile time or at run-time and components can be written in the same or different languages.

Q8 Is run-time interoperability for your REWERSE-related language/engine, with other languages/engines, needed?

Q8a If run-time interoperability is needed, for which other languages/engines do you see a need for interoperability?

Q9 Is reuse of program code for your REWERSE-related language needed?

2.4 Usability

Working with components also introduces new type of work, such as, understanding components, specifying in which order components are integrated, setting parameters and writing adaptation software. Sometimes there is a need for specialized languages for defining a composition recipe. An analogy can be made to the make-files of Unix.

Q10 Would reuse of program code make it easier to write/express programs/rules/statements in your REWERSE-related language?

Q11 Would reuse of program code make it harder to read and understand programs/rules/statements written in your REWERSE-related language?

Q12 Do you see a need to construct new languages (language descriptions) from component languages/schemas?

Q13 Do you see a need for development and composition support for constructing new XSD (XML Schemas) from component schemata?

2.5 Cooperation with I3

I3 people possess long experience with technical research and industrial experience with component-based software engineering. It is our hope that this knowledge can be utilized widely in REWERSE wherever a true need arises.

Q14 Do you feel you have a good understanding of what kind of composition techniques are offered by I3 for your particular REWERSE-related language?

Q15 Should I3 make a greater effort in trying to convey what kind of composition is being targeted?

Q16 From what has been presented so far at common meetings, do you feel that the composition techniques of I3 are useful to your project?

Q17 Do you feel that a special common meeting between your working group and I3 is needed to better understand how the composition ideas of I3 can be applied to your particular case?

Q18 Should there be more seminars/demos/tutorials at common REWERSE meetings where I3 composition is presented in more in detail and especially what the underlying composition techniques are?

Q19 What general expectations did/do you have of the composition techniques of I3?

Q20 Are the requirements for re-usability and modular development for your project/language addressed by I3 and their composition techniques?

Q21 Are you doing or have you done your own research in Component-based development? If yes, please shortly describe the status of your work.

Q22 Are you or have you cooperated with anyone outside REWERSE in component-based development? If yes, describe the forms of cooperation and the status of the work.

3 Results

The survey in Section 2 was sent out to 17 members and partners of the REWERSE network. The survey was answered by 9 participants (53%), making it a valuable resource for input into future direction of the composition work of I3. Answers were received from both application groups (A) and issue groups (I). The following REWERSE working groups were represented in survey responses received: A2 (Bioinformatics), A3 (Personalization), I1 (Rule Markup), I4 (Reasoning-aware Querying) and I5 (Evolution on the web).

Here we present the results as collected from the various working groups mentioned above. First, in Section 3.1 we describe the general results as found by the survey in Section 2. In Section 3.2 we highlight some requirements and specific input from certain selected and relevant working groups.

3.1 General survey results

First of all, it should be concluded that the consensus among the survey participants was that components were indeed needed and useful, to some extent, in their respective languages and applications. Thus, it was generally agreed that continued co-operation with I3 was worthwhile and valuable. Certainly, the kind of components considered depend largely on the specific language and application involved. Thus, more detailed information for each case can be found in Section 3.2.

Where applicable (for languages), the need for reuse of source code, i.e. source code as components, was seen as something positive and interesting. The foremost reason for this was the assumed advantage of having to write less code and the belief that programs could be made easier to read. Participants were also aware of the fact that sometimes it might also be harder to read such code, but that this possible effect is unavoidable. The desire to consider components

as reuse of source code goes along well with the current work carried out by I3, as this currently is the main approach on which we are focused.

For languages, where applicable, support for concrete syntax was desirable. Integration into and support from development tools such as Eclipse was also desirable. However, the research aspect of such efforts was put in question.

The composition work of I3 is largely focused on composing programs of languages, i.e. source code composition (albeit not white-box composition). Composition can also be considered at a different level, at the level of languages themselves instead of their instantiations (programs). One can in such a way construct new languages from component language descriptions (grammars). The only group interested in such language composition capabilities is I5 (for more discussion, see Section 3.2).

Some participants felt they possessed partial understanding of the composition work of I3. However, in general, the level of understanding of composition among the different working groups was poor. Though a somewhat negative result, a very valuable one for future directions, co-operations and common meetings. It is clear that I3 needs to spend more time trying to explain what different kinds of composition are available and can be provided.

Most participants expressed a wish to have tutorials and demos at common meetings to better understand the composition ideas. However, it was also clear that while there was a feeling of need for composition tutorials, there is an urgency for discussions focused around concrete examples and use-cases. For example, there was expressed a need for hands-on workshops where some selected scenario could be dissected and investigated for needs and possibilities of composition.

3.2 Highlighted results

In this section, specific working groups and their requirements are highlighted.

3.2.1 Bioinformatics (A2)

A2 is working with their showcase tool *GoPubMed* [1]. GoPubMed is a tool for searching for relevant medical articles, with the help of the *Gene Ontology*[1]. It allows for researchers to quickly get an overview of the different articles available in a particular field, and to which terms in the Gene Ontology they relate. This can give researcher a good overview on trends and focus in the field. The prototype GoPubMed allow searches to be viewed and browsed in context of the Gene Ontology.

There has not been much interaction and exchange between I3 and A2. Because of this the requirements and possibilities for interaction are still unclear and vague. The implementation of GoPubMed makes uses of components provided by the Cocoon framework[2]. These components are basically run-time services providing certain functionality to the system as a whole. The composition that I3 has focused on so far is not in line with the requirements needed for the system implementation of GoPubMed. However, there are situations in the GoPubMed implementation where the composition technology of I3 can come to use (see use case below).

[1] `http://www.geneontology.org/`
[2] `http://cocoon.apache.org/`

Use case: GoPubMed glue and SQL composition

GoPubMed makes use of several smaller sub-system for handling certain parts of the general implementation. The sub-systems can be varied in a number of ways, for example, different document publishing databases and ontologies might be used. The details of how they are connected and varied are described in a specific format and file, which could be composed with I3 technology, thus, implementing variability of sub-systems from template descriptions.

Further more, GoPubMed makes use of SQL queries to fetch documents, from a store, based on different criteria and strategies. SQL already makes available some type of generic *slots* in what is called *prepared statements*. I3 technology could be used in GoPubMed to provide more flexible, type-safe, SQL composition abilities for their purposes. This could also serve as a use-case for the more general problem of composing SQL queries and extending what is provided by means of prepared statements. Note also that SQL has relationships with the XML query language XQuery (which is interesting also for the related query language Xcerpt).

Summary (A2)

- Components used in the showcase tool of A2, GoPubMed, are service oriented and not so much focused on static code reuse.
- More interaction between the working groups is needed to fully understand how they can work together, if at all.
- Some initial requirements and needs for composition in A2 has been found through two use cases described above.

3.2.2 Personalization (A3)

The applications developed in A3 and their requirements have previously been described in Deliverables A3-D2 [4] and A3-D6 [6]. The requirements mentioned there are not specifically addressing composition, but rather describe their overall requirements in terms of many other aspects. Here instead, we discuss some composition related requirements of A3 in more detail.

A3 is working with, among other things, personalization and personalization services. In terms of composition, A3 is mostly centered around web service composition, but has also requirements for static source-code composition in development of personalization service functionality.

So far, there has not been much interaction between A3 and I3. The two groups should work more closely to see how they may co-operate better.

Use case: Personalization service development

A3 urgently needs help with facilitating ease of developing new personalization services. The whole idea of the Personal Reader Framework [5] settles on the idea that new personalization services are *plugged* into the framework at runtime, via registration through the Universal Description, Discovery and Integration (UDDI) protocol. Then, they are discovered through a matchmaking process, checked for their applicability to certain user tasks, and, in case of the applicability being sufficiently high, the services will be executed (maybe after some preceding customization and negotiation steps).

The goal is to make it as easy as possible for a Personalization Service developer to embed the personalization functionality he requires into a Personalization Service. Communication protocols need to be used and requests and replies handled (usually through some message protocol such as SOAP[3] with embedded RDF, so normally transformations into an appropriate Java structure are necessary, which should already be available for developers).

The help in development of personalization functionality could be used as a use case for showing the applicability of I3 technology.

Summary (A3)

- More interaction between the groups is needed to understand each others technology.
- The applications in A3 are mostly service oriented, but as seen in the use case above, there is also need for code reuse and composition.

3.2.3 Rule Markup (I1)

I1 works with modeling of rule languages. Specifically, R2ML [3] was developed by I1 and is a comprehensive and user-friendly XML rule format. Through the design of R2ML, I1 has gained much experience in this field and is now investigating how this language may be used by rule developers in the best way. The latest version of R2ML (version 0.4 at writing) supports, among other things, for the definition of vocabularies, which might be able to be used more flexibly if composed together with rule-sets. Questions such as how to compose rule bases and corresponding vocabularies in a way to make it easier to maintain, are of importance to I1. Even if it is not expected for users to write R2ML directly, investigation is carried out on usage patterns for mapping different rule languages into R2ML. These patterns will then serve as best practices for using R2ML.

The composition requirements for I1 revolves largely around the need to compose components of the rule markup language R2ML. This in order to allow rule authors to write large rule-sets in an easy way. The goal with composition within I1 would be to allow for efficient reuse of rules and vocabularies in the development of rule-based systems.

I1 are also active in the area of Semantic Web Services where their rule languages can be used to specify web service rules, which describe specific behavior in services. For example, *Reaction Rules* are used to specify interaction and behavior of web services. It should be investigated how composition should be addressed wrt. Web Services. I.e. if composition should be addressed locally at different Web Services (e.g. allowing reuse of rules for specifying behavior), or if global interoperability issues should be addressed (e.g. in allowing Web Services to make use of each other's services in a better way). See Section 5 for further comments.

I1 is also, to some degree, interested in composing XSD schemas. For example, R2ML is currently constructed from several schemas. It is also an issue for further investigation how R2ML can be re-used in some other markup rule language. Composition might serve as a technique for achieving this language integration.

R2ML will be proposed as a main input to the Rule Interchange Format (RIF)[4]. Also, probably, the final new standard for representing rules will be based on R2ML concepts and composition can most likely play a role there.

[3] http://www.w3.org/TR/soap/
[4] http://www.w3.org/2005/rules/wg

To allow for more successful interaction between I1 and I3, it is needed for I3 to better understand the work of I1, specifically, the details of R2ML. Also, I3 should devote some more time towards specifically targeting the question of how to compose markup languages in general and rule markup languages in particular.

A requirement of I1, and where research should be focused, is to develop a *proof of concept* for using the technologies of I3 in composing rule sets.

Use case: Re-constructing R2ML

R2ML is currently defined from several schemas, which is not done easily. One possible use case within I1 is to look more closely at R2ML and to see if it is possible to use the composition techniques of I3 to compose R2ML from schema components. Also, to investigate how other rule languages can make use of R2ML concepts through composition.

Summary (I1)

- I3 needs to better understand the work of I1, in particular the rule markup language R2ML.
- I3 should consider, in general, how to compose markup languages.
- Proof of concept is needed for composing rule sets using R2ML and related rule languages.
- Investigations should be carried out on how to address Web Service composition wrt. the rule languages developed by I1.

3.2.4 Reasoning-aware Querying (I4)

The work of I4 is mostly centered around the XML query and transformation language Xcerpt [2]. Xcerpt is the first REWERSE language that was targeted by I3 and its composition techniques. Composition tooling was in place and available for the second annual REWERSE meeting (March 2006), supporting concrete syntax Xcerpt. During the same time, the first concrete discussions centering around Xcerpt composition examples were conducted. The message is clear from I4 in that they see a need for components and component based development in the languages they are involved in, Xcerpt and its ECA language extension XChange [7]. The examples produced so far for Xcerpt were well received by I4 and they were understood, making it a good start for future discussions.

The main reason for introducing the composition techniques of I3 into Xcerpt is to allow rule authors to write large rule-sets in an easy way. This way, since reusable components are used, less code is needed to be written, less bugs appear and the rule-set as a whole is more easily understood. However, as noted by I4, the *proof of concept* is missing. What is still lacking is a good concrete example where the benefits of a composed Xcerpt rule set are clear.

For Xcerpt, the different syntactic structures that could be reused, and composed, include the following (non-exhaustive list): modules (rule sets), rules, query components, construction components, terms, type annotations, type declarations. Note that a large part of these constructs are also included in XChange. Furthermore, it seems interesting to try to develop a library of patterns (components) for Xcerpt, which can be used by Xcerpt program authors. Composing these patterns could be realized by the I3 composition techniques. What is clear is that there is a need for interesting examples.

Within the work of developing Xcerpt there is also research conducted in the field of *typing*, also specifically typing for Xcerpt. It is clear that there is a need for composition, not only for rule languages such as Xcerpt, but also for type systems used therein. Further research should be devoted to investigate the needs and possibilities for composition wrt. typing.

I4 has also mentioned interest in investigating how Xcerpt itself could be used to specify composition programs, i.e. programs specifying how components are put together in useful ways. Most likely, this would require a slight extension of Xcerpt itself, such as to support the additional composition constructs needed. However, in doing this, Xcerpt would then natively support composition. The possibilities in this direction should be investigated.

Finally, I4 noted that composition is an issue for web query languages in general, and not limited to Xcerpt. This is also along the lines of current work by I3 in trying to support the XML query language XQuery [8] with the same composition techniques. A case-study is being performed on XQuery in trying to find out what kind of composition is useful and needed for that specific language. Success in composition for Xcerpt is likely to be reusable for XQuery, and vice versa. It should be interesting to investigate the needs for composition for a query *core*, which is common to several (semantic) web query languages.

Use cases: Xcerpt composition

As a part of trying to find more interesting examples demonstrating the composition techniques of I3 on Xcerpt, two examples are being investigated.

- **Tree traversals** Whenever you want to traverse the content of an XML document tree, you have to program this in an Xcerpt rule. We are investigating how it is possible to construct such a tree traversal component, which can then be reused for several types of applications where such traversals are needed (e.g. to construct the content section of a document).

- **RDF/XML normalization** Documents described in RDF/XML syntax can express the same statements in different forms, sometimes making the actual information hard to visualize and understand. One can normalize this document, i.e. making each RDF triple explicit, by using Xcerpt rules. The Xcerpt program performing this job is made up of a lot of complicated rules, which possibly could be made easier to understand by dividing up the program into well-understood components. Then, reusing the components, the normalizing Xcerpt program could be composed into its original shape and be used by an Xcerpt engine.

Summary (I4)

- I4 has better understanding of the composition techniques than the other working groups, due to more involvement with I3.

- While the composition tooling is in place for Xcerpt, more concrete Xcerpt examples are needed to investigate the usefulness of composition. We are looking for proof of concept.

- How Xcerpt can support composition natively should be investigated.

- More research should be directed towards composition for type system, thus also realizing the long term goal of the I3 WG.

3.2.5 Evolution on the web (I5)

I5 develops a framework for Event-Condition-Action (ECA) rules for a reactive web. Under this umbrella topic, there is work going on with the actual ECA framework and, quite separately, with an existing ECA rule language, XChange [7]. XChange in turn is an extension of the XML query and transformation language Xcerpt [2], developed by I4.

Regarding composition, the ECA framework mostly needs language composition, since it wants to allow any component language (namely, event-, condition- and action-languages) to be plugged-in. Apart from composing languages, the execution of the ECA framework calls for interoperability between the engines and reasoners of the different component languages. E.g. the engine of an event language needs to communicate with the engine of a condition language (for checking the conditions) and so on. Within the ECA framework, the underlying hypothesis is that XML markup of code fragments of the component languages and variable bindings makes up the communication process between the engines.

The interoperability requirements for the ECA framework of I5 have not yet been addressed by I3. However, I3 has sent a student to work with I5 and investigate their needs in regards to interoperability. However, the final results from that investigation are not yet available at the time of writing. Furthermore, there has already been activities planned towards improving the communication situation between the groups (e.g. during the REWERSE supported summer school taking place in Lisbon in September 2006).

As noted above, the ECA rule language XChange is also part of I5 and has different composition requirements than the ECA framework. XChange is a language for evolution of data and reactivity on the Web. XChange borrows some of its syntax and functionality from the XML query language Xcerpt. Thus, to a large extent, requirements between Xcerpt and XChange are shared.

For XChange, the requirements for composition lies closer to the composition techniques currently targeted by I3; program composition and reuse of code. Thus, code of XChange rules can be considered for reuse and componentization. The following entities of XChange programs can be considered for composition:

- Programs (rule sets), i.e., what specifies the behavior of an reactive web node (in terms of reacting to incoming event messages).
- ECA rules, which are composed of event-, condition- and action-queries.
- Event queries, i.e., queries to incoming event messages.
- Condition queries, i.e., queries to (persistent/"normal") web data (cf. Xcerpt).
- Actions, which can be sending new event messages or updates to persistent web data.

Use case: Eighteenth Century Studies Society

Working group I5 has developed a use case scenario involving several distributed web resources of the Eighteenth Century Studies Society (ECSS), a fictitious historical scientific community. Each resource is locally administrated and may receive events from other resources, which then might cause updates of data to be required. These re-actions, caused by events, are implemented as XChange rules. The ECSS demo is distributed over six web nodes and has a total of about 120 rules; a third or a half of them do not contribute to the actual functionality but implement

just logging services to make visible what is going on in the demo. These rules have a high degree of overlapping, because the same functionality is found at different web nodes (e.g., each university node has rules for managing their employees, basically the same at each node). There are roughly 20-30 different rules wrt. functionality. While different nodes have similar rules, there are also differences in the rules. E.g., the data sources queried by each web resource have different URIs, but might be structured in the same way. The XChange demo for the I5 group might serve as an interesting example, and a case study, of how to compose XChange rules. It would give the two working groups of I3 and I5 a hands-on scenario to discuss composition around, as requested by the survey participants (see Section 3.1).

Summary (I5)

- There is a gap between the working groups in terms of what they understand of each others technology.
- The ECA framework calls for interoperability (run-time) issues more than code composition (compile-time). Interoperability issues have not yet been addressed by I3.
- XChange, as used as part of the group, has similarities to Xcerpt and ideas found for composition in one is likely to be reusable in the other.
- A good use-case, and starting point, for composing XChange programs has been found in one of the demo application of I5: ECSS.

4 Future work

The survey in Section 2 and the collected results, as summarized in Section 3, relate to fairly general requirements on composition. A reason for this was also to collect information on the level of understanding our partners currently have about the composition techniques used by I3. This level of investigation was considered suitable at the time of writing. However, more specific requirements have to be identified in the future. To this aim, one possibility is to use a tool like *Focal Point*[5].

Focal Point supports the traditional storage and management of requirements, but adds several features for prioritizing and planning. The central prioritizing algorithm builds upon the Analytical Hierarchy Processing (AHP) approach, where requirements are prioritized pair wisely and their overall priority can be calculated on a rational scale. Pair wise prioritizing has proven to be an efficient way for human users and/or committees to perform the prioritizing, not the least as a basis to discover consistency problems and disagreements in the analysis team. Focal Point also provides services to compare prioritations on different criteria, such as importance for different markets. As a result the analyst can start developing requirements that are of best use to many markets and save the more niched requirements to later releases. Figure 1 shows Focal Point in action, which plots (hypothetical) features of a mobile phone for the US and German markets. In REWERSE we have the opportunity to use a demonstration licence as a repository and structuring tool for requirements on a composition system. Up to 20 users can enter and modify requirements simultaneously and the composition system management can use planning tools to determine the prioritations of research efforts. The licence is valid until September 2007.

[5] http://www.telelogic.com/corp/products/focalpoint/index.cfm

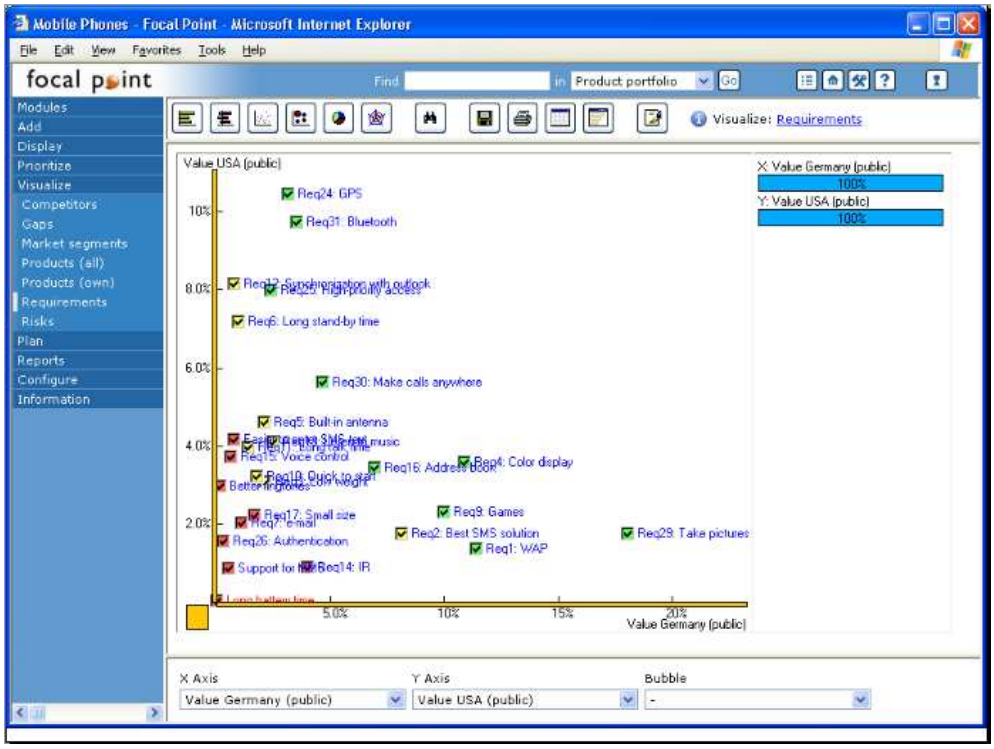

Figure 1: Focal Point plotting hypothetical features of a mobile phone for the US and German markets.

5 Conclusions

From the collation of the composition requirements and the results of the survey in Section 2, it is clear that composition is seen as an important and useful part in the work of the collaborating groups within the REWERSE network.

For applying composition on the languages and applications of the collaborating working groups within REWERSE, better understanding of each others work is needed. Thus, closer collaboration should be conducted with interested partners. However, it should be noted that attempting to collaborate with too many parties, unless the required man power is available, is likely to have a negative effect rather than a positive one.

It has been clear from this report that what is urgently needed is *proof of concept* wrt. the composition of I3. Toward this aim, some working groups should be selected for closer interaction and collaboration. Within these collaborations, practical use cases and scenarios should be studied in order to come to a conclusion on what can be achieved there wrt. composition.

The composition technologies from I3 are innovative and still in an experimental phase. Nevertheless, there are promising results that have to be considered further. One such aspect is the integration of these composition technologies with important trends coming out from work on the Semantic Web, such as Web Services and ontology merging and alignment. It should be mentioned that I3 has, from the start, had the objective of targeting the areas of Web Service

composition and ontology composition. Therefore, it is good time to start to investigate the applicability of our technologies in these areas, which require further studies and considerations. Closer integration with the working group A3 (Personalization) can help in achieving this goal.

The I3 working group within REWERSE works both on typing and composition. It is the long-term goal of the working group to finally merge these two fields in some way, where one can benefit from the other. There are two ways this can be achieved.

1. **Type-check composition** Typing work within I3 is largely focused on providing a type system for Xcerpt. I4 has shown interest (see Section 3.2.4) in having Xcerpt natively support composition. One can in such a scenario also think about how the typing of Xcerpt could be used to verify type-safeness of compositions.

2. **Composing types** As also noted in Section 3.2.4, I4 highlights the need for composing types and type systems. Therefore, it should be investigated to what degree the long-term goal of integrating types and composition within I3 can be realized this way.

Acknowledgement

We want to thank the reviewers for giving important and constructive critic making this a better report.

This research has been co-funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. `http://rewerse.net`).

References

[1] Andreas Doms and Michael Schroeder. GoPubMed: exploring PubMed with the Gene Ontology. *Nucleic Acids Research*, 33:W783, 2005.

[2] Francois Bry and Sebastian Schaffert. The XML Query Language Xcerpt: Design Principles, Examples, and Semantics. In *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems*, pages 295–310, London, UK, 2003. Springer-Verlag.

[3] Gerd Wagner and Adrian Giurca and Sergey Lukichev and Grigoris Antoniou and Carlos Viegas Damasio and Norbert E. Fuchs. Language Improvements and Extensions. Research report IST506779/Cottbus/I1-D8/D/CO/a1, Institute of Informatics, Brandenburg University of Technology at Cottbus, 2006. REWERSE Deliverable.

[4] Mattheo Baldoni and Christina Baroglio and others. Thread on Testbeds. Research report IST506779/Turin/A3-D2/D/PU/a1, Dipartimento di Informatica, Universita degli Studi di Torino Italy, 2005. REWERSE Deliverable.

[5] Nicola Henze and Marc Herrlich. The Personal Reader: A Framework for Enabling Personalization Services on the Semantic Web. In *Proceedings of the Twelfth GIWorkshop on Adaptation and User Modeling in Interactive Systems (ABIS04), Berlin, Germany, 2004*, 2004.

[6] Nicola Henze and others. Testbeds II - Early Prototypes. Research report IST506779/Hannover,Torino/A3-D6/D/PU/b1, Research Center L3S and ISI- Knowledge-Based Systems, University of Hannover, Germany, 2006. REWERSE Deliverable.

[7] P.-L. Pătrânjan. *The Language XChange: A Declarative Approach to Reactivity on the Web*. Dissertation/Ph.D. thesis, Institute of Computer Science, LMU, Munich, 2005. PhD Thesis, Institute for Informatics, University of Munich, 2005.

[8] Scott Boag and Don Chamberlin and others. XQuery 1.0: An XML Query Language. W3C Candidate Recommendation, 8 June 2006. Available at `http://www.w3.org/TR/xquery/`.