What *Is* the Goal?

Agile elicitation of semantic goal definitions using wikis

David Lambert, Stefania Galizia, and John Domingue

Knowledge Media Institute, The Open University, Milton Keynes, UK
{d.j.lambert,s.galizia,j.b.domingue}@open.ac.uk

Abstract. Formal goal and service descriptions are the shibboleth of the semantic web services approach, yet the people responsible for creating them are neither machines nor logicians, and rarely even knowledge engineers: the people who need and specify functionality are not those who provide it, and both may be distinct from the semantic annotators. The gap between users' informal conceptualisations of problems and formal descriptions is one which must be effectively bridged for semantic web services to be widely adopted. We show how a simple technique—using a wiki to collect user requirements and mediate a progressive, iterative refinement and formalisation of user goals by domain experts and their knowledge engineer colleagues—can achieve this. Further, we outline how the process can be automated, so as to itself benefit from semantic technologies.

1 Introduction

Service oriented computing (SOC) offers a promising new approach to programming, resource sharing, and organisational collaboration. Semantic web services address several of the problems SOC faces as the number and complexity of services grows, such as finding appropriate services, composing, and invoking them correctly. But the mechanisms used to enable this magic require formal, logical specifications of user goals and the web services that can satisfy them.

We are currently working with biomechanics researchers who have chosen semantic web services as the best platform to support their work. In this context, we faced the problem of capturing the users' notions of their goals, and translating them to formal representations. These formalisations, for the static Semantic Web as well as Semantic Web Services, are far from intuitive. Indeed, the 'call for papers' for this very conference offered this gem:

> Authors of accepted papers will be required to provide semantic annotations for the abstract of their submission for the Semantic Web (help will be provided for this task).

which would fairly entitle our medical colleagues to demand of us "Physician, heal thyself!". In our case, we have tried to bridge the chasm with a methodology where domain experts can express their requirements in natural language and,

through interaction with a semantic web expert mediated by a wiki, progressively refine their goal into one expressible in a formalism suitable for use by semantic web services.

We review the context of the work in the next section, then examine the problem of goal conception and description for users in section 3. In sections 4 and 5 we present our solution and a worked example of the method, respectively. Section 6 outlines the future direction of the work. Related research is discussed in section 7, and we conclude in section 8.

2 Background

In this section, we recount a short history of the two sides of our problem, as well as the source of our solution. First, we introduce our application domain, an ongoing programme to develop web services for use in a biomechanics application. Section 2.2 reviews semantic web services, noting why they have been selected as the most promising solution for our application. Finally, in section 2.3 we look at the existing software process for LHDL, with which they were comfortable and wished to use to develop semantic web services goals.

2.1 The Living Human Digital Library

The creation of in-silico models of entire organisms has been identified as a 'Grand Challenge' problem [1] for informatics, and several projects have begun working towards the construction of multi-domain, multi-scale models. Our work concerns one such project, the 'Living Human Digital Library' (LHDL) [2], which intends to lay a technical foundation for virtual physiomes by first developing techniques and infrastructure for distributed modelling and analysis of the human musculoskeletal system.

For the immediate purposes of supporting LHDL, web services are appropriate: they address the need for distributed, autonomous provision and invocation of computational services and data storage facilities that the web services approach provides. Longer term, simulations of entire physiomes will require integration across scales and between disciplines (e.g. chemistry, biomechanics, clinical) and sub-systems (e.g. neurological, renal, cardiac). These programmes are about coordination: the intention is not to create a single federation of services that define a single virtual physiome, but rather a framework to enable the integration of services to suit particular requirements—even to the point of modelling individuals for clinical purposes. As the number of services available for use, and the number engaged in any one simulation, increase, it will become infeasible to manage them manually. With the future in mind, LHDL is investigating semantic web services as the most promising technological solution.

2.2 Web services and semantics

Service-oriented computing [3], and especially web services [4], have forced a paradigm shift in computing provision. They enable computation to be dis-

tributed, and easily invoked over the internet. 'Virtual organisations' of services can be constructed for tasks the component services were not designed for. However, as services become more complex, and their numbers increase, it becomes more difficult to comprehend and manage their use. Tasks such as service discovery, composition, invocation, process monitoring and fault repair cannot be successfully automated for web services, because the descriptions involved are only syntactic, and require human engineers to interpret them. Semantic web services [5] add rich, formal semantics to enable this automation. By modelling the purpose and interfaces of the services in logical formalisms such as description logics [6] or abstract state machines, we allow machines to reason in powerful ways about the services in ways that otherwise must be done by humans, or are simply too expensive to be done at all.

The Web Services Modelling Ontology (WSMO) [7] is a leading framework for semantic web services. Its four key concepts of domain ontologies, goals, web services, and mediators evidence its commitment to separation of concerns. WSMO insists on a clear distinction between user goals and their realisation by web services, thus enabling capability-based invocation. The user's needs and context are given first-class status in the modelling process, while intelligent middleware can determine how to satisfy a user's goal with the services available to it. Similarly, the necessary loose-coupling of services, goals, and ontologies is handled by the systematics use of mediators, which intervene in several places where otherwise heterogeneity would cause incompatibility. Between ontologies, *OO-mediators* perform ontology mapping wherever necessary; *WW-mediators* allow web services to interact correctly, primarily addressing choreography mismatches; user goals are mapped to web services by *WG-mediators*; and *GG-mediators* allow the creation of new goals by composing others.

Our WSMO implementation is the Internet Reasoning Service (IRS) [8], a general-purpose semantic services platform which has been used in several domains including business process management, e-learning, and e-government. In its current implementation, it adopts and extends the epistemological commitments of WSMO. Its internal representation format is OCML [9], a frame based knowledge modelling language. The IRS can invoke web services exposed via SOAP or XML-RPC, and export legacy Java and Common Lisp code as web services by automatically generating wrappers. Goals can be executed by sending SOAP messages or making HTTP GET requests, thus supporting the REST paradigm. A process of 'elevation' deals with mapping the XML messages of services to internal ontological representations expressed in OCML.

2.3 LHDL's existing software development process

Even as LHDL moves towards a web-based infrastructure, the project must continue to support the development of the legacy client software. For some time the LHDL members responsible for the LhpBuilder software (covered in section 3.1) had been successfully using agile development methods, and wanted to retain them.

Agile development [10] is a software development philosophy which emphasises people and communication over (usually heavy-weight) processes. There are several flavours of agile development, but they agree on the following 'agile manifesto' (`http://agilemanifesto.org/`):

- individuals and interactions over processes and tools
- working software over comprehensive documentation
- customer collaboration over contract negotiation
- responding to change over following a plan

These principles are typically realised in the following ways:

- the writing of use-case 'stories' which capture a facet of functionality that the customer describes in their own terms, and that become specifications for the software developers
- rapid turnaround, where users see their requirements implemented within weeks, fostering trust between customer and engineers
- emphasis on working, executable code instead of design documents
- simple solutions, which should never be more complicated than the current requirements necessitate
- continuous improvement, including refactoring, lessens the cost of future development
- test-driven development, applying automated tests to code

In this paper, we are particularly interested in the first two points, since these are the aspects of agile development most concerned with requirements specification.

In LHDL, domain experts and software developers used wikis to develop and record the use-cases. Wikis [11] are websites where the content is user-editable. Wikis lower the bar for generating web content by both providing a simplified language for data entry, and sidestepping bureaucratic control of websites. The wiki engines which drive them often provide additional functionality such as versioning and notification. They are frequently used to support community websites, like BiomedTown, since they support a very collaborative workflow. Users can add their own material and edit the work of others, and the iterative, distributed efforts of many users—often experts—can quickly lead to impressive content.

3 What is involved in creating goals?

Having established that semantic web services are an appropriate way to attack the problems LHDL has set out to tackle, we face a new inconvenience: how can users who are not IT-experts construct the formal goal definitions? In this section, we examine the user's and then the middleware's perspectives on semantic web services, and then present criteria for reconciling the two in the context of LHDL project.

3.1 The user's view

The user experience in LHDL is mediated by the LhpBuilder and a community website, BiomedTown (`www.biomedtown.org`). The community services include forums, wikis, mailing lists and file storage, and are accessed via a web browser. The principle desktop tool is LhpBuilder [12], a legacy application which enables a user to create, store, and manipulate Virtual Medical Entities (VMEs). VMEs are collections of data such as MRI images, gait analysis data or finite element analysis results. LhpBuilder can perform operations such as extracting two-dimensional slices from volume data, virtual palpations, or combining motion-capture data with bone images.

Some of the tasks a user may wish to carry out include: registering as a member of BiomedTown (for any of several projects hosted there); searching and retrieving data resources; using data resources within LhpBuilder; creating new data resources by editing existing ones, or by defining processing pipelines on existing data; importing and exporting data resources from LhpBuilder; uploading data objects to the repository; and adding meta-data to stored data objects. These tasks are defined as 'stories', written by the users, and stored at BiomedTown.

There are different classes of users, who have different relationships with the goal generation processes. Most users will simply use existing goals, often without realising that they are goals: for example, by submitting a normal web form, or by invoking some functionality through LhpBuilder which is implemented through semantic services. Another class of users will go to the lengths of suggesting or requesting new goals, but will not take part in seeing them through the specification process. Those who actively participate in the generation of goals will be a small minority. Even these practitioners, who are technically savvy and familiar with particular computational tools of their trade, do not typically write Perl programs, as may bioinformaticians working in genetics or proteomics, nor are they familiar with the logical languages used on the semantic web.

3.2 The machine's view

Semantic web services require several components, which in the case of the WSMO framework, include the following:

- *user goal description*
- *domain ontologies*
- *web service description* description of web services
- *mapping goals to web services* either directly or using composition
- *identifying mediator requirements* mismatches between ontologies, goals, and web services identified and dealt with

of which only the first two should be of interest to the typical user, and we will only consider the first here. WSMO, and hence IRS, impose a strict division between goal and service. This allows us to explicitly model the user's needs,

without regard to how it might be implemented. This allows the middleware to better understand the context of a goal invocation, and flexibility in how to satisfy it. An IRS goal consists of several components:

name which identifies the goal
superclasses which may anchor the goal in a goal taxonomy
inputs the parameters passed to the goal
output the returned value
capability which is a context in which the goal is applicable

Goals may have several superclasses, so the taxonomy is a graph, not a tree. Inputs and outputs are named parameters, and each is typed by association with a concept from an appropriate domain ontology. The capability in turn is expressed by four kinds of axioms:

- *preconditions* conditions on the inputs that must be met for the goal to execute
- *postconditions* conditions on the output that must be met for the goal to complete
- *assumptions* conditions in the world which should hold true before invoking the goal
- *effects* conditions in the world which should be true after the goal completes

Preconditions and postconditions can be verified at invocation time by the middleware or the services themselves. Assumptions and effects are predicates on a world state which cannot be easily verified by the middleware or services at run time, and which may be unverifiable in principle. All four are sentences in restricted predicate logic, and all are optional (or true by default, whichever interpretation suits).

A goal definition in IRS's internal representation language of OCML, and a corresponding graphical representation are shown in figures 2 and 3 respectively.

3.3 Requirements for a goal formalisation process

Given the discrepancy between users who can describe their goals informally and perhaps imprecisely, and the representation required by semantic middleware, we required a process that meets the following criteria:

1. *Perform requirements capture* We are concerned not just with generating the formal goal, but with the very act of discovering what the user wants.
2. *Generate formal goal descriptions* Identification and description of semantic goals using requirements docs. necessary domain ontologies created or reused.
3. *Generate natural-language documentation* Not only are formal descriptions hard to write for non-specialists: they are not much easier to read.
4. *Easy to use* Users must be comfortable with the process itself.

5. *Fit well with current practice.* The users have a methodology which worked well for the non-web services version of the software and which they intend to use as they move to web services. They are happy with the results, and comfortable with the process.
6. *Support distributed development.* The teams responsible for LhpBuilder and the semantics are geographically separated, so collaboration must work at a distance. This will often be the case in SOC environments, since one of SOC's key features is its distributed nature.

4 A lifecycle for agile goal specification

Our solution is iterative collaborative refinement of goals, mediated by a wiki. Just as wikis simplify the HTML notation of websites, so we use a wiki to simplify the entry of goals. Where the wiki engine turns simplified markup into HTML, we use the intervention of ontology engineers to refine the informally stated, natural language requirements into OCML ones. The lifecycle then looks like this:

1. User conceives task and develops story
2. User enters natural language goal definition in wiki
3. Knowledge engineer clarifies the natural language
4. User agrees or refines this new definition
5. Knowledge engineer creates the formal goals, retaining the natural language as documentation

In actual use, the process will involve more iteration, sometimes a substantial amount, depending on circumstances.

The user's initial goal descriptions are lodged in terms of natural language descriptions. For instance, a user might say that they want to search for VMEs. We use a template to structure the definition (see figure 1 for a completed example). The distinction between precondition/postcondition versus assumption/effect is not only often subtle and difficult for domain experts to comprehend, it can also be an arbitrary distinction, since it depends on how the interface develops. This requires input from the engineer as well as the user, and emerges in the process. Initially, we just ask for 'before' and 'after' conditions.

Following submission, a semantic web services expert reviews the goal, refining it by making the types and conditions more concrete (i.e. aligning it with the current ontology). The goal may suggest a class of goals which are best separated, in which case the engineer can split the goal into several pages and proceed with each.

The domain ontology (or ontologies) may also require extension or revision in the light of the developing goal. The domain ontology can usefully be inspected in a graphical format by the domain expert, to ensure the correct terms are being used.

At this point, the engineer has essentially formalised the goal, but checks with the user via the formalised natural language. If this is correct, the engineer

proceeds to a fully formal representation but retains the natural language definitions as comments. This provides documentation, which can be hyperlinked to other pages in the wiki. This can also be used as a 'cookbook' by semantic engineers when they construct other goals.

5 Example

In this section, we illustrate the process of requirements elicitation and goal formalisation for an LHDL project goal. We use the example of a user requirement to find the URLs of VMEs which match given search criteria.

The user begins by filling the template form: figure 1 is a goal showing the use-case story. The ontology engineer begins by creating a new goal class, `search-goal`. The user seems to want several kinds of goal, searching by one of several criteria such as donor attributes, data type, or VME attributes, or creation attributes. The engineer divides them out into separate pages, linked from the general `search-goal` superclass's page. Common to all, however, is that every goal returns a list of URLs: this can be recorded on the top-level goal's page. We will focus here on searching by acquisition attributes.

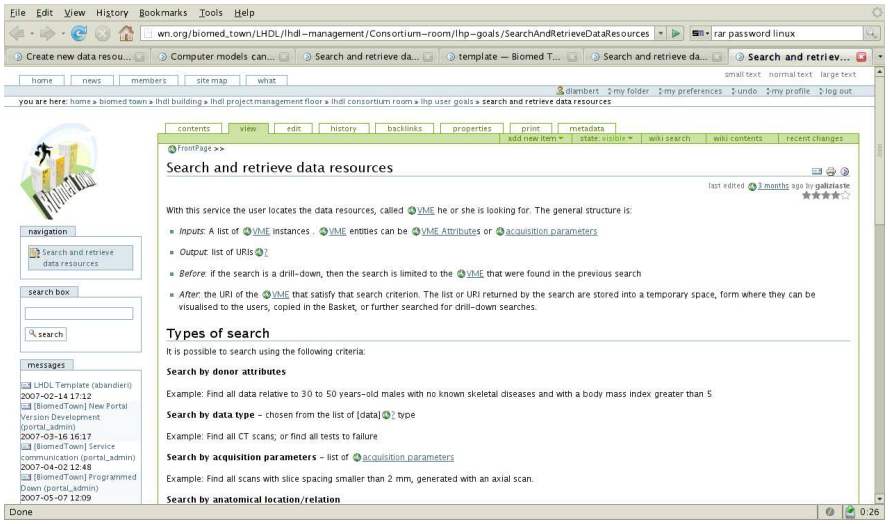

Fig. 1. Wiki page with a goal in development. Note that some parameters have been given types and are hyperlinked to the relevant pages.

The user's story for this particular goal type says the following:

Example: Find all scans with slice spacing smaller than 2mm, generated with an axial scan.

The search is expressed by a list of criteria which must be true of each URL returned. Again, another page is built where the engineer can develop and explain the a search filter for acquisition data:

```
(defclass acquisition-filter ()
  ((slice-spacing-max :type float)
   (slice-spacing-min :type float)
   (scan-type :type scan-type)))
```

But this is explained to the user in the following terms:

The user creates a search filter object with field which reflect maximum or minimum values that are acceptable for VMEs.

At this point, or perhaps after some iterations in which the user and engineer reach agreement via English, the OCML descriptions are in place. The result is fully formalised:

```
(defclass search-goal (lhdl-goal) ?goal
  (output-role :type (list-of vme-url)))

(defclass search-by-acquisition (search-goal) ?goal
  ((input-role acquisition-filter :type acquisition-filter)
   (has-postcondition
    (kappa (?goal)
           (and (has-value ?goal acquisition-filter ?filter)
                (has-value ?filter slice-spacing-max
                           ?slice-spacing-min)
                (has-value ?filter slice-spacing-min
                           ?slice-spacing-max)
                (has-value ?filter scan-type ?scan-type)
                (has-value ?goal output-role ?urls)
                (forall ?url ?urls
                        (and (<= (value ?url slice-spacing)
                                 ?slice-spacing-max)
                             (>= (value ?url slice-spacing)
                                 ?slice-spacing-min)
                             (= (value ?url scan-type)
                                ?scan-type)))))))))
```

Fig. 2. The search goal in OCML.

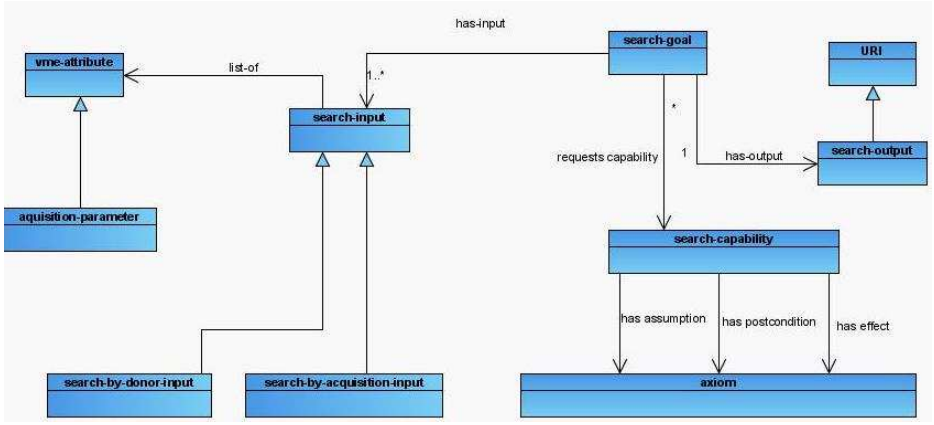

Fig. 3. An intermediate depiction of the search goal as UML.

6 Future development

We have used this method successfully to produce real goal definitions and built services to support them, but there is obviously scope for enhancement.

Most conspicuous is the absence of semantics, the use of which would open several options. The process could be partially automated and brought within the semantic web services umbrella. An obvious integration would be with semantic wikis, in which the final ontological goal descriptions are stored in a knowledge base and intelligently extracted into the wiki as required, instead of being merely presented as text in the wiki [13]. Goals could be categorised simultaneously goals at the wiki and semantic levels.

The larger granularity of web services makes it likely that case-based reasoning and computer-aided software engineering (CASE) might be more applicable.

If we reexamine the agile manifesto in section 2, we see that we have not addressed all the points. Without pushing the analogy too far, we can ask what it would mean to have 'working code': this might correspond to having the formal definitions stored in a reasoner which would continually check for consistency (and refactoring could be partially addressed by checking for redundancy).

Similar problems confront those creating service descriptions. Although service builders are likely to be software engineers and therefore might be expected to be more familiar with formal notations, they may still need help with particular formalisms like WSMO. Wikis provide a convenient meeting place for software engineers and semantic web services 'consultants'.

7 Related work

The semantic services literature is replete with work on service descriptions and useful, machine-reasonable semantics [14,15] and how to attach them to directory services [16,17], but the question of where the semantics themselves come from is largely ignored. Most of the talk is of describing services, or discovering them, not defining users' intent.

The IRS was previously used in MiAKT [18], brokering the invocation of services for medical imaging. The two best-known bioinformatics projects using web services are myGrid and BioMOBY. myGrid [19] is an on-going project which provides bioinformaticians with workflow tools which can alleviate the chores of manually discovering genome-related web services and data stores, and the subsequent programming to invoke them. They essentially worked backwards from already implemented services, annotating them and then using the annotations to constrain (by reasoning over input/output types) and suggest workflow construction (services were also (coarsely) categorised by task type). In the myGrid project, DAML+OIL was initially used [20], but moved to using an extended RDF [21]. In particular, they note that DAML-S does not intrinsically support task typing. This is a disadvantage, because users think more along the lines of tasks they must complete, and not about the inputs and outputs to them. They have also looked at the question of workflow discovery [22]. Where myGrid has generated third party annotations of existing, non-semantic web services, the BioMOBY [23] project set out to create a unified ontology, with services strictly adhering to the standard terminology and XML message structures. Despite the ontology itself being developed collaboratively, in an 'open source' way, this approach precludes incorporation of legacy services and third-party annotation.

myGrid and BioMOBY are targeted at the genetics and molecular biology communities where practitioners had long used scripting languages to call web services. They are thus not addressing the goal formulation problem to the same extent, since the users have already mostly formulated them, and have practice in refining them to an executable form, as well as being more conscious of what services are available. Even then, in both projects, familiarity of the practitioners with the ontology languages was considered more important than their expressivity. LHDL has a commitment to applying a comprehensive semantic web services framework in a domain where there has previously been little use of web or grid technologies. The goals and practises for the new computational environment are naturally less developed, and requirements elicitation plays a more prominent role.

8 Conclusions

The LHDL project is driven by researchers in biomechanics who have opted to use semantic web services technologies to simplify the provision and use of their computational and data services. They must specify semantic web services goals, but are not experts in the relevant formalisms. This problem has been largely

ignored in the literature, but threatens to be a bottleneck as demand for semantic web services increases from the small number currently built by semantic web researchers. The pragmatics of collecting these goals is not well explored in the semantic web, and ours is just one solution of what will surely be many.

In our approach, we closed the gap by using a wiki to mediate communication between domain experts and knowledge engineers, allowing the progressive formalisation of goals initially expressed in natural language. Since the BiomedTown citizens were already using the wiki to record use-cases for their agile development process, it was a natural step to adopt the wiki for goal requirements recording, and then further to perform the 'agile development' *in* the wiki. The wiki's normal function as a communal blackboard means the final definitions can be annotated by the users.

The point of the semantic web, of course, is to give the machine a greater understanding so that it can reason about our problems and provide intelligent assistance. We plan to implement this technique as a workflow within our web services platform, and offer more hints from the middleware, both to the domain experts and engineers.

Acknowledgements

This work was supported by the Living Human Digital Library project, European Union programme FP6-026932.

References

1. Sleep, R.: in Vivo—in Silico: First Steps. In: Grand Challenges in Computing (GCC '04) Conference. (March 2004)
2. Viceconti, M., Taddei, F., Van Sint Jan, S., Leardini, A., Clapworthy, G., Galizia, S., Quadrani, P.: Towards the multiscale modelling of musculoskeletal system. In: Bioengineering modeling and computer simulation. (2007)
3. Papazoglou, M.P.: Service-Oriented Computing: Concepts, Characteristics and Directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE 03). (2003)
4. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1 (2001)
5. McIlraith, S., Son, T., Zeng, H.: Semantic web services. IEEE Intelligent Systems (2001)
6. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic markup for web services (2004)
7. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services. Springer (2006)
8. Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V., Pedrinaci, C.: IRS-III: A Broker for Semantic Web Services based Applications. In: Proceedings of the 5th International Semantic Web Conference (ISWC2006), Athens, Georgia, USA (2006)

9. Motta, E.: An Overview of the OCML Modelling Language. In: 8 th Workshop on Knowledge Engineering: Methods & Languages KEML 98. (1998)
10. Highsmith, J., Cockburn, A.: Agile software development: the business of innovation. Computer **34** (September 2001) 120–127
11. Leuf, B., Cunningham, W.: The Wiki Way: Quick Collaboration on the Web. Addison-Wesley Longmann (2001)
12. Van Sint Jan, S., Viceconti, M., Clapworthy, G.: Modern visualisation tools for research and education in biomechanics. iv **00** (2004) 9–14
13. Fischer, J., Ganter, Z., Rendle, S., Stritt, M., Schmidt-Thieme, L.: Ideas and Improvements for Semantic Wikis. (2006)
14. Paolucci, M., Soudry, J., Srinivasan, N., Sycara, K.: A Broker for OWL-S Services. In: Proceedings of the 2004 AAAI Spring Symposium on Semantic Web Services. (2004)
15. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology (2003)
16. Miles, S., Papay, J., Payne, T., Decker, K., Moreau, L.: Towards a Protocol for the Attachment of Semantic Descriptions to Grid Services. In: Proceedings of The Second European across Grids Conference, Nicosia, Cyprus. (2004)
17. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Importing the semantic web in UDDI (2002)
18. Shadbolt, N., Lewis, P., Dasmahapatra, S., Dupplaw, D., Hu, B., Lewis, H.: Mi-AKT: Combining Grid and Web Services for Collaborative Medical Decision Making. In: Proceedings of The UK e-Science All Hands Meeting 2004. (2004)
19. Stevens, R.D., Robinson, A.J., Goble, C.A.: myGrid: personalised bioinformatics on the information grid. Bioinformatics **19 Suppl. 1** (2003) i302–i304
20. Wroe, C., Stevens, R., Goble, C., A., R., Greenwood, M.: A Suite of DAML+OIL Ontologies to Describe Bioinformatics Web Services and Data. International Journal of Cooperative Information Systems **12**(2) (2003) 197–224
21. Lord, P., Bechhofer, S., Wilkinson, M., Schiltz, G., Gessler, D., Hull, D., Stein, C.G.L.: Applying semantic web services to bioinformatics: Experiences gained, lessons learned. In McIlraith, S.A., Plexousakis, D., van Harmelen, F., eds.: The Semantic Web — ISWC 2004. Volume 3298 of Lecture Notes in Computer Science., Springer (November 2004) 350–364
22. Goderis, A., Li, P., Goble, C.: Workflow discovery: the problem, a case study from e-science and a graph-based solution. In: International Conference on Web Services (ICWS '06). (September 2006) 312–319
23. Wilkinson, M.D., Links, M.: BioMOBY: An open source biological web services proposal. Briefings in bioinformatics **3**(4) (2002) 331–341