Managing Software Engineering Artefact Metadata

R.G.Dewar

Department of Computer Science, Heriot-Watt University, Edinburgh, UK

rick@macs.hw.ac.uk

Abstract. This article describes a proposed P2P-based environment called SESAME to manage semantically enhanced artefacts from the software engineering domain. We argue that such artefacts, for instance code, models and documents, are not currently semantically enhanced (a state we call *premantic*) and that there are few tools that could manage and use such metadata to derive some added value for searching, sharing and organisation. Yet, semantic-based services can aid resource discovery and provide advanced information management. These services depend on the quality of metadata extracted, however, to manually provide this metadata is an expensive process. Current tools that automatically acquire, process and relate semantics to content are limited. Therefore, the SESAME project will develop an open architecture and toolset to manage the elicitation and configuration of artefact semantics; specifically in the domain of software engineering.

1 Introduction

With software processes becoming increasingly complex, there is a greater demand for tool support throughout the software-lifecycle. All software tools, regardless of their use, produce software artefacts and for combinations of tools to be truly effective, they must work together [2]. However, toolsets can be diverse and, despite almost two decades of research and practice after Wasserman's seminal work, their integration is currently piecemeal at best [18]. Tools often produce artefacts in proprietary formats, which further restrict intercommunication.

Content-specific metadata presents additional benefits in an open software development environment. Semantically enabled services could use information regarding artefact quality, ownership and purpose to assist artefact management and maintenance, re-use decisions, plagiarism and searching within a complex component-based or model-driven information space. Additionally, traceable relationships that define connections between artefacts could be established to group and organise complex software projects using semantics.

However, a large number of software artefacts are not currently semantically enhanced; here we call their impoverished state *premantic*. And the semantic payloads of these artefacts possess considerable untapped resources that could benefit the software engineering process. Ontology exist that describe certain content, but currently tool support for using these ontology to create semantic descriptions is limited. There are also few tools that manage and use semantically enhanced content to derive some added value; for instance searching, sharing and linking.

In addition, software engineering tools and frameworks have often been criticised for being too tightly coupled to other components and the data. This coupling has resulted in an unacceptable implementation overhead and lack of flexibility; in turn these problems have reduced take up and impact. P2P technology offers lose coupling between artefacts, metadata and tools, but has not been fully exploited in this domain as yet.

We begin this article by considering background issues relevant to our arguments, before we present a motivating scenario and then go on to describe the proposed architecture for SESAME

(Software Engineering Artefact Management Environment) – not to be confused with the TripleStore RDF Database of the same name [22].

2 Background

2.1 Integration in Software Engineering

Web-integrators such as SourceForge [3] provide a comprehensive software development environment for collaborative efforts in an Internet environment. Such environments offer useful functionalities such as CVS, project administration, document management, mailing lists, bug tracking and file publishing. Nonetheless, although popular, these closed environments do not provide semantic enhancements to artefacts or extensibility for users or their communities.

The rise and rise of Eclipse [4] as "a kind of universal tool platform" has seen the possibility of platform extensibility through its plug-in capabilities. Furthermore, distributed collaboration is commonly achieved through the use of Eclipse and the CVS repositories of SourceForge. However, such integration still currently lacks semantic enhancement. In other words, artefacts are predominantly premantic; save for the occasional natural language descriptions added to CVS version tags.

Of course, studies that look at the semantics of software engineering artefacts are not new. Cattaneo et al. [5] consider the creation and management of the semantics of documents associated with software engineering projects. They also address issues of notification and consistency, but the approach does not consider the possibility of automatic generation of meta-data and relationships and omits any notion of standard ontology for the domain. Similarly, Olsen and Grundy [6] stop short of automation and ontology.

On the other hand, Sherba and Anderson [7] do explore the possibility of generating relationships by programmatically extracting and linking common terms (e.g. method names) from different artefacts.

2.2 Ontology-Based Software Development Environments

Having noted the omission of ontological consideration in some earlier work, we are now beginning to see some efforts in this area.

For instance, Falbo et al. [8] are working towards an Ontology-based software Development Environment (ODE), comparable to a Semantic Web, where software engineering knowledge is organised and accessible to developers and tools. They concentrate on exploiting metadata for domain engineering but currently lack tool support for automated semantic creation and management.

Furthermore, the W3C's Software Engineering Task Force [9] have stated their intent to evaluate ideas for Ontology Driven Architectures (ODAs). To this end, recent work by Knublauch [10] has focused on an ODA for Web services and agents for the semantic Web. Here the ontology do not describe software engineering artefacts; the example scenario in the paper uses the tourism domain. However, the work shows that software architectures can be influenced during their development (not just at run time) by an extant explicit ontology. Similarly, Oberle et al. [11] have considered the impact of ontology on application servers. They have implemented a hierarchy of ontology where the most abstract describe software components, which are then specialised by a profile of that component and its API. The hierarchy then specialises down to a specific domain ontology. This ontological hierarchy, Oberle et al. argue, simplifies the development and maintenance of software components.

2.3 Metadata Modelling, Generation and Management

It is interesting to note, we could consider the UML [12] as an ontology language where the class diagram might be employed to model the domain; particularly when it is used with OCL. Currently the OMG have a live request for proposals to map the UML and OWL [13]. As such, the UML may yet prove to be the ontology language of choice for software engineering since it is already well know by domain experts. However, the richness and power of OWL would be a more conventional choice at this stage. For this reason, there has been a Request For Proposal concerning the Ontology Definition Metamodel [25]. This RFP is wide ranging and has evolved considerably since its inception, but one aspect relevant to our discussions is of ontology being developed using UML, and implemented in the OWL. This may also allow ontology to be forward and reverse engineered.

Although it is clear a foundation for the description of resources could be established, manually inspecting large volumes of premantic legacy artefacts and then encoding metadata about them is an error-prone, time consuming and expensive task. However, in some situations it is conceivable that it may prove feasible and worthwhile to do just this on a well bounded artefact repository. In such circumstances, any opportunities for automated support are certainly welcome.

While work to automate metadata generation is limited, there are some notable exceptions. For instance, Jenkins et al. [14] developed a system to automatically extract a RDF description from HTML documents. Later Huang et al. [15] recognised the limitations of metadata generation by looking merely at the contents of isolated artefacts. Instead their work shows that thematic metadata can be derived by applying artificial intelligence techniques to groups of related textual artefacts; so affording the possibility of concept searches. Subsequently, Cardinaels et al. [16] have been able to automatically generate metadata for learning objects. They recognise that metadata can be derived from a single artefact and a group of related artefacts, as well as from the context artefacts experience and the uses that are made of them.

In terms of established tool support for metadata management, there are numerous tools, but worthy of note are Protégé [23] (an extensible open-source ontology editor) and SNOBASE [24] (an ontology management system that can allow queries to run across multiple ontologies).

2.4 Peer-to-Peer Technology

In a Peer-to-Peer (P2P) network, peers give up central control and organise themselves dynamically to provide file sharing. P2P success depends on a balanced distribution of data; or at least knowledge about data. Popular implementations of P2P technology have been Napster and Gnutella, and more recently BitTorrent.

With the rise of industrial exploitation of P2P networks, there is a need for standardisation and the IRTF Peer-to-Peer Research Group [19] has been set up with this remit in mind.

In the meantime, JXTA [20] technology does offer a set of open protocols that allow networked devices to communicate and collaborate in a P2P manner. JXTA peers create a virtual network where any peer can interact with other peers. It enables activities such as finding peers and resources on the network even across firewalls, sharing files across the network and creating a group of peers of devices across different networks.

One of the most important issues that SESAME has to deal with is tracing artefacts and notifying interested parties about those artefacts as they change. This is to be achieved using Publish/Subscribe mechanisms [17]. Indeed, we have already seen that publish/subscribe P2P networks have begun to support metadata about resources [1]. Furthermore, Chirita et al. have suggested mechanisms for handling notifications when peers are off-line.

Evolving out of earlier manifestations of JXTA was SPLASH; a P2P repository for learning objects popular in academic circles [21]. SPLASH also provides metadata tagging capabilities for these artefacts. One of the reasons that SPLASH diverged from JXTA was that these learning objects can become large (e.g. a movie file of a one hour lecture), so the SPLASH developers required a more reliable connection mechanism than was needed for sharing small files. Interestingly, a lesson learned by the SPLASH community was that the artefacts' authors find it difficult to enter meaningful and consistent metadata about those artefacts; making human interpretation difficult –

never mind enabling automatic machine reasoning. This chimes with the motivations of Cardinaels et al. [16], mentioned earlier. Another practical realisation was that learning objects need to be used in a variety of situations such as their author's own teaching, a SPLASH network and in a local, proprietary virtual learning environment. As such, it is important that the artefacts should not be customised for, or tightly coupled to, P2P technology; or indeed their own metadata.

3 Motivating Scenario of Use

To help illustrate SESAME's potential impact, we present the following scenario.

Software development company A maintains a large number of diverse premantic artefacts created by different tools at various points in the development lifecycle. For example, legacy COBOL code, J2EE component code, models in UML/XMI and ER diagrams, and documentation in Word, Adobe and Lotus formats.

Company A has two key objectives. Firstly they wish to make these historic artefacts more readily available for future projects in order to encourage re-use and support project planning and design; identifying and targeting potentially reusable components will reduce the time to market of future projects. The second objective is to measure the quality of output from projects. This will help Company A learn from past experience, manage current activity and improve their standards.

A technical solution is proposed that uses metadata such as unique identifiers, code descriptions and quality assessments, to enable company-wide searching and sharing. However, to achieve this, they need to invest a significant amount of time and resources manually creating semantic information and links for each artefact; which can only be achieved by error-prone subjective observation. This is unfeasible for large software development projects where they have thousands of artefacts.

Imagine now they had SESAME, which Company A could use to define ontology and rules to automate the acquisition and management of metadata from their artefacts' content. Once Company A has generated this collection of semantically enhanced components, traceability concepts in the SESAME system would allow for the creation of relationships between artefacts. As a result, Company A can deliver resources to support re-use in future projects. In addition, they will be able to see artefact dependencies and can assess the provenance of the artefacts.

In addition, the P2P implementation of SESAME means that there is no tight coupling between the semantics of an artefact and the artefact itself. Each peer is effectively a designated file storage location which can function without alteration as a host for premantic artefacts. This will prove particularly useful to Company A as there may be collections of artefacts that remain partially premantic and which need to be used by traditional toolsets and projects that are not yet interested in semantic enrichment.

4 Proposed Architecture

In support of scenarios such as this, we propose to develop an architecture and toolset to manage the definition, elicitation and configuration of artefact semantics, specifically in the domain of software engineering. In particular, SESAME will be: a distributed, P2P environment, with cooperating, heterogeneous artefact sources; extensible through a plug-in mechanism; and supportive of multiple platforms. SESAME will provide built in mechanisms for creating both secure corporate semantic networks and generic software engineering communities sharing their experience and public artefacts.

We consider two areas of semantic enhancement; knowledge acquisition (where the semantic data is created to describe a software artefact, potentially extracted from the artefact content itself and its context), and knowledge management (where that semantic data is managed and used). Specifically, SESAME's architecture comprises of four layers. With reference to Fig. 1, the Artefact Layer refers to an environment containing resources that are not semantically enriched; what we call *premantic*. This environment could be comprised of different data repositories (e.g. in our diagram a file system, database, and CVS repository). It could be globally distributed, or limited to a single organisation in a single location.

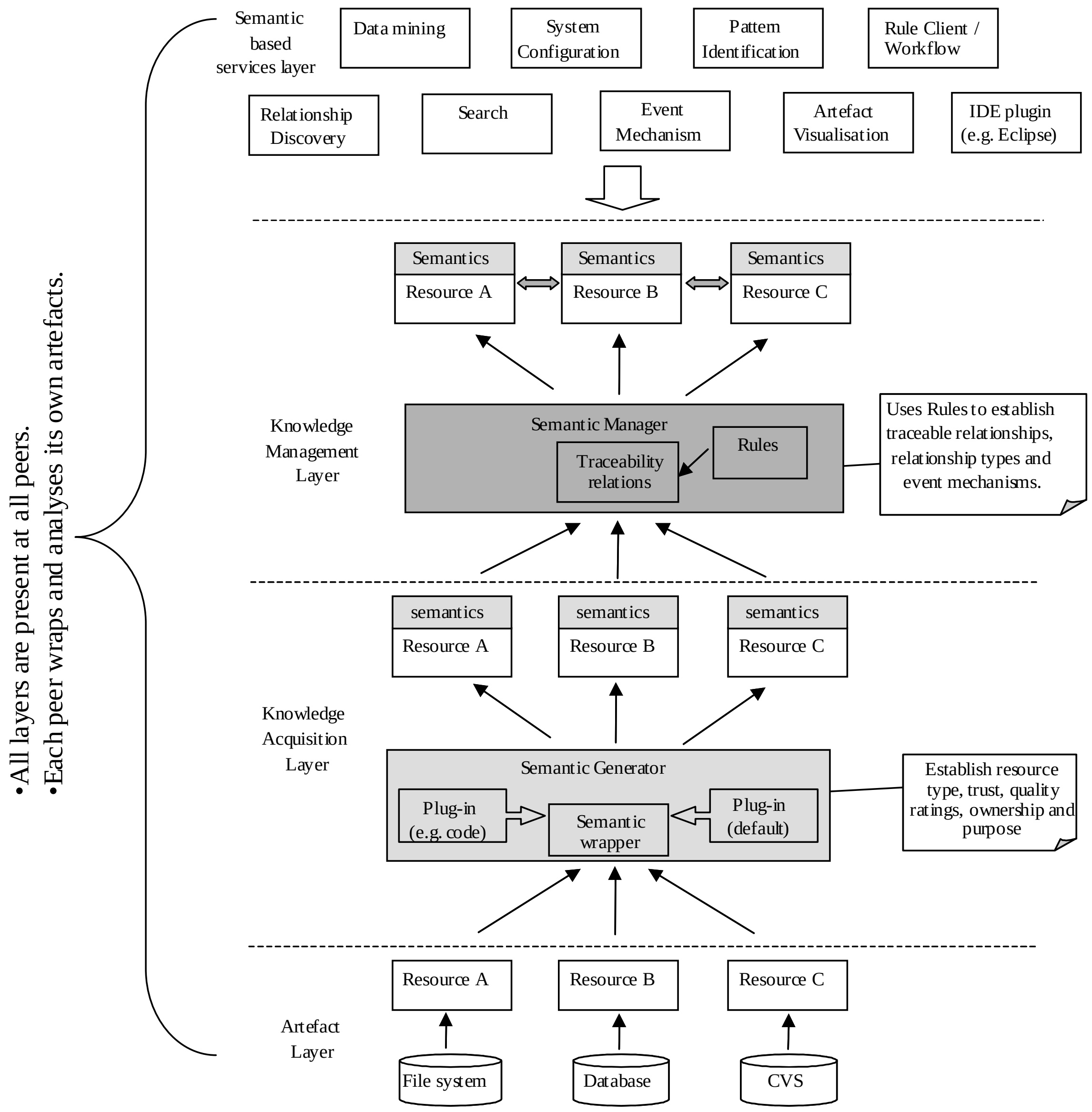

Fig. 1. The Proposed SESAME Architecture

The Semantic Generator in the Knowledge Acquisition Layer semantically enhances artefacts from the Artefact Layer. It is composed of a semantic wrapper and multiple plug-ins. The semantic wrapper has the ability to apply metadata to each artefact. It will have a default plug in to provide core "default" metadata, such as the date the metadata was created. This simple service is extended through interrogator plug-ins responsible for automatically examining an artefact type to extract additional "artefact specific" metadata. Where automation is not possible, user interface plug-ins will allow users to manually augment core metadata. However, such manual enhancement can be facilitated optionally by the application of ontology which can suggest the necessary vocabulary.

Relationships between semantically enhanced artefacts are defined in the Knowledge Management Layer. The semantic manager uses rules to create and define traceability relations between artefacts and potentially derive extra metadata from such context.

Once inter-artefact relationships have been established, a number of semantic based services add value to the overall architecture. There are clients to access and further define the traceability relations, and the rules governing them. These rules can be used to define workflow within a project, and therefore to specify the software development methodology to be followed. Artefacts can be searched, and patterns identified within and between code artefacts. In addition, an event mechanism can establish notifications between related artefacts and interested users. Finally, the entire environment can be visualised to aid understanding.

SESAME’s usefulness and acceptance will be enhanced if it can be integrated into existing IDE technology, so plug-ins for at least one major IDE (initially Eclipse) will be created. This will allow the Eclipse user to access services from all three tiers of the SESAME system through the Eclipse IDE.

Furthermore, pragmatics dictate that a thoroughbred P2P approach will not offer the performance or flexibility that could be obtained if other paradigms were embraced. Therefore, SESAME will incorporate *super peers* and novel resource discovery mechanisms to improve performance. In addition, workflow orchestration would not discount the possibility of incorporating salient Web services as they become available.

5 Conclusions

This article has described the motivations for, the background to and the architecture of SESAME. The open development environment that this framework provides for software engineers addresses a number of key socio-economic problems. SESAME’s resource discovery facilities, and the resulting artefact metadata, will support component technologies and facilitate content reuse, driving down costs and reducing time to market. Metadata can also contain an assessment of quality, which should assist vendors to monitor and maintain higher standards and consumers to make better-informed choices.

Furthermore, the creation of metadata will make artefacts more comprehensible, transparent and searchable to users with different understandings and business needs. In addition, the semantic based services of the SESAME architecture will help users to understand the system that produced the artefacts. For example, artefact visualisation will provide an overview of all artefacts and their relationships within a system, which can be mined for specific business perspectives.

Returning to Wasserman and his five dimensions of tool integration [2], we could argue that platform and presentation integration as generally solved in contemporary environments. This leaves data, control and process integration. Fundamentally, P2P technology provides data integration, but SESAME’s rich architecture offers opportunities to accommodate the remaining two dimensions. Combined with our discussions of low coupling between artefacts, tools and environment, this all bodes well for our approach.

In developing a modern knowledge-based society, it will become increasingly important to efficiently incorporate legacy data within semantic-enabled systems and services. Providing an indication of artefact quality through metadata will encourage trust and confidence in those wishing to access or reuse those artefacts. In addition, the detection of artefact signatures (e.g. the identification of authorship from content, structure, use, etc) is an area SESAME will investigate. Should this be achieved, issues of trust in both producer-to-customer and customer-to-producer directions can be addressed. Moreover, the intellectual property rights of the developer could potentially be upheld should an artefact be plagiarised or used out of licence, and consumers will have greater assurances about the provenance of the artefacts they use.

6 Acknowledgements

Grateful thanks are given to the partners in the SESAME project.

References

1. P.-A.Chirita, S.Idreos, M.Koubarakis, and W.Nejdl (2004) Publish/Subscribe for RDF-based P2P Networks, The Semantic Web: Research and Applications: First European Semantic Web Symposium (ESWS), Crete, Greece (Lecture Notes in Computer Science), Vol.3053, pp.182 – 197.

2. A I Wasserman. “Tool integration in software engineering environments”. In F Long, editor, The International Workshop on Environments (Software Engineering Environments), volume 647 of Lecture Notes in Computer Science, pages 137-149. Springer-Verlag, Berlin, September 1989. Chinon, France.
3. SourceForge, http://sourceforge.net/, last accessed 5/5/05.
4. Eclipse, http://eclipse.org/, last accessed 5/5/05.
5. F.Cattaneo, E.Di Nitto, A.Fuggetta, L.Lavazza, G.Valetto (2000) “Managing software artifacts on the Web with Labyrinth”, 22nd International Conference on Software Engineering, pp.746-749.
6. Olsen, T. and Grundy, J.C. (2002) “Supporting traceability and inconsistency management between software artefacts”, IASTED International Conference on Software Engineering and Applications, Boston.
7. SA Sherba and KM Anderson (2003) "A Framework for Managing Traceability Relationships between Requirements and Architectures". Second International Workshop From Software Requirements to Architectures at ICSE, USA.
8. R.A.Falbo, G.Guizzardi, Natali ACC, Bertollo G, Ruy FF, Mian PG (2002) “Towards Semantic Software Engineering Environments”, International Conference on Software Engineering and Knowledge Engineering, Italy.
9. Software Engineering Task Force, http://www.w3c.org/2001/sw/BestPractices/SE/, last accessed 5/5/05.
10. H. Knublauch (2004) “Ontology Driven Software Development in the Context of the Semantic Web: An Example, Scenario with Protégé/OWL”, 1st International Workshop on the Model-Driven Semantic Web (MDSW2004)
11. D.Oberle, A.Eberhart, S.Staab, R.Volz (2004) “Developing and Managing Software Components in an Ontology-based Application Server”, In Hans-Arno Jacobsen (ed.), Middleware 2004, ACM/IFIP/USENIX 5th International Middleware Conference, Toronto, Ontario, Canada, volume 3231 of LNCS, pp. 459-478. Springer.
12. UML, http://www.uml.org/, last accessed 5/5/05.
13. OWL, http://www.w3.org/TR/owl-features/, last accessed 5/5/05.
14. C.Jenkins, M.Jackson, P.Burden, J.Wallis (1999) “Automatic RDF metadata generation for resource discovery”, Journal of Computer Networks, 31(11-16), pp.1305-1320
15. C.C.Huang, S.L.Chuang, L.F.Chien (2004) “Using a web-based categorization approach to generate thematic metadata from texts”, ACM Transactions on Asian Language Information Processing (TALIP), 3(3), pp.190-212
16. K.Cardinaels, M.Meire, E.Duval (2005) “Automating Metadata Generation: the Simple Indexing Interface”, preprint of an Article accepted for publication in ACM International World Wide Web Conference, WWW 2005, Chiba, Japan.
17. S. Baehni, P. Eugster and R. Guerraoui (2002) “OS Support for P2P Programming: a Case for TPS”, Int. Conference on Distributed Computing Systems, (ICDCS), Austria.
18. M.N.Wicks and R.G.Dewar (2005) Ad Hoc Tool Integration; a Case Study, International Conference on Software Development (SWDC-REK), Reykjavik, Iceland, (May-June) accepted.
19. Peer-to-Peer Research Group, http://www.irtf.org/charters/p2prg.html, last accessed 5/5/05.
20. JXTA (2004) http://www.jxta.org/, last accessed 5/5/05.

21. SPLASH (2003) http://www.edusplash.net/, last accessed 5/5/05.
22. openRDF; home of Sesame (2005) http://www.openrdf.org/, last accessed 5/5/05.
23. Protégé (2005) http://protege.stanford.edu/, last accessed 5/5/05.
24. SNOBASE (2004) http://www.alphaworks.ibm.com/tech/snobase, last accessed 5/5/05.
25. ODM document (2003) http://www.omg.org/cgi-bin/apps/doc?ad/2003-03-40, , last accessed 5/5/05.