

# Intelligent Selection of Components

Valerie Maxville

<sup>1</sup> Murdoch University, Perth WA, Australia,  
valerie@eng.murdoch.edu.au,  
WWW home page: <http://eng.murdoch.edu.au/~valerie>

2

**Abstract.** With an increase in the range of software components available, selection of the most appropriate component can be difficult. This paper presents the initial work of a research project that aims to develop strategies and tools to assist in the process of testing and evaluating candidate components, using artificial intelligence techniques. Components will be verified against their supplied specification and test sets generated for execution and evaluation of results. An XML schema has been developed to allow component specifications to be recorded in a standardised manner. Although the schema is designed for subsequent use in this project, it is applicable to component developers, brokers and registries in provision of information to potential users.

## 1 Introduction

Software reuse can help to reduce instances of overtime, over budget and unreliable software. Object-oriented techniques, new programming languages and new software frameworks have enabled a growing industry to develop reusable software components. An indication of this trend is that one component broker now has over 8000 components registered [1]. Component Based Software Engineering (CBSE) looks at the issues and approaches particular to systems built from reusable components. To gain the full potential of component based software development, the access and integration of the components has to be made as smooth and problem free as possible. Unfortunately, there are difficulties encountered when developing software from components that include:

- Availability of components (sourcing)
- Extra time needed for development
- *Selecting the “best” component(s)*
- *Component trust*
- Testing a black-box component
- Integration problems (architectural mismatch)
- Vendor reliability and risk management
- Configuration management and interdependencies

Component brokerages and software repositories have been developed and continue to improve access and awareness of available components [2]. Systems for

ranking candidate components exist, and allow for various attributes and their values to be given weightings to indicate relevance and importance [3]. These are static selection processes, not involving execution of the component being assessed. Some approaches to improving component trust include programming by contract (PBC) [4] and 3rd party certification of components [5]. Testing of software components draws on the existing work in black-box testing [6]. Component specific work includes techniques for drawing information out of undocumented components [7] and testing for robustness [8,9]. Integration issues come about from mismatches between a component and its surroundings (interfacing components and frameworks) [10]. Remedies for these problems include wrappers and mediators to massage a component into submission. The risk of dependence on 3rd party vendors can deter use of components, with some support provided by open source developers. Configuration management is critical in CBSE as there is a need to track components and their interrelationships through tools such as dependency maps [11].

### 1.1 This Project

The research in this project aims to address the issues of *trust* and *selection* of software components, with indirect benefits in *testing* and *sourcing*. Developers need to trust and see clear benefits before they will use 3rd party software. To gain maximum benefit from CBSE, the developer needs to minimise the amount of effort in the selection and associated testing for each component. It takes time to evaluate software, with the first drain on this resource when searching for candidate components. Any approach to a broad search (i.e. no specific name to search for on the Internet) will be time consuming and likely to return many results that are not relevant, while brokers, such as Component Source [1] and Component Planet [2], are restricted to those components that have been registered to the catalogue. An alternative is to have standardised metadata about software components that can then be indexed and matched through existing search engines. This enhances the relevance of the results by only returning information about matching software components (and not assorted document types).

At this point the developer can be confident that they have a representative set of available components. The next time concern is the evaluation and ranking of the harvested components. Developers need to decide on criteria to rate the components to allow for comparison. This may include performance, security, ease of integration, or a combination of these and other weighted criteria. *This project* aims to assist developers in the evaluation phase through the provision of a standardised specification and through strategies and tools for testing and ranking components.

The development of the component data model and the swvML XML schema provides a foundation for the development of tools and strategies for the intelligent selection of software components. XML simplifies the transformation of an XML instance document to an internal tree through DOM and SAX implementations. This tree can then be processed using existing computing science

techniques. New W3C developments for working with XML documents allow nodes to be accessed via XPath and XQuery without traversing the entire tree. Much of the information in the first level of the data model is for searching and short-listing purposes. The more detailed information under the technical branch of the tree will be extracted and used to develop test cases using artificial intelligence techniques. These tests will be executed on the candidate component and the results evaluated and ranked, again using artificial intelligence. It is expected this work will assist in making component based systems a more efficient and reliable method of software development.

## 2 Describing Components

Component developers often choose to make use of the Internet to market and promote their products. A common mechanism for providing information about electronic resources is the use of metadata data describing the resource. It is increasingly common to see this implemented using a standardised metadata scheme and XML documents. The metadata file is a logical place to store information to help Internet searching for components as well as the more technical information about a component.

The deliverables for the first phase of this project have been to develop an XML schema and associated templates to hold metadata for software components. Requirements for the schema were that it should contain data useful for component selection and would attempt to adhere to current and foreseen trends and standards in the documentation of electronic resources. The attributes to include for each component had to allow for standardised searching along with more a technical specification to provide for later testing and evaluation of components.

### 2.1 XML

There are various options for holding component specifications, but if the aim is for interoperability, then an interchange standard should be followed. Extensible Mark-up Language (XML) has rapidly become the standard for the interchange of information. It improves on previously common methods of transfer: text, binary or csv files through its machine independence and integration with Internet technologies. The power of XML comes from the use of schemas or Document Type Definitions (DTD) to create a structure for the data model. The XML instance document is a text file that can be read and edited easily on any platform. Nesting of these tags creates a hierarchy. XML Schemas improve on DTDs by providing namespaces to reuse existing schemas, have more datatypes and tighter validation of instance document against structure.

By defining a structure, the developer can create a vocabulary for a domain which can then be processed by 3rd party or in-house software to meet local requirements. This processing may include transformation for display or data conversion purposes using XSL and/or style sheets. Applications can successfully

work with files of unknown structure, ignoring all but the tags they expect. The extensibility of XML allows novel structures, as well as new schemas to be built out of existing schemas.

## 2.2 Schema Development

One area where standards are being developed is in the description of electronic resources through metadata. The Dublin Core Metadata Initiative has developed a data model to describe diverse electronic resources in a standardised manner [13]. Dublin Core is widely accepted and often other metadata standards build upon the dataset through application profiles to suit more specific needs. An example is the Australian Government Locator Service (AGLS) which has added four fields to the 15 in Dublin Core in line with their application for metadata [14].

Accessible via the Internet, our components can be viewed as electronic resources, making Dublin Core an appropriate starting place for a metadata description. Developing a schema can be approached in number of ways: use existing, modify or combine existing or create new schemas [15]. Dublin Core is intentionally generic: this project requires more specific information about the component to provide information for decision-making. Although schemas for components exist, these are mainly aimed at automating the registration to component broker websites and for searching within the site [2]. The future phases of this project require fields for more specific technical information than is required by brokers. There is value in the work already done in describing components and electronic resources, so to create a completely new schema would lose that information and the chance of wider compliance. The preferred approach is to modify and combine existing schemes to create a schema as a proposed standard. Revisions of the schema will continue this approach, tracking and adopting new standards as appropriate.

## 2.3 Data Model

Existing schema and component characterisation work was surveyed to compile a list of relevant fields for inclusion in the data model [22]. In the interests of accessibility at the electronic resource level, any fields matching Dublin Core were aligned and renamed accordingly. This ensures Dublin Core compliance for component metadata. Other fields were grouped and arranged in a hierarchy based on a class diagram of the data model (figure 1). An XML Schema can enforce rules of cardinality, and these are indicated in the diagram. In Dublin Core, all elements are repeatable and optional.

An interesting variation in the resulting schema is the depth of the component specific hierarchy as compared to the Dublin Core sections. In order to keep the data model simple, and allow for easy processing, Dublin Core fields are restricted to the first level of the tree, as have newfields of similar purpose. For the more component specific fields, a logical hierarchy has been formed to indicate dependencies and to allow parsers to easily pass over information. Where

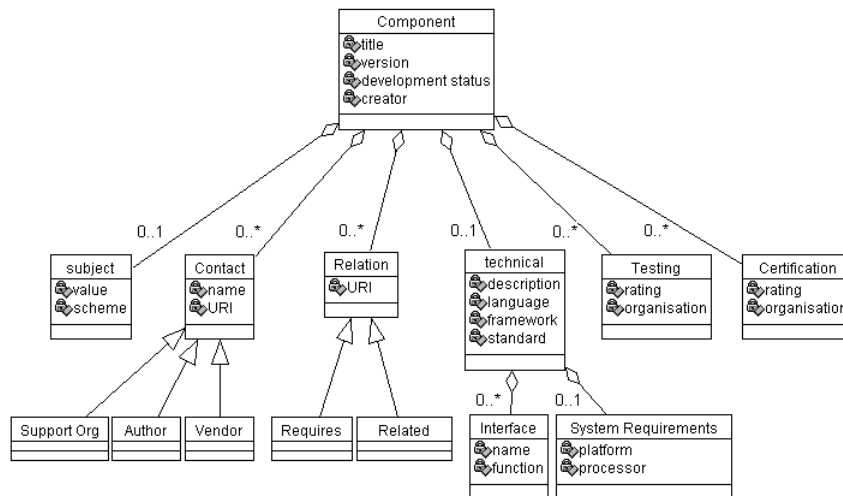


Fig. 1. swvML Data Model

possible, fields are set up to adhere to the Dublin Core “dumb down” principle [16], that is, the information in fields should be sensible, even if the hierarchy is flattened. An XSL transformation is provided to convert a schema document to standard Dublin Core [12].

Along with the specification of fields and their position in the hierarchy, it is important to describe valid data for a field. The default datatype for XML is string which can be parsed simply. Other encoding schemes refer to the data format (e.g. ISO8601 for dates) or enumerate allowed values (e.g. Library of Congress Subject Headings). Faceted classification provides a controlled vocabulary to describe resources with respect to certain attributes, or facets. A domain specific encoding scheme avoids ambiguity due to differences in terminology and provides a context for a descriptor through its association with a facet. Prieto-Diaz and others have used faceted classification to categorise software for reuse [17]. Faceted classification has been used to encode aspects of the swvML schema. Through comparison of the faceted classification of pairs of components it is possible to give an indication of similarity [18] that will be important in the later phases of this project.

The swvML data model has been implemented in XML. An example of an instance document for a particular component is shown in figure 2. The swvML schema site provides design documentation, style-sheets and a user guide including a template for instance documents to encourage correct usage of the schema. The style-sheets provide a more readable version of the specification in HTML and provide a dumb-down conversion to standard Dublin Core. The documents and templates were developed in XMLWriter [19] and XMLSpy [20], with validation through W3C XSV [21]. W3C standards and trends will be observed

```

<dc:title>softML Application Profile</dc:title>
<soft:version>1.0</soft:version>
<dc:creator>Valerie Maxville</dc:creator>
<dc:creator>Valerie Ruth Maxville</dc:creator>
<subject subjectScheme="SFC">
  XML, metadata
</subject>
<dc:description>
  The softML schema and application profile aim for
  standardisation of XML documents describing software
  components.
</dc:description>
<soft:detail>
  The softML schema and application profile aims to help
  Standardise XML documents describing software components.
  It uses the Dublin Core standard and has an application
  profile bringing in software component
  specific fields.
</soft:detail>
<dc:publisher>Murdoch University</dc:publisher>
<soft:support>valerie@eng.murdoch.edu.au</soft:support>

```

**Fig. 2.** swvML instance document

to maintain schema currency and to make the best use of available tools (e.g. XQuery) [23].

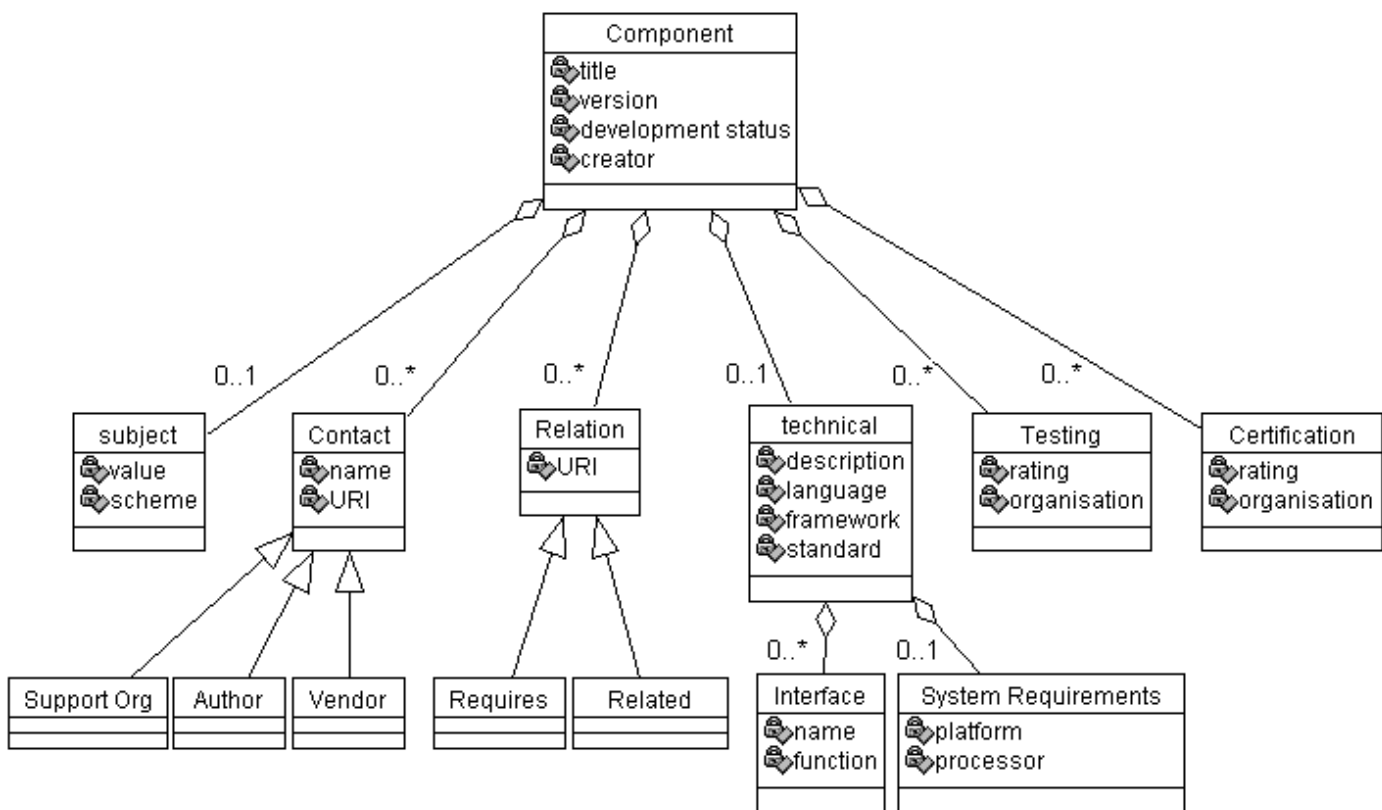
### 3 Conclusion and Future Work

In this phase of the project, the XML schema and associated templates were developed and tested. The schema adheres to standards and conventions, including the widely accepted Dublin Core and is expected to be refined following publication and application in the field. The schema will be used to provide standardised information for the later testing and evaluation of candidate components. The intention is for this schema to be used for executable software components, but it is also suited to describing related resources, including XML schemas, style-sheets and translation documents. This allows for consistent descriptive information for software and other necessary resources, and records the relationships between them. Future work in this project is to continue with the development of strategies and intelligent tools for the evaluation of components.

### References

- [1] Component Source: website, URL: <http://www.componentsource.com/>, January 2002
- [2] Component Planet: website, URL: <http://www.componentplanet.com/>, January 2002
- [3] Seacord, R.C., Mundie D., Boonsiri S.: K-BACEE: Knowledge-Based Automated Component Ensemble Evaluation. in Proceedings of the 2001 Workshop on Component-Based Software Engineering. 2001. Warsaw, Poland: IEEE Computer Society.
- [4] Meyer B.: Design by Contract: The Lessons of Ariane, in Computer (IEEE), vol. 30, no. 1, January 1997, pages 129-130

- [5] Voas J.: Developing a Usage-Based Software Certification Process, *IEEE Computer*, vol. 33, pp. 32-37, 2000.
- [6] Myers G. J.: *The Art of Software Testing*. New York: John Wiley & Sons, 1979.
- [7] Korel B.: Black-box understanding of COTS components, presented at Proceedings of the Seventh International Workshop on Program Comprehension, Pittsburgh, Pennsylvania, 1999.
- [8] Yoon H., Choi B.: Inter-class technique between black-box-class and white-box-class for component customization failures, presented at Proceedings of the Sixth Asia Pacific Software Engineering Conference, Takamatsu, Japan, 1999.
- [9] Ghosh A., Schmid M.: An approach to testing COTS software for robustness to operating system exceptions and errors, presented at Proceedings of the 10th International Symposium on Software Reliability Engineering, Boca Raton, Florida, 1999.
- [10] Shaw M., Garlan D.: *Software Architecture: Perspectives on an Emerging Discipline*. Upper Saddle River, NJ: Prentice Hall, 1996
- [11] Larsson M., Crnkovic I.: Configuration Management for Component-based Systems, URL: <http://www1.ics.uci.edu/~andre/scm10/papers/larsson.pdf>
- [12] Open Archives Initiative: website, URL: <http://www.openarchives.org/>, January 2002
- [13] Dublin Core Metadata Initiative: website, URL: <http://dublincore.org/>, January 2002
- [14] Australian Government Locator Service: website, URL: <http://www.naa.gov.au/recordkeeping/gov>
- [15] O'Neill, E.T., Childress E., Dean R., Kammerer K., Vizine-Goetz D., Chan L. M., El-Hoshy L.: FAST: Faceted Application of Subject Terminology. in IFLA Satellite Meeting. 2001. Dublin, Ohio.
- [16] Baker T.: A Grammar of Dublin Core, *D-Lib Magazine*, 6(10), 2000, URL: <http://www.dlib.org/dlib/october00/baker/10baker.html>
- [17] Preito-Diaz, R.: Implementing Faceted Classification for Software Reuse. *Communications of the ACM*, 1991. **34**(5): p. 89-97.
- [18] Girardi M. R., Ibrahim B.: A Similarity Measure for Retrieving Software Artifacts. *Proc. Int. IEEE Conf. on Software Engineering and Knowledge Engineering (SEKE94)*, Riga, 1994.
- [19] XMLWriter: website, URL: <http://www.xmlwriter.com/>, January 2002
- [20] XMLSpy: website, URL: <http://www.xmlspy.com/>, January 2002
- [21] XSV: Validator for XML Schema, URL: <http://www.w3.org/2001/03/webdata/xsv/>, January 2002
- [22] Maxville V.: swvML Schema Design Documentation, URL: <http://eng.murdoch.edu.au/~valerie/>, January 2002
- [23] World Wide Web Consortium: website, URL: <http://www.w3c.org/>, January 2002



```
<dc:title>softML Application Profile</dc:title>
<swv:version>1.0</swv:version>
<dc:creator>Valerie Maxville</dc:creator>
<dc:creator>Valerie Ruth Maxville</dc:creator>
<subject subjectScheme="SFC">
  metadata, components, XML
</subject>
<dc:description>
  The swvML schema and application profile aim for
  standardisation of XML documents describing software
  components.
</dc:description>
<swv:detail>
  The swvML schema and application profile aims to help
  Standardise XML documents describing software components.
  It uses the Dublin Core standard and has an application
  profile bringing in software component
  specific fields.
</swv:detail>
<dc:publisher>Murdoch University</dc:publisher>
<swv:support>valerie@eng.murdoch.edu.au</swv:support>
```