

# **XmlTRAM+: Using XML Technology To Manage Software Requirements And Architectures**

Jie Wu, Faculty of Information and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing210016, China/School of Network Computing, Monash University Jie.Wu@infotech.monash.edu.au

Jun Han, School of Network Computing, Monash University, McMahons Road, Frankston/Melbourne, Vic 3199, Australia Jun.Han@infotech.monash.edu.au

## **Abstract**

Information systems are increasingly being used in all aspects of an organization's business activities. These systems will inevitably evolve over time. The system development knowledge is a key to the understanding and evolution of these systems. As such, the system development knowledge is part of the corporate knowledge that needs to be properly managed. In particular, the system requirements and architecture design are the most important system development knowledge. In this paper, we introduce an XML-based tool for managing system requirements and architectures. The use of the XML technology allows the system requirements and architectures to be easily shared across the organization. The issues discussed include a conceptual model, a logical model and a physical model for the management of system requirements and architectures. In fact, this three-step development process points to a typical development methodology for the development of XML-based systems. Keywords: Requirement, Architecture, XML schema

## **1. Introduction**

Management of system requirements, system architectures and the traceability between them provides critical support for system development and evolution. The requirements for a system are the basis of planning, developing, evolving and using the system. The system architecture provides the blueprint or vision for the system's design. The traceability between the system requirements and the system architecture is the key to test whether the requirements are met by the architecture design. In the light of changes to systems, the management of system requirements, system architectures and their traceability has even a greater role to play. It facilitates analysis of how a new or changed requirement will affect the system design and how an architectural design decision will impact on the system's functionality and quality. In current practice, system requirements are often kept in some monolithic word-processing files. They are difficult to analyse and maintain. The specification of system architectures is usually ad hoc, again hard to analyse and maintain, and difficult to be kept up-

to-date. Even with certain tool support, the system requirements and the system architectures are kept separately, and support for their traceability is very limited. Furthermore, most support tools available are either very generic so that only low-level assistance is possible, or too specific by dictating the use of a particular notation. To address the existing problems, we have developed a dedicated tool TRAM for requirements and architecture management, which has been used at UK National Air Traffic Services (NATS) [Han, 2001]. The tool involves an information model, a set of document templates detailing the model, and templates' implementation in the generic document management tool DOORS. Experience has shown that the tool has delivered great improvement in requirements and architecture management [Han, 2000]. The information model and templates were also easily implemented in another generic document management tool RTM due to the preference of another user organization. However, this has caused an information exchange problem because both DOORS and RTM have their own proprietary document representation schemes. The requirements and architecture information held in one tool can not be used in the other tool. This is a major issue as it hinders information or knowledge exchange between relevant organizations can not be easily exchanged, e.g., between client and developer organizations. In this paper, we report our effort in improving the TRAM tool, xmlTRAM+. This new tool has the following two major improvements over the TRAM: vAn enriched model for requirements and architecture management. In TRAM [Han, 2001], the requirements are managed in a relatively flat structure, not recognizing subsystems or component systems, while the architecture has a component-based management structure. In xmlTRAM+, the requirements are also managed relative to component systems so that there is a close correspondence between the requirements management structure and the architecture management structure. vImplementation of the requirements and architecture model and related document templates using the XML technology. This resolves the information interchange issue between different implementation, as the XML technology affords a standard way of representing information. The paper is organized as follows. In section 2, we introduce our approach to XML-based management of system requirements and architectures. Section 3 discusses the conceptual information model of requirements and architecture management. Section 4 presents the logical design of the requirements and architecture management model. In section 5, we describe the actual (physical) implementation of the logical design in the XML technology. Section 6 discusses some key issues arising from the prototype implementation of the tool and a real-world case study.

## **2. XML-based requirements and architecture management**

The system requirements and architectures involve a large amount of information, which presents a great challenge for management, analysis, sharing and successful use. As discussed above, an arising need is to facilitate interchange of such engineering information, to develop proper methods for better information management across the various engineering activities, and to improve analysis, sharing and use of such information resources. UML is a well accepted modeling notation in the areas of requirements analysis and architecture design, where objects are presented in various views. But, how to

manage such data and knowledge is outside of the UML domain and remains a challenge. Furthermore, it is not always the case that system requirements and architectures are modeled using UML. Other formal, semi-formal and informal notations are often used as well. As such, there is an urgent need for powerful tools to represent and manage the requirements and architecture information in general. The tool should support the access to the requirements and architecture information by developers from different platforms. The XML technology is a tool that is capable of facilitating the above tasks. It has such a versatile format that it can be applied in any industry. It has become the de facto standard for managing and exchanging corporate information. XML is object-oriented in the sense of being suitable for describing objects of the real world or any abstract problem domain by modeling their properties as they are. One particular feature worth noting is that XML/Web-based knowledge management naturally accommodates distribution. That is, the enterprise architecture information may be managed in a distributed manner at different sites across an organization. Based on the above features, we have chosen the XML technology for the representation and management of the system requirements and architectures. That is, the system requirements and architecture model mentioned above is represented using XML for storage, management and analysis. We notes that our information model for system requirements and architectures is not restricted UML, but can accommodate various development methodologies or languages. Our approach does not get into the details of specific analysis and design notations. For any specific notation like UML, however, its representation in XML, such as in XMI, can be incorporated into our general representation framework. In using the XML technology to represent the system requirements and architectures, we take a three-step design approach. In particular, we first introduce a conceptual information model for system requirements and architecture – the conceptual model. Then, the conceptual model is structured to reflect the logical partition of the information involved – the logical model. Finally, the logical model is mapped to XML representations using XML schema – the physical level. This overall approach is illustrated in Figure 1 as referred in [Routledge, 2002]. The following sections will introduce these three levels in detail.

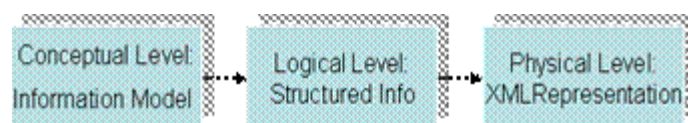


Figure 1. Three-step XML system design

### 3. The conceptual model (conceptual level)

As discussed in the previous section, the first step in our proposed approach is to develop a conceptual model to capture the key information of requirements, architectures and their relationships. Figure 2 presents the conceptual information model as an Entity Relationship (ER) Diagram referred to [Tan, 2001]. The concepts involves are stakeholders, goals, values, authorities, assumptions, risks, acceptance- criteria, components, interfaces, services, quality of service, and use cases. Detailed explanation of these concepts and their relationships is beyond the scope of this paper as we focus on the XML-

based representation of this conceptual model. Interested readers are referred to [Han, 2001] for details.

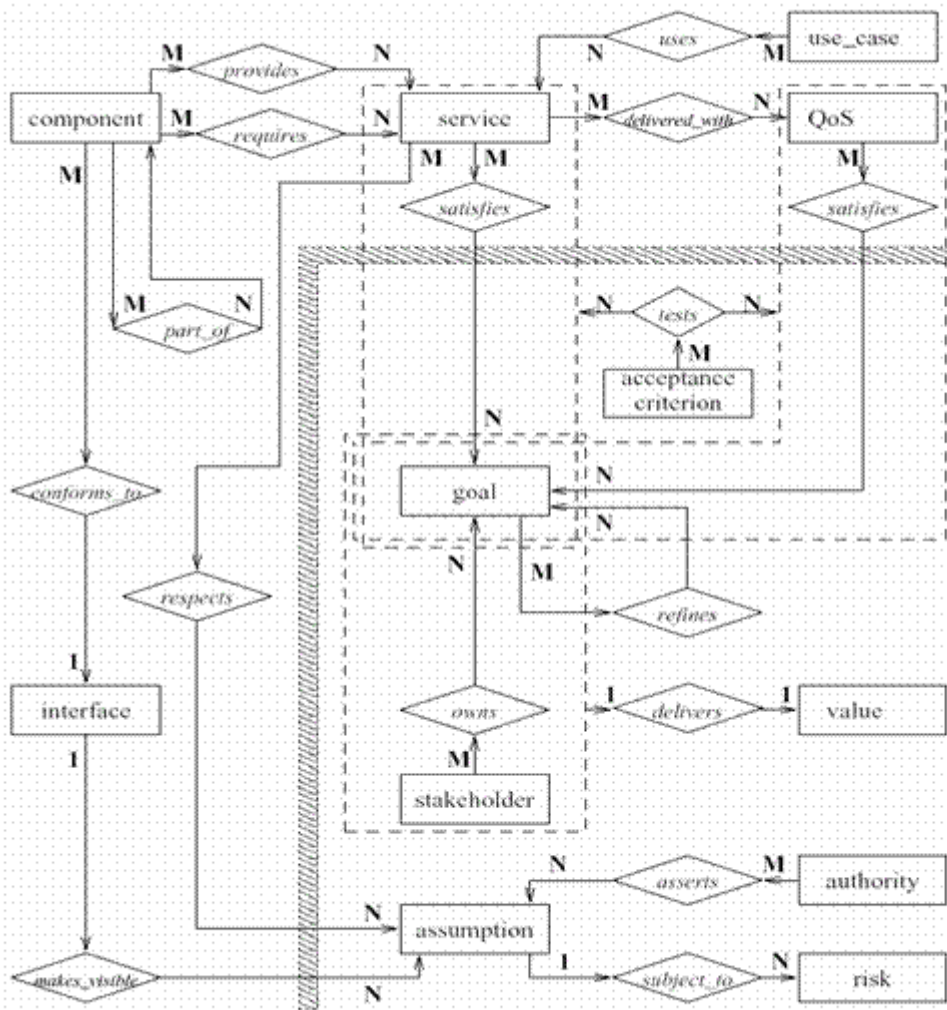


Figure 2. A conceptual model of requirements and architectures

#### 4. The logical structure (logical level)

To be properly managed, the requirements and architecture information needs to be organized and structured according to its logical relationship and grouping. The logical structure is to transfer the conceptual model into a form of XML schema in an abstract and graphical way, that is, a model for better representation using XML. In our approach, both the requirements and architecture aspects from the concept level are structured in a component-based manner. Generating the framework of logical level (which is shown in figure 3) may start from the following steps: vGrouping entities in separate spaces. Although figure 2 gives a practical model for the requirement and architecture management, relatively separate spaces are required for the concepts of each aspect so that elements can be grouped for better presentation. For example, from the conceptual model, concepts like stakeholders, authorities, assumptions, goals, risks and value can be put into "Requirement" Definition, while interface, services, QoS (quality of services),

interactions, constrains and components are grouped into "Architecture" Definition.

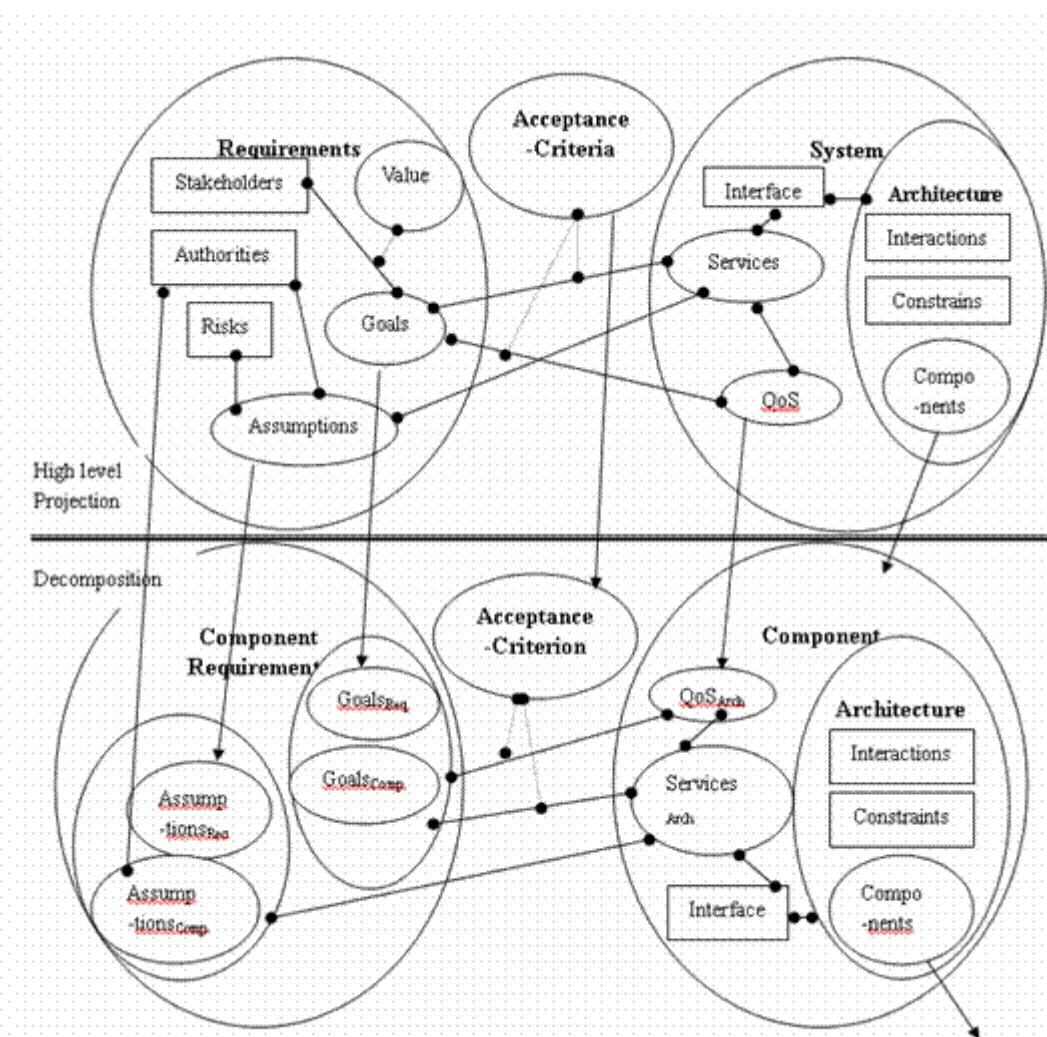


Figure 3. Logical Requirement and Architecture model based on conceptual level

Inheriting component requirements space from global requirements space and introducing templates for system/component design. From the component-based point of view, Requirements of a system has several components where some structures of component elements can be inherited from the root element. Same as the architecture design since system can be seen as a super component. So we put decomposition to clarify the levels in each space in the logical diagram. Figure 3 presents the overall logical design. It also highlights some of the rationale behind the design decisions taken. As a result in our component-based model for management on requirement and architecture, the related concepts are easily identifiable as they are organized into separated information spaces. This logical structure graph clearly identifies, separates and relates the hierarchical composition of the system, from the entire system level down to the atomic component level. Further more, template-based structure for components is provided which is powerful for better representing with XML /data objects. It's aimed as a bridge between the requirement and architecture engineering and its XML management. This bridge facilitates the

task of so called logical level.

## 5. The XML representation (physical level)

Once the logic structure/design has been determined, the physical XML schema can be created accordingly. A specific decision is to have two separate schemas: one for system requirements and one for system architecture. Elements in the logical diagram can be broken down into a generic hierarchical tree-like structure with elements being composed of several other elements. For the design of Requirements schema, for instance, Stakeholders and Authorities are children of Requirements-Definition which is the root element. Values belongs to the Goals and the same for Risks with Assumptions. All of them are defined as the Req-Space, which is another child of the root. Acceptance-Criteria is the concept linking between the Requirements-Definition and the Architecture-Definition. It is now put in the former space since the xlink technology can't be shown in any browser at the moment. These elements are then grouped and encapsulated within specific code representations describing how the elements are organized. Figure 4 gives the XML schema profile for requirements-related concepts.

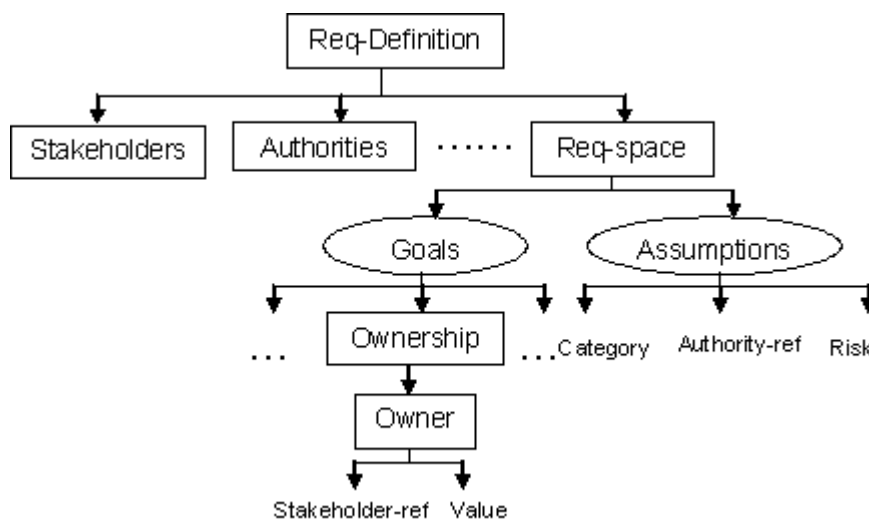


Figure 4. Physical View of Requirement Definition

For the architecture model, the design is similar, and Figure 5 shows the profile for architecture-related concepts.

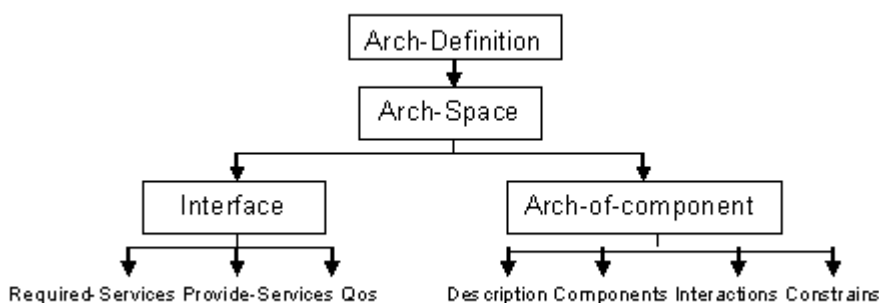


Figure 5. Physical View of Architecture Definition

Figure 6 shows the XML schema for the Requirement model generated from Figure 4.

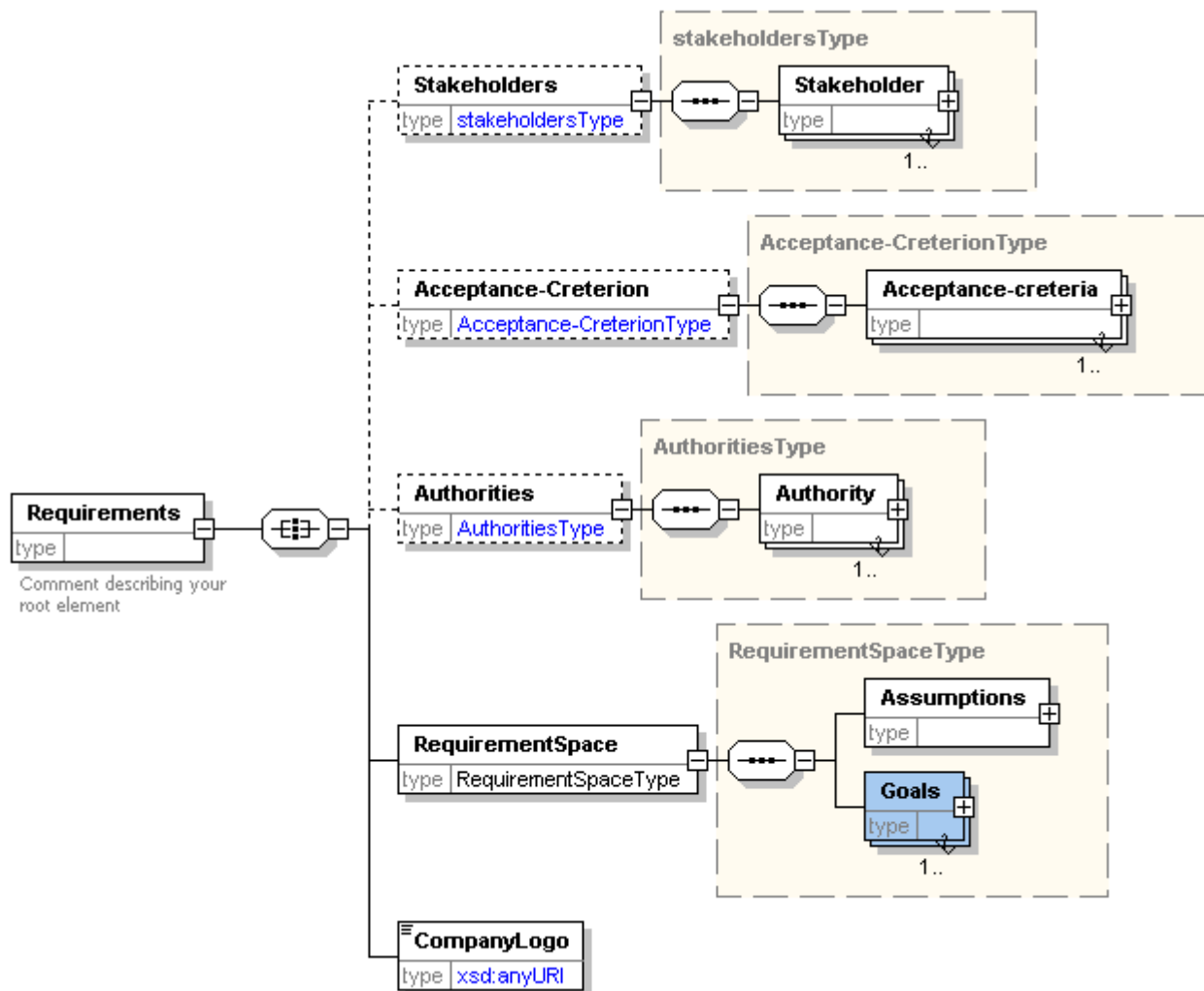


Figure 6. The Requirement XML schema

## 6. Prototype and case study

Prototype. We have implemented a prototype tool based around an XML editor, XML Spy, to replace a word processor or a browser. This is advantageous because it produces XML documents that can be parsed, read into database, transformed through a style sheet, manipulated through the myriad of XML tools available to us, and imported into a familiar word processor-like environment. Compared with other products, XML Spy is significantly more powerful. It is the first true Integrated Development Environment (IDE) for XML development and contains many useful functions that can not only simplify typical XML editing tasks, but also provides graphical processor interface instead of text-style mark-up. This is ideal for applications targeted at non-technicians. It is worth mentioning that the special linking mechanisms are required to present the relationships between the elements. As the Xlink and Xpointer technologies are not supported by any browser yet, the IDREF

attribute nodes are used for intra-document references for the time being. In Figure 4 for instance, the element "stakeholder" is referenced by the element "owner". So the attribute named "REF" is added on the element "Stakeholder-REF". In this way, the referenced element will occur when clicking the "parent element" on the browser. In the prototype, we have also defined the XSL style sheets for generating the presentable HTML documents based on the XML documents. Case Study. The system requirements and architecture model has been instantiated in the prototype tool to the "Short Term Conflict Alert" project of NATS. The requirements and architecture specifications were formulated according to the concepts of the model. The XML documents, which are partly shown in Figure 7, are able to be converted in and out of database, and be manipulated using scripting languages. Figure 8 is the HTML representation for Figure 7. It gives a new methodology for representation, analysis and management of requirements and architectures.

The screenshot shows the XML Spy interface with the following structure:

- XML
  - Requirements
    - RequirementSpace
      - Acceptance-Criterion
        - Stakeholders
          - Stakeholder (12)
 

ID	Name	Position	Description
1 sh-atc-02	Mary Wu	Air Traffic Controller, LATCC	Air traffic control, representing primary user of the STCA system
2 sh-pilot-03	Alen Zhang	Pilot, BA	Representing pilots of aircraft targeted by the STCA system
3 sh-atc-01	John Smith	Air Traffic Controller, MACC	Air traffic control, representing primary user of the STCA system
4 sh-caa-04	person 4	Senior Officer, Safety Regulation Group. CAA	Representing the regulatory authority on

Figure 7 XML solution of STCA

Requirements			
Stakeholders			
ID	Name	Position	Description
sh-ate-02	Mary Wu	Air Traffic Controller, LATCC	Air traffic control, representing primary user of the STCA system
sh-pilot-03	Alen Zhang	Pilot, BA	Representing pilots of aircraft targeted by the STCA system
sh-ate-01	John Smith	Air Traffic Controller, MACC	Air traffic control, representing primary user of the STCA system
sh-caa-04	person 4	Senior Officer, Safety Regulation Group, CAA	Representing the regulatory authority on air traffic sa

Figure 8 HTML representation based on XML document of STCA

## 7. Conclusions and future work

In this paper, we have introduced an XML-based management tool for system requirements, system architectures and their relationships. The use of the XML technology affords us the capability of easily exchange requirements and architecture information within and across organizations. At the same time, it also provides opportunities to subject the requirements and architectures to analysis by various internal and external tools. In fact, we are currently developing a suite of analytical tools for better presentation and consistency checking of the requirements and architecture documents. We believe that the three-step design approach for our XML-based requirements and architecture management system.xmlTRAM+ is generally applicable to the development of XML-based systems. It involves the development of a series of conceptual, logical and physical models for the systems concerned. Acknowledgement. We would like to thank Chan Kai Tan for his contribution in the development of xmlTRAM+.

## References

- [Han, 2000] J. Han. Experience with Designing a Requirements and Architecture Management Tool. In Proceedings of International Conference on Software Methods and Tools, Wollongong, Australia, November 2000, pages 179-188. IEEE Computer Society Press. [Han, 2001] J. Han. TRAM: A Tool for Requirements and Architecture Management. Australian Computer Science Communications, 23(1): 60-68, 2001. [Tools survey]Tools survey: Requirements management tools. 2001. <http://www.incose.org/lib/index.html>. [Tan, 2001] Chan Kai Tan.XML-based Requirement and Architecture Management. Honors Thesis of School of Network Computing, Monash University, Australia, October, 2001, page 14-29 [Routledge, 2002] N.Routledge,L.Bird and A.Goodchild. UML and XML Schema. Database Technology2002, Australia, Jan 2002, pp157-166 [L. Boldt,1999] Larry Boldt. Managing Requirements at the Object Level. Technology Builders Inc, 1999. [D.W. Bustard,1995]D.W. Bustard and P.J. Lundy.Enhancing soft systems analysis with formal modelling. In Proceedings of the Second IEEE International Symposium on Requirements Engineering, pages 164-171, 10662 Los Vaqueros

Circle P.O. Box 3014 Los Alamitos, CA 90720-1314, March 1995. IEEE Computer Society, [A.Finkelstein,2000]Anthony Finkelstein and Wolfgang Emmerich.The future of requirements management tools.Information Systems in Public Administration and Law, Austrian Computer Society, 2000. [I. S. Graham,1999] Ian S. Graham and Liam Quin. XML Specification Guide.John Wiley & Sons Ltd, 1999. [T. Hammer,1998] Theodore Hammer and Lenore Hu®man. Automated requirements management– beware how you use tools an experience report. In Proceedings of Third International Conference on Requirements Engineering, pages 34–40, 10662 Los Vaqueros Circle P.O. Box 3014 Los Alamitos, CA 90720-1314, April 1998. IEEE Computer Society. [Bass, 1998] L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice.Addson-Wesley, Reading, MA, USA, 1998. [Emmerich, 1999] W.Emmerich, A. Finkelstein, C. Montangero, S. Antonelli, S. Armitage, and R. Stevens. Managing Standards Compliance.IEEE Transactions on Software Engineering, 25(6):836–851, 1999. [W3C, 2000]W3C Working Group. Extensible Markup Language (XML) 1.0. <http://www.w3.org/TR/REC-xml>