Structured Repository of Information Related to Software Certification

Ishwaree Argade
MEng (Computer Science)
Department of Computing and Software
McMaster University

June 20, 2014

Abstract

Software certification is the process of collecting evidence that the (software) parts of a software intensive system satisfies its requirements. As this is a rather new field, it is important to collect current materials on this topic, to serve as reference material. This is the particular task described here: choosing, indexing and classifying various materials pertaining to software certification, with an eye towards searchability and reuse. For concreteness, rather than selecting all materials, three areas were chosen: challenge problems, course modules, and certified libraries (with their accompanying tools). The data thus collected was made into a repository. A choice was made to use standard technologies (XML, Schema, XSL, etc.) for this purpose, to leverage the great many tools already in existence.

Throwing data into a repository is insufficient for searchability and reusability. At the very least, the data needs to be described in a relatively uniform manner. Thus, after an initial data gathering process was done, commonalities in the meta-data associated with each area (challenge problems, etc.) were identified. These were then codified in a Schema (using XSD), one per area. The design of these schemas is explained, and their adequacy is verified by instantiating it with some data. For human understandability of the results, we also created some XSL to display the data, guided by the schema.

Contents

1 Introduction **3**

2 Global Requirements **6**

3 Overview of Software Repository **8**
3.1 Challenge Problems 8
3.2 Course Modules 9
3.3 Certified Libraries and Tools 10

4 Methodology **11**

5 Challenge Problems **14**
5.1 Schema Design 16
5.2 Implementation 18
5.2.1 Common Schema 18
5.2.2 Schema for Execution Environment 20
5.2.3 Challenge Problem Schema 21
5.3 Viewing Data in Browser 23

6 Course Modules **25**
6.1 Schema Design 26
6.2 Implementation 27
6.3 Viewing Data in Browser 29

7 Libraries and Tools **30**
7.1 Schema Design 32
7.1.1 Attributes Listing: Libraries 32
7.1.2 Attributes Listing: Tools 33
7.2 Implementation 33
7.2.1 Common Schema for Libraries and Tools 34

7.2.2 Schema for Libraries 34
7.2.3 Schema for Tools 35

8 Testing **38**
8.1 Test Cases 38
8.2 Case Study 41

9 Conclusion **43**

A Attribute Listings **44**

B Schema Implementations **52**

Chapter 1

Introduction

Software Certification deals with the process of certifying a system containing some software inside it, but restricting the certification process to the software aspect only [Ben11]. The certification process ensures the reliability and safety of the software system to be certified listing all the information necessary for its assessment. It encompasses the wide range of formal, semi-formal and informal assurance techniques, which include even formal verification of security policies, system simulation, testing and code reviews [DF05]. Thus, the certificates can have different types and certification processes follow various mechanisms.

The most popular approach for software certification is the process based certification of systems. The process through which a software system is developed is evaluated rather than evaluating the final product. As many software certifiers find the evaluation of software process easier than evaluation of the product itself, process based certification is widely used [AW10]. One reason for this is that it is not possible to test the final product entirely even with the help of a huge number of test cases. Hence, the focus is given on certain supportive evidences which would guarantee the quality of the software systems. Secondly, it is difficult to determine the metrics/attributes essential in assessing the final software product, more emphasis is given on the software process instead [AW10]. Some examples of this approach, like ISO 9000 and CMMI, certify that the proper engineering methods and processes are followed to manufacture the product [Voa14].

Though process based certification is a general approach, it does not guarantee the reliability of the software as it focuses only on the process and not on the individual product. It certifies overall products and not a specific product. Thus, another approach called product based certification

is put forward. A detailed analysis of this aspect of software certification is found in the paper by Wassyng, Maibaum, and Lawford [AW10]. According to them, the goal of the certification should be to ensure that the product satisfies certain characteristics by assessing some measurable attributes of the product. This approach to the software certification believes that there should be a mandated software development process, which would guarantee the quality of development process of the product, and then the product can be evaluated without consideration of the actual process followed to develop a specific product [AW10].

Another certification method based on product based approach is proposed by Voas [Voa14]. According to him, by hiring a third party to issue software certification based on end users' feedback provides more unbiased and reliable software certification. Using this concept, he proposes a certification process involving automated methods to assess the behaviour of the software and to avoid the problem of miscertification [Voa14].

Software development nowadays widely follows reusability of components. Reuse of components is an important factor to reduce the cost of software development. Thus, the reliability of the part to be reused has to be evaluated. One method to determine the reliability of software that builds the structural model and usage profile of software components and then evaluates it against a set of test cases is given by Wohlin and Runeson [CW94] and is applicable to both part as well as system certification.

A software certification management system is used for management of certification [DF05]. It stores the information about different systems and varieties of certificates along with the entire certification history of the specific system. One of the challenges related to software certification is storing and providing the useful information. Hence, the goal of this report is to create a repository to store some material in various areas related to Software Certification.

This report focuses on three areas related to Software Certification namely Challenge Problems, Course Modules and Certified Software Libraries and Verification Tools to index some available material in the respective areas. These areas were already chosen by the principals running the Software Certification project at McMaster [McC13]. The objective is to design a repository that would store relevant material for the areas related to software certification. The central structure of the repository is shown in 1.1. The repository has five main areas. To give this project a reasonable scope, it was decided that only the first three areas in Fig. 1.1 would be pursued at this time. Challenge problems are sets of prototypes in the software certification area. More details about the challenge problems are given in section

3.1 and chapter 5. Relevant course modules are modules related to safety critical systems, real time systems and embedded systems. Details are explained in section 3.2 and in chapter 6. The third area contains libraries and tools. Libraries either can be a part of a verification tool or it can be a library that is verified by a verification tool. The repository intends to store both the libraries and verification tools. More information is in section 3.3 and chapter 7.

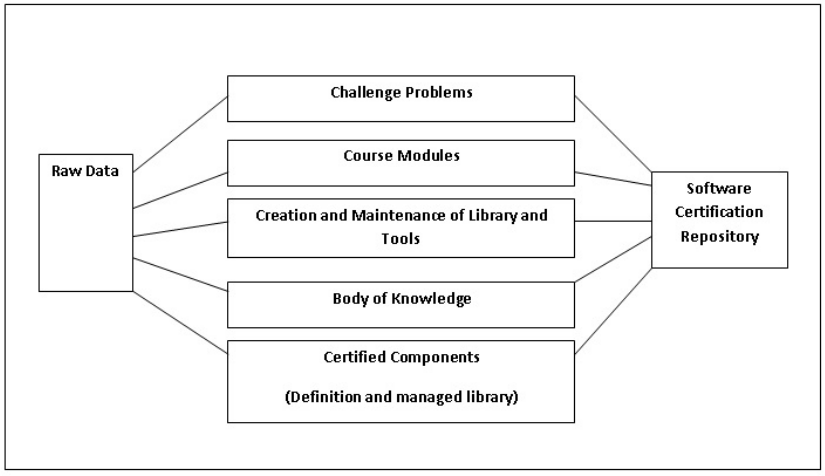

Figure 1.1: Design of Software Repository

The latter chapters introduce the three areas (Challenge problems, Course Modules and Certified Software Libraries) in detail, the idea of the software repository meant to store available material around these three areas, their schema designing and testing processes, finally concluding with future scope.

Chapter 2

Global Requirements

The first task here is to find the reference material in all the three areas (Challenge Problems, Course Modules, Certified Libraries and Tools). The next step is to provide a mechanism to store the gathered material in a uniform format. This can be accomplished be designing a schema using the meta-data for each area. The material is analyzed and various elements describing the particular area are found. The meta-data gathered can contain specific information. Hence, the next requirement would be to find the common elements in all the meta-data gathered to figure out the attributes for the schema. This then builds the foundation for schema implementation for each of the chosen areas.

The requirements can be listed as follows:

- **Finding reference material:-** The primary requirement is to find the reference material related to three chosen areas. For example, sample challenge problems like SAT Challenge, CADE ATP challenge etc., some sample course modules containing software certification contents, a set of certified libraries and verification tools were selected as reference problems for challenge problem, course modules, libraries and tools areas respectively. This completes the data gathering process for each area.

- **Analyzing reference material:-** The gathered data needs to be stored in a uniform structural format that makes it searchable and reusable. Schema design imposes a logical structure on the data. It organizes the information in a structured way, which makes it easy to search, index and visualize. Thus, a schema is required here to

store the material in a uniform manner. Material around the three areas is independent of each other and is characterized by different attributes. Thus, the repository would have a schema design for each of the components of the repository. So, there would be separate schemas for challenge problems, course modules, libraries and tools.

The schema designing process can be started by analyzing the gathered sample data material. For example, after analyzing twelve course modules in software certification areas, element set for each course module is found out. Element set is the important information that should be stored about the particular course module. This would give us twelve elements sets which can be called as meta-data for each of the twelve course modules. Similarly, this can be done for the other areas.

- **Finding common elements for each area:-** The meta-data can be specific to the corresponding data sample. For example, as explained above, after analyzing twelve course modules we get twelve sets of meta-datas, one for each course module. The meta-data describes each course module. So, the information can vary depending upon the course module content, their aims, materials used, assessment structure and rules. Thus, there is a need to find the commonalities in all the meta-data that would help to structure the important generic information about the course modules. For example, all the course modules have some description, module name, code, high level course content description, rules etc. This information can be stored in a structured format.

- **Schema Designs:-** The common information can be formulated in schema designs, which would allow users to store and index the material around the three areas. Thus, the respective schemas can be designed using the common elements for challenge problems, course modules, certified libraries and tools.

The corresponding schema designs are then implemented using standard technologies like Xml Schema Definitions (XSDs) [XSD11], Extensible Markup Language (XMLs) [XML10] and Extensible Style Sheets (XSLs) [Bri13b].

Chapter 3

Overview of Software Repository

The design of the repository is provided by the principals running the Software Certification project at McMaster, as mentioned earlier. The report considers first three components of this repository namely Challenge Problems, Course Modules and Creation and Maintenance of Library and Tools. This chapter gives a brief overview of the three areas. The detail explanation of all the three areas is given in subsequent chapters.

3.1 Challenge Problems

The Challenge Problem part of the repository is meant to store some challenge problems in the area of software certification. Challenge problems are sets of prototypes in the software certification area. Examples of various challenge problems are SAT Challenge [SAT12b], SMT COMP challenge [SMT12], CADE ATP challenge [CAD13] and Pacemaker challenge [Pac07]. SAT challenge consists of challenges around Boolean Satisfiability area. CADE ATP offers problems in automation deduction area. SMT COMP focuses on Satisfiability Modulo Theories Solvers. Pacemaker challenge is the challenge to the formal methods community hosted by Software Quality Research Laboratory (SQRL) [Pac07]. More details of all the challenge are given in chapter 5. Most of the challenge problems are offered every year as a part of various conferences and workshops held for software communities. For example, the CADE ATP System Competition (CASC) is held at each CADE and IJCAR conference. CASC evaluates the performance of sound, fully automatic, classical logic ATP systems. CASC

evaluates the performance of sound, fully automatic, classical logic order ATP systems.

Challenge problems are offered in various areas such as boolean satisfiability (SAT solving), automation, object orientation, Satisfiability Modulo Theories etc. Some challenges are only open to academia while some are open to both industry and academia. Every challenge has a different set of rules, expected format of solution and system requirements to test the solutions. The repository needs to store these kinds of challenges. However, all the information related to challenge problems should be stored in such a way that would make challenges searchable and reusable. Hence, the goal is to design a schema that would capture all the generic information such as format of the problem, expected solution, rules for the challenge, available solutions, deadlines etc. for all the challenges and store in a uniform format.

3.2 Course Modules

Course modules intend to have all the information about the courses involving topics in software certification. The course modules typically have some aims (key objectives on higher level), pre-requisites, content, assessment format and learning outcomes (point out outcomes in detail). The repository needs to store mainly this information. Other information such as professors, feedback, rules, workload, credits is also stored if available. As mentioned earlier, the schema is meant to be flexible and should allow storage of all the available information. So, only the available information is tagged. The schema is designed to store data for a range of course modules offered at various universities. The rest of the details and entire list description are explained in chapter 6. Relevant course modules are modules related to safety critical systems, real time systems and embedded systems. Some examples of such course modules are modules like 'Foundations of System Safety Engineering', 'Hazard and Risk Assessment' and 'Real-Time Embedded Systems Programming' offered at York University and Ryerson University. Around twelve course modules from various universities are considered here as reference materials and then as a test data set.

The information about course modules again needs to be stored in the uniform format. Hence, the separate schema for course module is needed. The schema mainly targets all the time independent information like aim of the course, learning outcomes, contents, assessment format etc. as explained above.

3.3 Certified Libraries and Tools

The last component manages information about libraries and verification tools. Libraries are of two types. It can either be a library that is a part of a verification tool or it can be a library that is verified by a verification tool. For example, Coq [COQ] is a tool used for verification of libraries and contains a number of libraries [COQ]. Many libraries can be a part of a tool and can be verified by a tool. The repository mainly targets to save the information about the libraries. However, as tools are used to verify the libraries, it is useful to store the tools as well in the repository.

Around eleven libraries and three tools are considered as reference material for this. Names and more details about these libraries are given in chapter 7. The libraries are analyzed and attributes for the schemas are figured out. The schema for the library should be able to tag all the data related to a library such as its current versions, contents, downloading, tools used for compilation or verification, examples, references and dependencies, if any. The schema stores the source codes in the form of links. Details of the location, format and the size of the downloadable files are point to the source code. Along with these elements, some additional elements like execution environment, functionalities, getting the tool are added to the tools' schema. However, even though the tools' and the libraries' schemas contain some common elements they have their own attributes too and needs to be stored separately. Thus, they have their own attribute listings and implementations. Libraries and tools are stored using different schemas.

Chapter 4

Methodology

Data gathering is the first requirement here. Data samples for each area are searched on the web. Web sites for challenge problems can be found out via web sites of their associated conferences and workshops. For example, SV-Comp challenge for software verification is a part of International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) [TAC13]. Thus, all the information about SV-Comp challenge 2013 can be found out via web site of TACAS 2013. Course modules can be found out on various universities' sites. However, relevant course modules can be filtered out using keywords like safety critical systems, real time and embedded systems. Specifications about choosing the relevant course modules were given by again principals of Software Certification at McMaster (McSCert) [McC13]. So, modules containing non-safety critical systems are considered here. However, as use of this schema depends on McSCert, the schema can be modified further to accommodate relevant course modules. Similarly, libraries and verification tools can be found out by using relevant keywords and looking at the contents of the libraries as well as the certification details.

The material associated to each area is loosely bound, meaning that they have various optional elements and can be stored effectively using XMLs instead of relational databases. The key purpose of the repository is to provide a flexible, simple database that would be easy to search, query, update and maintain. Further use of this repository would be decided by the principals running the Software Certification project at McMaster [McC13]. Relational databases impose tight dependencies and constraints on the data making it complex. This is not helpful here, as the data doesn't have dependencies and and also has too many optional attributes. Another approach is to col-

lect all the documents in a simple text/excel file under different headings. However, this approach is error prone and not as efficient as XMLs. There can be mistakes while inserting/filtering and retrieving the data. Additionally, data parsing, querying and rendering is inefficient and complicated. XML databases are used to store data persistently in XML document format [XML10]. So, similar to the rows in relational databases tables, we have XML documents. Data is stored in XML documents according to the corresponding XML schema design (XSD) [XSD11]. This would create a XML database. The database can be queried effectively using XML Path Language (XPath) [XPa99] and XML Query (XQuery) [XQu01]. XSLs can be used to render the XML data files in the browser, thus allowing users to create views for the stored documents.

At this stage, the main focus of this report is to design and implement the schemas for the three areas of the repository and use simple XSLs to display the XML data in the browser. This completes the fundamental step for the repository. However, as a future work, an effective but simple web portal can be designed as a front end to this XML repository that would allow users to insert, update, search and manipulate the XML documents in the database without displaying the underlying schema details to the end user. This would also create the scope for developing interactive web pages instead of just displaying data using static html pages. Hence, by considering its future application, use of XML/XSD/XSL approach adds simplicity, flexibility and efficiency to this type of repository rather than creating a rigid relational database or just indexing documents in the text files.

As explained earlier, the repository would have a schema design for each of the components of the repository. So, there would be separate schemas for challenge problems, course modules, libraries and tools. The schemas would be created as XSDs that would cover all the required parameters to store the required data for that component. Prior to the actual schema implementation, some data samples are analyzed and attributes are figured out as explained in the requirement gathering process. The schema is then implemented using the attribute listing. The corresponding XMLs are validated against the corresponding XSD schemas using an online tool [Bri13a]. Finally, XSLs designed according to particular schemas of the components of the repository, are used to view XMLs in the browser. An online XSL transformer tool is used to transform XML using the respective XSL into HTML code [Bri13b]. This HTML is then viewed in the browser. This process is repeated for all the components of the repository.

The subsequent chapters of the report document the entire process of

schema designing, XML, XSL creation and provide sample of schemas and overview of the testing process. The source code of the project is stored in a git repository at `https://github.com/ish1289/MEng_Project_Final`.

Chapter 5

Challenge Problems

Challenge problems are sets of prototypes of problems in software certification area. The software repository intends to store all the available and relevant challenge problems including both solved and unsolved challenge problems. The solution is also saved if it is available. Challenge problems can be a part of several conferences held for the software community. For example, there are challenges called SAT challenge [SAT12b], CADE ATP challenge [CAD13], Pacemaker challenge [Pac07], SMT COMP challenge [SMT12], SV-COMP challenge [SV13] etc. have been offered as a part of various conferences and workshops (SAT Conference [SAT12a], CADE Conference [CAD12], TACAS [TAC13], IJCAR [IJC12]). Each challenge has different dimensions. The sample dataset for challenge problems consists of following challenges:-

- **SAT Challenge:** SAT challenge revolves around the problems in SAT solving area. The SAT Challenge is a competitive event for solvers of the Boolean Satisfiability (SAT) problem. The event organized in 2012 is considered here. It is organized as a satellite event as a part of International Conference on Theory and Applications of Satisfiability Testing in 2012. The details of this challenge can be found out at `http://baldur.iti.kit.edu/SAT-Challenge-2012/` [SAT12b]

- **The CADE ATP System Competition:** This competition offers problems in automation deduction area. This dataset considers challenge problem offered 2013. The CADE and IJCAR conferences are the major forums for the presentation of new research in all aspects of automated deduction. To stimulate ATP research and system development, and to expose ATP systems within and beyond the ATP

community, the CADE ATP System Competition (CASC) is held at each CADE and IJCAR conference. The details can be found out at `http://www.cs.miami.edu/~tptp/CASC/` [CAD13]

- **Satisfiability Modulo Theories Competition (SMT-COMP):** It focuses on Satisfiability Modulo Theories Solvers. The SMT workshop will include a block of time to present the competitors and results of the SMTCOMP competition. The Satisfiability Modulo Theories Competition (SMT-COMP) arose from the SMT-LIB (Satisfiability Modulo Theories Library) initiative to spur adoption of the common, community-designed SMT-LIB formats, and to spark further advances in SMT, especially for verification. The dataset considers problem offered in 2012. Details are found at `http://smtcomp.sourceforge.net/2012/index.shtml`. [SMT12]

- **Competition on Software Verification (SV-COMP):** This competition offered challenge problems in the area of software verification. There are several new and powerful software-verification tools around, but they are very difficult to compare. The reason is that no widely distributed benchmark suite is available and most concepts are only validated in research prototypes. This competition wants to change this. Thus, one of the goals of this competition is to establish a set of benchmarks for software verification in the community. This means to create and maintain a set of programs together with explicit properties to check, and to make those publicly available for researchers to be used in performance comparisons when evaluating a new technique. The details of the challenge considered here are found at `http://sv-comp.sosy-lab.org/2013/index.php` [SV13]

- **Pacemaker Formal Methods Challenge:** This is the challenge to the formal methods community hosted by Software Quality Research Laboratory (SQRL). The details are found at `http://sqrl.mcmaster.ca/pacemaker.htm`. [Pac07]

The requirements and specifications depend upon the actual challenge problem and the committee who is putting forward this challenge. As the purpose of the repository is to collect all the challenge problems, a general schema which would be able to catch all the information of diverse challenge problems is needed. The schema provides a schematic structure to the repository in order to store challenge problems.

5.1 Schema Design

The first step in schema designing process is to analyze different challenge problems and try to find out some attributes that are common in all the challenges. The collective attributes are the various parameters in different challenges that make it possible to preserve their information in a structured manner.

As specified earlier, the schema is designed using XSD. Thus, all the information is tracked in the form of tags using XMLs. The XMLs of various challenges are then validated against the same created schema. The schema portrays the general structure for all the challenges still allowing users to embed challenge problem specific information in the XMLs.

Meta-data in the form of attributes is collected. The meta-data depicts the information of the particular challenge. For example, in case of SAT challenge, its organization, location, tracks allowed, its assessment, hardware/software requirements, supporting documentation, available solutions, result details, benchmark details, input/output specifications are noted down. This is done for five challenge problems mentioned above.

The next task is to find common attributes using all the meta-datas. The common attributes eliminate all the challenge specific information, but keep the generic information such as location, supporting documentation, deadlines, related tools details, description details and contact details. The challenge specific information also can be stored in a generic format. For example, each challenge has different rules, assessment process and execution environment. Some challenges have benchmark problems which are sets of small challenges and they have to be solved before facing the actual challenge. However, all this challenge specific information can also be stored in a generic way. The schema provides tags to store generic information along with the tags which points to the links giving specific information about the challenges.

The attribute list starts with the challenge name. Challenge name is unique for each challenge and thus acts as a primary key. The next attribute denotes the area of the challenge (SAT Solving, Automated Deduction, Object Oriented Systems). Challenge description contains information such as description, associated conference, start and end date, series information and location. Each rule can be tagged with generic attributes like category, description, input/output requirements and links that point to specific information. All the supporting documentation is stored using document name, its description and the link to the document. All types of documents including books, online materials, tutorials and discussions can also be stored in

this way.

Every challenge has different methods for assessment. However, some generic attributes can be used to store all the assessment related information for all the challenges. Assessment description, jury contact details (name, phone, email, web page link), score details and link to specific information together describe the whole assessment information. The score details may contain different points for different levels. So, it is described using special attributes 'points' and 'description'. A challenge can be open for all people or academia or industry people. The participant tag along with the description denotes information about the participants. Some challenges have benchmarks as mentioned earlier. Each benchmark has a category, description, a specified format and a deadline. Deadline can be stored with and without announcement. All the implementation type details are explained in the next section in the listing 5.1. Specific information related to benchmarks is stored using links.

Every challenge problem specifies a format in which a solution should be provided. It can be an executable solution, a formal specification or some other format. So, all the allowed solution forms are stored by allowed forms attribute. Input/output requirements for challenges are different for different challenges but can be captured by generic attributes input/output requirements. Execution environment details such as compilers, processors, compatible OS, allowed libraries for the solution are captured using respective tags. The execution environment in most challenge problems describes the test environment in which the solution would be tested. The solutions for solved challenge problems are also stored in a similar way. Various deadlines information like name, description and date is captured using separate deadline tag. Most of the challenge problems allow the use of some external tools. This information is captured under tools. Results of the challenge problem if available are tagged under results. However, the results are stored in the form a link which points to the specific results information page. Some challenge problems also have information describing the changes from previous challenge. This information is captured using the description and the link to specific information if available for each change.

Attributes' names indicate the purpose of the attribute. Therefore, this listing of attributes provides a hierarchical structure to tag information about various challenge problems and thus makes it convenient to implement this schema design using XSD. The Listing A.1 in appendix A shows all the attributes derived after analyzing some challenge problems.

5.2 Implementation

The above schema design is implemented using XSD. The schema is designed to store a wide range of challenge problems and not for some specific challenge problem. Therefore, the schema contains many optional attributes which add to the flexibility to the schema. For example, in case of contact details, any available contact such as email, phone of web page link can be stored. Additionally, the schema does not even impose the constraint on number of phones, emails or web page links and allows collecting any number of contacts. Similar constraints are imposed on all other tags such as rules, supporting documents, assessment details, benchmarks, execution environment etc. This allows users to store the maximum amount of information using available details. For example, any number of rules can be stored even if some information is absent in the rules. One rule may contain input/output requirements while the other may only contain the description and the link. Hence, the schema is implemented in a flexible way by considering all the possibilities in a set of challenge problems. All the links are implemented using Xlink. This permits us to save the links along with the description of the link.

The whole schema is broken down into three schemas. The first one has all the elements common to the areas covered in this report meaning that it includes all the attributes common in the areas of challenge problems, course modules, libraries and tools. The second schema contains the elements required to tag detail information regarding execution environment. The third schema is the actual main schema for challenge problem that includes the above two schemas. An online tool is used to validate XMLs against this schema [Bri13a].

5.2.1 Common Schema

As described earlier, this schema represents all the common attributes to the three areas covered. This is done to avoid redundancy in implementing the entire schema. The common parts mainly contains information of common types like contacts, supporting documents, related links and tools, which is common to all the three areas. This schema would be included in all the main schemas for challenge problems, course modules, libraries and tools. The Listing 5.1 displays the pseudo code for the common schema and the listing B.1 in appendix B shows the code of common schema. All the pseudo codes given here are written using notation similar to Clafer notation which uses regular expression notation in its types [Cla13].

Listing 5.1: Pseudo Code : Common Schema

```
\\ Schema Common to Challenge Problems, Course Modules,
    Certified Libraries and Tools
\\ Contains declaration of all common types

\\ Stores all the contact details
abstract Contacts:
abstract Contact [1..]
 name: string?
 description:  string?
 Phone: string*
 Email: string*
 Link: XLink*

\\ Stores all relevant tools details
abstract Tools:
abstract Tool: [1..]
   name : string
   description : string?
   link : XLink*

\\ Tags all the documentation information
abstract documents :
abstract document :[1..]
   name: string
   description : string?
   link : XLink*

\\ Tags all the links specific to the area
abstract relatedLinks :
 Link: XLink*

\\Stores all the specific results information in the form
    of links
abstract results:
 Link: XLink*

\\Stores details
abstract DescriptionType:
  name: string?
   description: string?
   Link : XLink*
```

5.2.2 Schema for Execution Environment

This schema represents particularly the information specific to the execution environment for the challenge problems. It separates all the execution environment data from the other elements of challenge problems such as its description, assessment details, contacts etc. This schema stores data like expected solution format, libraries, compilers, processors and OS permissible to solve a challenge and all the elements needed to describe the required solution as well as existing solutions. The Listing 5.2 shows the pseudo code for Execution Environment Schema and the listing B.2 in appendix B gives the code for the same.

Listing 5.2: Pseudo Code : Execution Environment Schema

```
1  \\ Schema describes all the types required to store the
       execution environment details for challenge problems
2
3  abstract inputRequirements:
4  \\ Stores the entire input requirement specified for the
       challenge problems
5    abstract inputRequirement : string*
6
7  abstract outputRequirements:
8  \\ Stores the entire input requirement specified for the
       challenge problems
9    abstract outputRequirement : string*
10
11 \\Stores libraries
12 abstract libraries:
13   abstract library : [1..]
14     name : string?
15     description : string?
16     link :XLink*
17
18 \\Stores compilers
19 abstract compilers:
20   abstract compiler : [1..]
21     name : string
22     description : string?
23
24 \\Stores processors
25 abstract processors:
26   abstract processor : [1..]
27     name : string
28     memory : string
```

```
29      description : string?
30
31  \\Stores OS
32  abstract OSUsed:
33    abstract OS : [1..]
34      name : string
35      version : string
36
37  \\Stores Deadlines
38  abstract Deadlines :
39    abstract Deadline : [1..]
40      name : string
41      actual deadline : datetime
42
43  abstract executionEnvironment :
44     description : string?
45     libraries : librariesType?
46     compilers : compilersType?
47     processors : processorsType?
48     OSUsed : OSUsedType?
49
50  abstract expectedSolution :
51     AllowedForms :string?
52     inputRequirements : inputRequirementsType?
53     outputRequirements : outputRequirementsType?
54     executionEnvironment : executionEnvironmentType?
55     deadlines : deadlinesType?
56     allowedSubmissions : string?
```

5.2.3 Challenge Problem Schema

This is the main schema for challenge problems. The structure of the schema is according to the schema design described earlier. It includes the two supporting schemas viz. Common Schema and Execution Environment Schema explained in the above subsections. The Listing 5.3 displays the main elements in challenge problem schema and the listing B.3 shows a part of the implementation.

The entire code for schema can be found `https://github.com/ish1289/MEng_Project_Final/tree/master/XmlSchema_ChallengeProblems/Schema`

Listing 5.3: Pseudo Code : Challenge Problem Schema

```
1  \\ Include common schema and execution environment schema
2  \\ Root element
```

```
abstract ChallengeProblems :
 abstract Challenges : [1..]
   ChallengeName : string
   Area : string?
   abstract ChallengeDescription : ?
  Description: string?
  abstract ChallengeDate : ?
    To : Date
    From : Date
        Location : string?
        AssociatedConference : string?
        Part of Series : string?
   abstract Rules : ?
       abstract rule : [1..]
         Category : string
         description : string?
         \\inputRequirementsType : defined in execution
             environment schema
         inputRequirements : inputRequirementsType?
         \\outputRequirementsType is defined in execution
             environment schema
         outputRequirements : outputRequirementsType?
         Link: XLink*
   \\DocumentationType : defined in common schema
   Documentation : DocumentationType?
   Year : year?
   abstract AssessmentDescription : ?
   \\ Stores all assessment details
     description : string?
     Link : XLink*
     \\ContactsType : Defined in Common Schema
     Jury : contactsType?
     abstract ScoreDetails : ?
       abstract Score : [1..]
         Points : string?
         description :string?
   abstract Participants : ?
     participant : string+
     Link :XLink*
   abstract benchmarks : ?
     abstract benchmark : [1..]
       Category : string?
       description : string?
       Format : string?
       Timeline :duration?
```

```
46          Link: XLink*
47      \\ExpectedSolutionType : Defined in execution
            environment schema
48      ExpectedSolution : ExpectedSolutionType?
49      \\ ResultsType : Defined in common schema
50      Results : ResultsType?
51      abstract Changes : ?
52        abstract change : [1..]
53           description : string?
54           link : XLink*
55      abstract solutions : ?
56      \\ Stores solutions if available
57         abstract solution : [1..]
58            Format : string?
59            description : string?
60            \\inputRequirementsType : defined in execution
                 environment schema
61            inputRequirements : inputRequirements?
62             \\outputRequirementsType is defined in execution
                  environment schema
63            outputRequirements : outputRequirements?
64            \\executionEnvironmentType defined in execution
                 environment schema
65            executionEnviornment : executionEnvironmentType?
```

5.3 Viewing Data in Browser

The data stored in the repository is in the form of XMLs that are validated against the above schema designed for Challenge Problems. Hence, some mechanism is needed to visualize those XMLs in the browser making them easy to interpret and read. This is done using XSLT. XSL is used to transform XML code into HTML code that is read by the browser and thus displays the contents of the XML file in the form of a html page.

The XSL for challenge problem reads the contents from the XML files according to the challenge problem schema and displays the information in a proper format if it is present in the XML file. An online tool is used to transform XML with XSL [Bri13b]. The tool accepts both the XML and XSL and then transforms XML code into HTML code understood by the browser. This html file is then viewed in the browser. Thus, XSL is used to display the data stored in the repository in readable format. The entire code for challenge problem XSL is found `https://github.com/ish1289/MEng_`

`Project_Final/tree/master/XmlSchema_ChallengeProblems/XSLT`. The testing of the above schemas and XSL along with a case study is described in detail later in the report.

Chapter 6

Course Modules

The software repository collects some course modules related to software certification area. Thus, the main objective behind this schema designing process is to identify the essential elements in several programs offered in these areas. These elements should provide a schematic structure to be able to tag all the information related to various courses. Some examples of course modules are courses related to safety critical systems, embedded systems and real time systems. Hence, relevant course modules are those, which are related to some systems containing software configuration. The sample set of course module used here contains nine course modules offered at York University, Canada [Yor13], one course module offered at Victoria University, Australia [Vic], one at university of Waterloo, Canada [Wat14] and one at McMaster University, Canada [MaC13]. The sample dataset of course modules contains following course modules.

- **Course Modules offered at York University:** York University in Canada offers nine course modules as a part of MSc program in Safety Critical Systems. The details about each module include course contents, aims, learning outcomes, assessment details, rules of the modules etc. All this information can be found out at `https://www.cs.york.ac.uk/postgraduate/taught-courses/msc-scse/#tab-2`. [Yor13]

- **Real-Time Systems:** This is a course module offered at McMaster University. The module mainly focuses on teaching students how to build computer systems that can be trusted in situations where the system's response to external events must be both timely and accurate. The details are found out at `http://www.cas.mcmaster.ca/~downd/courses/4aa4/4aaout.htm` [MaC13]

- **Embedded Computer Systems:** The course module is offered at University of Waterloo. The module provides students with the knowledge and skills to design modern embedded systems. More information is located at `https://ece.uwaterloo.ca/~rpellizz/ECE423.php` [Wat14]

- **Embedded and Networked Systems:** Victoria University in Australia offers this relevant course module. It includes basic concepts of computer communication. Link to the course module is `http://www.vu.edu.au/units/ENE3202`.

6.1 Schema Design

Similar to challenge problem schema design, the first step here is again to analyze different course modules and figure out common elements. The elements altogether represent all the information of the course modules. Note that here the main focus is to collect all time independent data. Therefore, the major goal of an analysis of various challenges is to find out attributes such as learning outcomes, aim of the module, pre-requisites and contents rather than focussing on deadlines and timings.

The course module is offered in some school/university. This information should be tagged followed by module name and module code. Module name and module code uniquely identify the course modules. Then, the professors' details offering the course module should be stored. This is done by tagging name, description, available contacts like emails, phone and web site. The next tags capture the status of course module (core/optional/Active etc.) and whether it is full-time or part-time. Some courses are allowed for certain numbers of tracks i.e. are eligible for students in certain programs. Supporting information like number of credits, teaching term, feedback, description, location, results is also captured using the appropriate tags.

The major purpose in indexing such course modules is to get information about their aims, learning outcomes, topics covered in the courses, material used in the courses (books/online tutorials, links etc.). Information about the assessment is stored under different tag. All the assignments' and exam details using separate tags one for each assignment and exam under assessment tag. Some course modules use external tools. They are tagged under tools tag using appropriate sub-tags. Links are stored in most of the cases (aims/assessment details, tools etc.) to point to course specific information.

This schema is also implemented using XSD. Thus, all the attributes found by analyzing various course modules are organized in a hierarchical

structure making the translation to XSD easy. The Listing A.2 in appendix A shows all the attributes used to implement course module schema.

6.2 Implementation

Similar to challenge problems, various course modules have different amount of information. So, the schema should be able to catch all the available information for the course modules. Naturally, the implemented schema has many optional elements. Some examples are feedback, workload, allowed tools, information about books etc. Some tags like aim, rule, book, prerequisites etc. have no restrictions on number of elements. This allows users to record any number of elements. For example, there are many aims for a course module. Every aim is represented by "aim" tag inside outer tag "aims". Similarly, all rules, pre-requites, tools, learning outcomes, assignments and exams are recorded. The schema implementation follows the schema discussed in an above section. It also includes the common schema implemented given in Listing B.1. The pseudo code of common schema is given in listing 5.1. This schema again covers the general structure to tag all the attributes of course modules in diverse areas while allowing saving some module specific data, as well. The schema is then tested with some sample course module XMLs by using an online XML schema validation tool [Bri13a]. The testing details are discussed later in the report. The Listing 6.1 shows the pseudo code for course module schema and the listing B.5 shows the part of the implementation.

The complete code is found out at `https://github.com/ish1289/MEng_Project_Final/tree/master/XmlSchema_Course_Modules/Schema`

Listing 6.1: Pseudo Code : Course Module Schema

```
1  \\ Include common schema
2
3  \\ Root Element
4  abstract modules :
5    abstract module : [1..]
6      School/University : string?
7      Module Name : string
8      Module Code : string?
9      \\ConactsType : defined in common schema
10     Professors : ContactsType?
11     Status : string?
12     Required For : string?
13     abstract Allowed Tracks : ?
```

```
14          Track : string+
15        Number of Credits : string?
16        abstract Teaching Term : ?
17          Term : string+
18        abstract pre-requisites : ?
19          pre-requisite : string+
20        description : string?
21        abstract aims : ?
22          aim : string+
23        abstract LearningOutcomes : ?
24          LearningOutcome : string+
25        abstract Rules : ?
26          Rule: string+
27        abstract Workload : ?
28          Total Lecture Hours :string?
29          Each Lecture Time : string?
30          Total Private Study Time : string?
31          Assessment Time : string?
32        abstract Feedback : ?
33          description :string?
34          Link : Xlink*
35        abstract Content : ?
36          abstract Topics Covered : ?
37            description : string?
38            Link : XLink*
39          abstract Teaching Material : ?
40           abstract Material : [1..]
41            Name : string?
42            description : string?
43            Type : string ?
44            Link : XLink*
45          abstract Books : ?
46            abstract Book : [1..]
47              Name :string?
48              Type : string?
49              Author : string[1..]
50              Title : string?
51              Publisher : string?
52              Year : year?
53           start Date : date?
54           end Date : date?
55           abstract Assessment : ?
56            description : string?
57            abstract assignments : ?
58              abstract assignment : [1..]
```

```
59              description : string?
60              weight : string?
61              Link : XLink*
62            abstract exams : ?
63              abstract exam : [1..]
64                description : string?
65                weight : string?
66                Link : XLink*
67        \\ToolsType : defined in common schema
68         AllowedTools: ToolsType?
69         \\ RelatedLinks Type : defined in common schema
70         RelatedLinks : RelatedLinksType?
71         Location :string?
```

6.3 Viewing Data in Browser

Similar to challenge problems XMLs, course module XMLs also need a XSL to transform XML code into HTML code. This makes it able to be viewed in the browser. The XSL is implemented according to the course module schema given above. It checks if the information is present in the XML or not and then converts it into appropriate HTML code. The HTML code displays information in an eye pleasing and organized format. An online tool is used to transform XMLs using this XSL [Bri13b]. As discussed earlier, the tool accepts both XML and XSL, then transforms the given XML according to the XSL finally providing an HTML code as output. The code for course module XSL is found `https://github.com/ish1289/MEng_Project_Final/tree/master/XmlSchema_Course_Modules/XSLT`

The testing of the XSL with some sample XMLs is done and is later discussed in the report.

Chapter 7

Libraries and Tools

The third component of the repository saves the data related to libraries and tools. Libraries can be a part of the tool or can be verified using a tool. There is a many to one relationship between tools and libraries. Tools used for verification are considered here. The main aim here is to look for the certified libraries. Some of the certified libraries are verified by verification tools. So, storing information about the tools along with the libraries is useful considering the future scope in the area. Thus, the repository requires schemas to store some certified software libraries as well as some verification software tools used in the area of software certification. Libraries and tools are related to each other. So, they have some attributes like versions, overview, content files, extensions, contacts and documentation in common. However, as libraries are smaller than tools and can be a part of tool, tools have some different downloading ways and execution environment details. Libraries can also have attributes like examples, references and dependency details. Thus, it is helpful to store libraries and tools using separate schemas in the repository and hence two separate schemas for libraries and tools are implemented respectively. The further use of all this information on libraries and tools will be decided by McSCert [McC13] as mentioned earlier. The sample dataset for libraries contains following libraries.

- **The GNU C Library (glibc):** The GNU C Library is used as the C library in the GNU systems and most systems with the Linux kernel. The library defines the system calls and other basic facilities such as open, malloc, printf, exit etc. More information is at `https://www.gnu.org/software/libc/index.html` [Gli]
- **Bouncy Castle libraries for cryptography:** Bouncy Castle offers libraries for cryptography, one in java and one in C sharp. Both the

libraries are certified. Detail information about the C sharp project is at http://www.bouncycastle.org/index.html [BCa, BSh]

- **Microchip Software Certified Libraries:** Microchips MCUs and DSCs a set of certified software libraries and code examples. These libraries are mainly related to digital filters, DSP, encryption/decryption, maths and communication. The entire list of libraries is available at http://www.microchip.com/SoftwareLib.aspx. [MCH] The sample dataset contains four libraries viz. 'Microchip Certified Class B Safety Software Library for 16 bit and PIC32 MCUs', 16-bit CPU Self-test Library, SMPS Control Library and PIC32 Audio Equalizer (EQ) Filter Library .

- **OpenSSL:** OpenSSL is full-strength general purpose cryptography library. The details of the library are at http://www.openssl.org/. [SSL]

- **Standard Template Library:** The Standard Template Library, or STL, is a generic C++ library of container classes, algorithms, and iterators; it provides many of the basic algorithms and data structures of computer science. The link to the library is https://www.sgi.com/tech/stl/stl_introduction.html. [STL]

- **Alea: a library for reasoning on randomized algorithms in Coq:-** This library forms a basis for reasoning on randomised algorithms in the proof assistant Coq. The details of the library are found at https://www.lri.fr/~paulin/ALEA/. [Ale13]

- **A Coq Library on Floating-Point Arithmetic:** The library can be used by using Coq verification tools. More information about the library can be found out at http://lipforge.ens-lyon.fr/www/pff/. [DFC]

The sample dataset for tools consists of following tools.

- **The Coq Proof Assistant:** Coq is a formal proof management system. It provides a formal language to write mathematical definitions, executable algorithms and theorems together with an environment for semi-interactive development of machine-checked proofs. More information about Coq can be found out at http://coq.inria.fr/. [COQ]

- **Construction and Analysis of Distributed Processes:-** CADP ("Construction and Analysis of Distributed Processes", formerly known

as "CAESAR/ALDEBARAN Development Package") is a popular toolbox for the design of asynchronous concurrent systems, such as communication protocols, distributed systems, asynchronous circuits, multiprocessor architectures, web services, etc. Detail information is found out at `http://cadp.inria.fr/`. [CAD]

- **NuSMV: a new symbolic model checker:** NuSMV is a symbolic model checker developed as a joint project between: The Embedded Systems Unit in the Center for Information Technology at FBK-IRST The Model Checking group at Carnegie Mellon University , the Mechanized Reasoning Group at University of Genova The Mechanized Reasoning Group at University of Trento. It is an open architecture for model checking. Link to the tool is `http://nusmv.fbk.eu/`. [NUS]

7.1 Schema Design

The design process again begins with some sample libraries and tools mentioned above. Around eleven libraries and three tools are considered as reference material for this. They are analyzed and attributes for the schemas are figured out. The schema for the library should be able to tag all the data related to a library such as its current versions, contents, downloading, tools used for compilation or verification, examples, references and dependencies if any. Along with these elements, some additional elements like execution environment, functionalities, getting the tool are added to the tools' schema. However, even though the tools' and the libraries' schemas contain some common elements they have their own attributes too and needs to be stored separately. Thus, they have their own attribute listings and implementations.

7.1.1 Attributes Listing: Libraries

The analysis of some libraries such as Microchip certified libraries, COQ libraries, a PRL math library, glibc, Bouncy Castle library for Java and C sharp, SSL, and STL. The attribute listing starts with library name which is a mandatory attribute followed by optional attribute overview. The high level overview is tagged using overview description attribute and link is used to navigate to detail information page of the library. All the available versions including current and previous are stored using tags version name, description and link to the version. Attribute status is used to indicate whether it's a current or previous version. Content files in the library are

stored using name, description and link tags for each content file. Dependency details, examples, references, documentation and experimental library contents are captured in similar ways. This type of name, description hierarchy allows users to tag available information in a flexible way. Any number of links can be stored using link tags. Contributors are stored using the same way used in Challenge problems and Course modules. Information on how to download the library is captured using the tags description, size and links and compatible OS. The Listing A.3 in appendix A shows the elements for the libraries' schema.

7.1.2 Attributes Listing: Tools

After analyzing some tools like COQ [COQ], CADP [CAD], NuSMv [NUS]. Tools and attributes have some attributes in common. Content files, version details, name, documentation, related links, extensions, contacts, related tools and features are captured using the similar way. All the generic information is stored using mainly name and description tags and detail information is tagged using links. The major difference between the tools' and libraries' schema is the download methods and execution environment. Execution environment contains the languages used in the tool, input requirements, compatible OS, compiler and processor requirements. OS, compiler and processor requirements are stored using appropriate attributes like name, memory, version and description for each compiler, OS or processor. There can be three downloading formats for tools. Either it can be downloaded as source code or a binary file or in some other format. For source code only size, description and link is needed. However, for binary files compatible OS is also stored. Sometimes, there are some other ways to download the tools. This information is caught using description and links. The Listing A.4 in appendix A shows the elements for the tools' schema.

7.2 Implementation

As mentioned earlier, libraries and tools are related to each other. So, naturally they share some common elements which can be implemented separately. Moreover, both of these schemas also include the common schema discussed earlier in Listing 5.1. Similar to challenge problems and course modules, these schemas also contain the optional attributes. This allows the user to store only available information and adds flexibility. Adding no restrictions on the number of elements permits the schema to store multiple

elements like compilers, processors, versions, documentation, content files etc. The latter chapter talks about the testing of these schemas in detail.

7.2.1 Common Schema for Libraries and Tools

The schema mainly includes implementation of a few common elements such as version details, contents' details and extensions. The Listing 7.1 shows the pseudo code for common schema for libraries and tools and the listing in appendix B B.4 gives the implementation for the same.

Listing 7.1: Pseudo Code : Common Schema for Libraries and Tools

```
\\ Common Schema for libraries and Tools
\\ Contains complex Types common for both the schemas

abstract Versions :
  \\Stores all versions of libraries and tools
  abstract Version : [1..]
  \\ Has attribute Status (Current/Previous)
    name : string?
    description : string?
    year : string?
    Link : XLink*

abstract ExContentsType :
   description : string?
   content : descriptionType+

abstract ContentFiles :
   description : string?
   ContentFile : descriptionType+
```

7.2.2 Schema for Libraries

The Listing 7.2 contains the pseudo code for libraries' schema and the listing B.6 contains its implementation.

Listing 7.2: Pseudo Code : Schema for Libraries

```
\\ Include Common Schema and Common schema for libraries
   and tools

abstract libraries :
  abstract library : [1..]
    Name : string
```

```
6       \\descriptionType : defined in Common schema for
            libraries and tools
7       Overview : descriptionType?
8       \\versionsType : defined in Common schema for libraries
            and tools
9       versions : versionsType?
10      \\ExContentsType : defined in Common schema for
            libraries and tools
11      Experimental Library Contents : ExContentsType?
12      \\ContactsType : defined in common schema
13      Contributors : ContactsType?
14      \\ContentFilesType : defined in Common schema for
            libraries and tools
15      ContentFiles : ContentFilesType?
16      abstract Downloads :
17       abstract Download : [1..]
18         description :string?
19         Format : string?
20         size :string?
21         Link : XLink*
22      \\ToolsType : defined in common schema
23      Required Tools : Tools Type?
24      Documentation : ?
25      abstract Dependency Details : ?
26      \\description type : defined in common schema
27       dependency : description type+
28      abstract Examples : ?
29        Example : description Type+
30      abstract References : ?
31        Reference : description Type+
32      \\Related Links Type : defined in common schema
33      Related Link : Related Links Type?
```

7.2.3 Schema for Tools

The Listing 7.3 gives the implementation for tools' schema and the listing B.7 shows its implementation.

Listing 7.3: Pseudo Code : Schema for Tools

```
1  \\ Include Common Schema and Common schema for libraries
       and tools
2
3  \\ Root element
4  abstract Tools :
```

```
5  abstract Tool : [1..]
6    Name : string
7    \\description Type : defined in Common Schema
8    Overview : description Type?
9    \\versions Type : defined in Common schema for libraries
         and tools
10   Versions : versions Type?
11   abstract Functionalities : ?
12     \\description Type : defined in Common Schema
13     Function : description Type?
14   abstract Intended Users : ?
15     User :  string+
16   \\ExContents Type : defined in Common schema for
       libraries and tools
17   Extensions : ExContents Type?
18   \\Contacts Type :defined in common schema
19   Contacts : Contacts Type?
20   \\Contents Type : defined in Common schema for libraries
         and tools
21   Contents : Contents Type?
22   abstract How To Obtain : ?
23    abstract Source : [0..]
24      size : string?
25      description :string?
26      Link : XLink*
27    abstract binary : [0..]
28      description : string?
29      format :string?
30      size : string?
31      link : XLink*
32    Others : descriptionType*
33    abstract executionEnvironment
34     abstract languages : [0..]
35       language : string+
36     abstract inputRequirements : ?
37       inputRequirement : string+
38     abstract Compatible Compilers : ?
39       compiler : description Type+
40     abstract Compatible Processors : ?
41        abstract Processor : [1..]
42         Name : string
43         Memory : string
44         Description : string?
45     abstract OSUsed : ?
46       abstract OS : [1..]
```

```
47          Name : string
48          Version :string
49      \\Tools Type : defined in common schema
50      Related Tools : Tools Type?
51      \\documenation Type : defined in common schema
52      Documentation : documentationType?
53      \\ RelatedLinks Type : defined in common schema
54      Related Links : Related Links Type?
```

The complete code for libraries' and tools' schema can be found out `https://github.com/ish1289/MEng_Project_Final/tree/master/XmlSchema_Libraries_Tools/Schema`

View Data in Browser

Similar to challenge problems and course modules, libraries and tools XMLs are viewed in the browser with the help of XSLs.

XSLs for both libraries and tools are located at `https://github.com/ish1289/MEng_Project_Final/tree/master/XmlSchema_Libraries_Tools/XSLT`

Chapter 8

Testing

The testing begins with preparing some sample XMLs for all the areas discussed. The whole schema design, implementation, XSLs implementation and testing process follow an iterative approach. The entire three areas viz. challenge problem, course modules and libraries and tools are tested individually. However, they follow the same testing process.

8.1 Test Cases

After the schema design and implementation, an XML is prepared from a sample challenge problem, course module, library and a tool. This XML is validated against the first version of the schema. Some test cases described in the table 8.1 are tested to check the strength, validity and effectiveness of the schema. According to the results, improvements are made to the schema. This process is repeated till the schema is tested against reasonable number of samples and passes all the test cases correctly without the further need of modification to an existing schema.

The set of XML samples is then transformed using the XSL designed according to the finally tested schema. The HTML file produced as an output is tested against some test cases given in table 8.1. This again follows the iterative approach till all the test cases are satisfied.

The table 8.1 lists down the test cases used for both XSDs and XSLs of challenge problems, course modules, libraries and tools. These test cases

Sr.No.	Test Case Name	Description	Applicable For	Result
1.	Information coverage	check whether all the information is covered	XSD/XSL	Passed
2.	Appropriate attribute tagging	All the information should be tagged appropriately.	XSD	Passed
3.	Allowance of specific information	Specific but important information for the challenge should also be included without affecting the general structure of the schema	XSD	Passed
4.	Flexibility	Not all the elements are present in all the samples. The schema should allow users to insert only available information. Checking the use of optional attributes.	XSD/XSL	Passed.
5.	Attribute coverage	Testing on reasonable number of samples to verify all the attributes in the schema or XSL are utilized	XSD/XSL	Passed
6.	Common schema	Use of common schema to implement common elements	XSD	Passed
7.	Look and feel for XSL	Checking whether all the information when viewed in the browser is properly aligned and look and feel of HTML page	XSL	Passed

Table 8.1: Test Cases

are applicable to all the schemas and all the scenarios. However, the use of test cases can be illustrated with the help of a small example. The first design of the challenge problem schema captured all the rules using just the rule tag unlimited times. so, the XML file would contain the structure $<$ $rules>< rule>< /rule>< rule>< /rule>< /rules>$. On the very first sample challenge, it worked fine for all the test cases. The schema provided mainly flexibility, information coverage. However, later while testing it on some other challenge, where rules had a lot of specific information, test cases failed. For example, the first test case "information coverage" failed because, the detailed information about the rule was not stored by the schema. The schema could store only the high level description of the rules. The second and third test cases failed because the schema had to store all the information only as a description. So, there was no other way to insert information about the input/output requirements or rule background or to store links. This was naturally hampering the flexibility of the schema though it was allowing multiple rules. Thus, the fourth test case also failed.

The above problem was fixed by changing the structure of the rule tag. More tags were added under the rule tag. This permitted the users to insert bigger rules' information in more categorized manner. Rule category was added to indicate the purpose of the rule. Rule description would store the main information about the rule. Some rules may contain input/out requirements. Appropriate tags and inner tags were added to catch this information. Rule information that is challenge specific is considered as detailed information. Links tag is added using XLink to store the links. Any number of links can be stored for a rule. Any of these inner tags category, links, input/output requirements are kept optional. So, if the rule is simple, only its description can be stored. If the rule contains only description and the link or the number of links, this can be caught as well. This new schema with new rule structure is again tested against the test cases using old and new sample challenge problems. All the test cases were passed as the schema covers all the information. It covers all the information, tags information properly, stores specific information and provides flexibility. In this way, the testing is carried out for all the elements in the schemas until all the test cases are satisfied on all the sample data sets. Fig 8.1 shows this sample transformation.

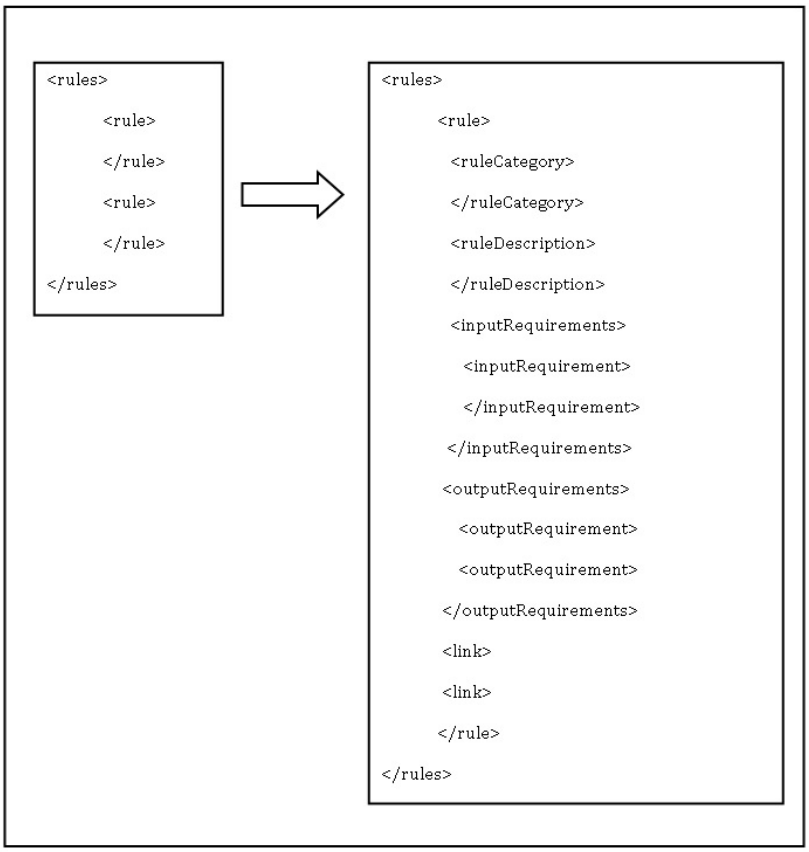

Figure 8.1: Testing Example: Change in 'Challenge Problems "rules" structure'

8.2 Case Study

This section provides a sample XML and its transformed HTML code for a challenge problem. The XMLs illustrate how the actual information is stored in the repository using the newly designed schemas and HTML files depict how the XMLs are transformed and seen in the browser using the newly implemented XSLs according to the corresponding schemas.

Note that the report just provides one sample XML and HTML for a challenge problem. However, the schemas and XSLs are tested against five XMLs to verify their validity and effectiveness and similarly XMLs and HTMLs can be obtained for other categories and data samples. The whole

set of XMLs and HTMLs for all the categories is stored in a git repository and can be viewed at https://github.com/ish1289/MEng_Project_Final

The https://github.com/ish1289/MEng_Project_Final/blob/master/XmlSchema_ChallengeProblems/Sample_Xmls/XML_Challenge_Problem_SAT_Challenge.xml gives XML for SAT challenge. The XML is prepared by extracting the information from SAT challenge website and then tagging it according to the challenge problem schema given in the Listing B.3 [SAT12b]. The XML is then validated using the online tool against the challenge problem schema [Bri13a]. The corresponding HTML can be found out at https://github.com/ish1289/MEng_Project_Final/blob/master/XmlSchema_ChallengeProblems/Sample_HTMLs/HTML_SAT.html.

Chapter 9

Conclusion

Software certification refers to the process of certification of the software part in the system. The report talked about some approaches to accomplish this. Component reusability is now getting popular in software certification. However, the component to be reused needs to have the appropriate certification and certification management system. One of the challenges in this area is the management of useful information regarding the certification. Therefore, the design of the software repository, which would manage information related to various areas around software certification, mainly focusing on the three areas of challenge problems, course modules and libraries and tools is proposed here.

Future tasks for the repository can be categorized into two streams. First, would be to find more samples for all the three components discussed here. A major challenge for the work presented in the report was data gathering. It is a challenging task to find the relevant data samples. This is true for all the three areas. The second, is to create and implement schema for the rest of the two areas of the repository. The schemas can be designed and implemented by following the same methods illustrated in the report. This would complete the implementation of the proposed repository. Additional future work after the implementation of the entire repository would be a simple front end as explained in chapter 4, that would manage data queries, updates, insertions and maintenance of XML documents in the repository efficiently. Again, the use of this repository would be decided by the principals working on Software Certification project at McMaster.

The report provides an effective mechanism to index some available material for the three areas around software certification using standard technologies such as XML, XSD and XSL.

Appendix A

Attribute Listings

Listing A.1: Attributes for Challenge Problem Schema

```
A. Challenge Name
B. Area
C. Challenge Description
   a. Description
   b. Challenge Date
      1. To
      2. From
   c. Challenge Location
   d. Associated Conference
   e. Part of Series
D. Rules
   a. Rule
      1. Rule Category
      2. Description
      3. Input Requirements
         i. Input Requirement
      4. Output Requirements
         i. Output Requirement
      5. Links
E.  Supporting Documents
   a. Document Name
   b. Description
   c. Link
F. Year
G. Assessment Description
   a. Description
   b. Link
   c. Jury
```

1. Jury Name
2. Description
3. Phone
4. Email
5. Web page Link

d. Score Details
i. Score
1. Points
2. Description

H. Participants
a. Participant
b. Participant Description Link

I. Benchmarks
a. Benchmark
1. Categories
2. Description
3. Format
4. Timeline
5. Link

J. Expected Solution
a. Allowed Forms
b. Input Requirements
i. Input Requirement
c. Output Requirements
i. Output Requirement
d. Execution Environment
1. Description
2. Libraries
i. Library
A. Name
B. Link
3. Compilers
i. Compiler
A. Name
B. Description
4. Processors
i. Processor
A. Name
B. Memory
C. Description
5. OS Used
i. OS
A. Name
B. Version
e. Deadlines

```
        1. Deadline
          A. Name
          B. Date
    f. Allowed Submissions
K. Allowed Tools
    a. Tool
        1. Name
    2. Description
    3. Link
L. Required Contacts
    a. Contact
        1. Name
        2. Description
        3. Phone
        4. Email
        5. Web page Link
M. Results
    a. Link
N. Changes from Previous Challenges
    a. Change
        1. Description
        2. Link
O. Solutions
    a. Format
    b. Description
    c. Input Requirements
    d. Output Requirements
    e. Execution Environment
```

Listing A.2: Attributes for Course Modules Schema

```
Modules
a. Module
    1. School/University
    2. Module Name
    3. Module Code
    4. Professors/Lecturers
    5. Status (Core/Optional)
    6. Required For (Full-time/part-time)
    7. Allowed Tracks
    8. Number of credits
    9. Teaching Term
    10. Pre-requisites
    11. Description
    12. Aims
```

15 a. Aim
16 i. Description
17 ii. Link
18 13. Learning Outcomes
19 14. Rules
20 15. Workload
21 i. Total lecture hours
22 ii. Each lecture time
23 iii. Total Private study time
24 iv. Assessment Time
25 16. Feedback
26 i. Description
27 ii. Link
28 17. Content
29 i. Topics covered
30 a. Description
31 b. Link
32 ii. Teaching Material
33 a. Name
34 b. description
35 c. Type (Slides/Case Studies/exercise)
36 d. Link
37 iii. Books
38 a. Name
39 b. Type (Required/Recommended)
40 c. Author
41 d. Title
42 e. Publisher
43 f. Year
44 vii. Assessment
45 a. Description
46 b. Assignments
47 1. Assignment
48 a. Description
49 b. weight
50 c. Link
51 c. Exams
52 2. Exam
53 d. Description
54 e. weight
55 f. Link
56 viii. Start Date
57 ix. End Date
58 x. Allowed Tools
59 a. Tool Name

```
        b. Description
        c. Link
      xi. Location
      xii. Results
   18. Related Links
```

Listing A.3: Attributes for Libraries' Schema

```
A. Libraries
  i. Library
  1. Name
  2. Overview
    a. Description
    b. Link
  3. Current Version
  4. Versions (Current/Previous descriptions)
    a. Version name
    b. Status (current/previous)
    c. Description
    d. Link
  5. Experimental Library Contents (Similar to Extensions)
    a. content
      i. Description
      ii. Link
  6. Contributors
    a. Contributor
      i. Name
      ii. Contribution Description
      iii. Email
      iv. Phone
      v. Web page link
  7. Content Files
    a. Content File
      i. Name
      ii. Description
      iii. Link
  8. Downloads
    a. Description
    b. Format (OS)
    c. Size
    d. Link
  9. Required Tools for compilation
    a. Tool
      i. Tool name
      ii. Description
```

```
        iii. Link
    10. Documentation
      a. Document
        i. Name
        ii. Description
        iii. Link
    11. Dependency details
      a. Description
      b. Link
    12. Examples
      a. Example
        i. Name
        ii. Description
        iii. Link
    13. Related Links
      a. Link
    14. References
      a. Description
      b. Link
```

Listing A.4: Attributes for Tools' Schema

```
Tools
o Tool
1. Name
2. Overview
  a. Description
  b. Link
3. Current Status
  a. Latest available version
  b. Description
  c. Link
4. Functionalities/Features
  a. Function/Feature
    i. Description
    ii. Link
5. Intended Users
  a. User
6. Extensions
  a. Extension
    i. Description
    ii. Link
7. Contacts
  a. Contact
    i. Name
```

```
        ii. Email
        iii. Phone
        iv. Web page link
    8. Contents
      a. Content
        i. Name
        ii. Description
        iii. Link
    9. How ToObtain
      a. Source
        i. Size
        ii. Description
        iii. Link
      b. Binaries
        i. Binary
          I. Description
          II. Format (OS)
          III. Size
          IV. Link
      c. Others
        i. Other forms
          I. Description
            II. Link
    10. Execution Environment
      a. Languages Used
        i. Language
      b. Input Requirements
        i. Input Requirement
      c. Compatible Compilers
        i. Compiler
          I. Name
          II. Description
      d. Compatible Processors
        i. Processor
          I. Name
          II. Memory
          III. Description
      e. Compatible OS
        i. OS
          I. Name
          II. Version
          III. Distribution
    11. Related Tools
      a. Tool
        i. Tool name
```

69 ii. Description
70 iii. Link
71 12. Documentation
72 a. Document
73 i. Name
74 ii. Description
75 iii. Link
76 13. Related Links
77 a. Link

Appendix B

Schema Implementations

Listing B.1: Common Schema

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:xlink="http://www.w3.org/1999/xlink">
3  <xs:import namespace="http://www.w3.org/1999/xlink"
       schemaLocation="http://www.w3.org/1999/xlink.xsd"/>
4  <!--Schema Common to Challenge Problems, Course Modules,
       Certified Libraries and Tools-->
5   <xs:complexType name="contactType">
6    <xs:sequence>
7      <xs:element name="name" type="xs:string" minOccurs
           ="0"></xs:element>
8      <xs:element name="contactsDescription" type="xs:string"
           minOccurs="0"></xs:element>
9      <xs:element name="phone" type="xs:string" minOccurs="0"
           maxOccurs="unbounded"></xs:element>
10     <xs:element name="email" type="xs:string" minOccurs="0"
           maxOccurs="unbounded"></xs:element>
11     <xs:element name="linkDescription" minOccurs="0"
           maxOccurs="unbounded">
12       <xs:complexType>
13         <xs:simpleContent>
14           <xs:extension base="xs:string">
15             <xs:anyAttribute namespace="http://www.w3.org
                   /1999/xlink"/>
16           </xs:extension>
17         </xs:simpleContent>
18       </xs:complexType>
19     </xs:element>
```

```
20    </xs:sequence>
21  </xs:complexType>
22    <xs:complexType name="contactsType">
23    <xs:sequence>
24      <xs:element name="contact" type="contactType" maxOccurs
          ="unbounded"></xs:element>
25    </xs:sequence>
26  </xs:complexType>
27  <xs:complexType name="toolsType">
28      <xs:sequence>
29        <xs:element name="tool" maxOccurs="unbounded">
30          <xs:complexType>
31            <xs:sequence>
32              <xs:element name="toolName" type="xs:string"></
                    xs:element>
33              <xs:element name="toolDescription" type="xs:
                    string" minOccurs="0"></xs:element>
34              <xs:element name="toolLinkDescription"
                    minOccurs="0" maxOccurs="unbounded">
35                <xs:complexType>
36                  <xs:simpleContent>
37                    <xs:extension base="xs:string">
38                      <xs:anyAttribute namespace="http://www.
                            w3.org/1999/xlink"/>
39                    </xs:extension>
40                  </xs:simpleContent>
41                </xs:complexType>
42              </xs:element>
43            </xs:sequence>
44          </xs:complexType>
45        </xs:element>
46      </xs:sequence>
47    </xs:complexType>
48  <xs:complexType name="relatedLinksType">
49      <xs:sequence>
50       <xs:element name="relatedLinkDescription" minOccurs
            ="0" maxOccurs="unbounded">
51          <xs:complexType>
52            <xs:simpleContent>
53                    <xs:extension base="xs:string">
54                      <xs:anyAttribute namespace="http://www.
                            w3.org/1999/xlink"/>
55                    </xs:extension>
56              </xs:simpleContent>
57          </xs:complexType>
```

```
58          </xs:element>
59        </xs:sequence>
60    </xs:complexType>
61    <xs:complexType name="documentType">
62        <xs:sequence>
63          <xs:element name="documentName" type="xs:string"
                minOccurs="0"></xs:element>
64          <xs:element name="documentDescription" type="xs:
                string" minOccurs="0"></xs:element>
65          <xs:element name="documentLinkDescription" minOccurs
                ="0" maxOccurs="unbounded">
66            <xs:complexType>
67              <xs:simpleContent>
68                      <xs:extension base="xs:string">
69                        <xs:anyAttribute namespace="http://www.
                              w3.org/1999/xlink"/>
70                      </xs:extension>
71              </xs:simpleContent>
72            </xs:complexType>
73          </xs:element>
74        </xs:sequence>
75    </xs:complexType>
76    <xs:complexType name="supportingDocumentsType">
77        <xs:sequence>
78          <xs:element name="document" type="documentType"
                maxOccurs="unbounded"></xs:element>
79        </xs:sequence>
80    </xs:complexType>
81    <xs:complexType name="descriptionType">
82      <xs:sequence>
83        <xs:element name="name" type="xs:string" minOccurs
              ="0"></xs:element>
84        <xs:element name="description" type="xs:string"
              minOccurs="0"></xs:element>
85        <xs:element name="linkDescription" minOccurs="0"
              maxOccurs="unbounded">
86            <xs:complexType>
87              <xs:simpleContent>
88                      <xs:extension base="xs:string">
89                        <xs:anyAttribute namespace="http://www.
                              w3.org/1999/xlink"/>
90                      </xs:extension>
91              </xs:simpleContent>
92            </xs:complexType>
93          </xs:element>
```

```
94    </xs:sequence>
95  </xs:complexType>
96  <xs:complexType name="resultsType">
97      <xs:sequence>
98       <xs:element name="resultsLinkDescription" minOccurs
             ="0" maxOccurs="unbounded">
99          <xs:complexType>
100           <xs:simpleContent>
101                   <xs:extension base="xs:string">
102                     <xs:anyAttribute namespace="http://www.
                            w3.org/1999/xlink"/>
103                   </xs:extension>
104             </xs:simpleContent>
105         </xs:complexType>
106       </xs:element>
107     </xs:sequence>
108 </xs:complexType>
109 </xs:schema>
```

Listing B.2: Execution Environment Schema

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:xlink="http://www.w3.org/1999/xlink">
3  <xs:import namespace="http://www.w3.org/1999/xlink"
       schemaLocation="http://www.w3.org/1999/xlink.xsd"/>
4   <xs:complexType name="inputRequirementsType">
5      <xs:sequence>
6        <xs:element name="inputRequirement" type="xs:string"
             maxOccurs="unbounded"></xs:element>
7      </xs:sequence>
8   </xs:complexType>
9    <xs:complexType name="outputRequirementsType">
10     <xs:sequence>
11       <xs:element name="outputRequirement" type="xs:string"
              maxOccurs="unbounded"></xs:element>
12     </xs:sequence>
13   </xs:complexType>
14   <xs:complexType name="libraryType">
15     <xs:sequence>
16       <xs:element name="library" minOccurs="0" maxOccurs="
             unbounded">
17         <xs:complexType>
18           <xs:sequence>
19             <xs:element name="libraryName" type="xs:string
```

```
                    "></xs:element>
20                <xs:element name="libraryLinkDescription"
                      minOccurs="0">
21                  <xs:complexType>
22                    <xs:simpleContent>
23                      <xs:extension base="xs:string">
24                        <xs:anyAttribute namespace="http://www.
                              w3.org/1999/xlink"/>
25                      </xs:extension>
26                    </xs:simpleContent>
27                  </xs:complexType>
28                </xs:element>
29              </xs:sequence>
30            </xs:complexType>
31          </xs:element>
32        </xs:sequence>
33      </xs:complexType>
34      <xs:complexType name="compilersType">
35        <xs:sequence>
36          <xs:element name="compiler" maxOccurs="unbounded">
37            <xs:complexType>
38              <xs:sequence>
39                <xs:element name="compilerName" type="xs:string
                      "></xs:element>
40                <xs:element name="compilerDescription" type="xs
                      :string" minOccurs="0"></xs:element>
41              </xs:sequence>
42            </xs:complexType>
43          </xs:element>
44        </xs:sequence>
45      </xs:complexType>
46      <xs:complexType name="processorsType">
47        <xs:sequence>
48          <xs:element name="processor" maxOccurs="unbounded">
49            <xs:complexType>
50              <xs:sequence>
51                <xs:element name="processorName" type="xs:
                      string"></xs:element>
52                <xs:element name="processorMemory" type="xs:
                      string"></xs:element>
53                <xs:element name="processorDescription" type="
                      xs:string" minOccurs="0"></xs:element>
54              </xs:sequence>
55            </xs:complexType>
56          </xs:element>
```

```
57      </xs:sequence>
58    </xs:complexType>
59    <xs:complexType name="osType">
60      <xs:sequence>
61        <xs:element name="OS" maxOccurs="unbounded">
62          <xs:complexType>
63            <xs:sequence>
64              <xs:element name="osName" type="xs:string"></xs
                    :element>
65              <xs:element name="osVersion" type="xs:string
                    "></xs:element>
66            </xs:sequence>
67          </xs:complexType>
68        </xs:element>
69      </xs:sequence>
70    </xs:complexType>
71    <xs:complexType name="deadlineType">
72      <xs:sequence>
73        <xs:element name="deadlineName" type="xs:string"></xs
              :element>
74        <xs:element name="submissionDeadline" type="xs:
              dateTime"></xs:element>
75      </xs:sequence>
76    </xs:complexType>
77    <xs:complexType name="deadlinesType">
78      <xs:sequence>
79        <xs:element name="deadline" type="deadlineType"
              maxOccurs="unbounded"></xs:element>
80      </xs:sequence>
81    </xs:complexType>
82    <xs:complexType name="ExecutionEnvironmentType">
83      <xs:sequence>
84        <xs:element name="environmentDescription" type="xs:
              string" minOccurs="0" ></xs:element>
85        <xs:element name="libraries" type="libraryType"
              minOccurs="0"></xs:element>
86        <xs:element name="compilers" type="compilersType"
              minOccurs="0"></xs:element>
87        <xs:element name="processors" type="processorsType"
              minOccurs="0"></xs:element>
88        <xs:element name="OSUsed" type="osType" minOccurs
              ="0"></xs:element>
89      </xs:sequence>
90    </xs:complexType>
91    <xs:complexType name="expectedSolutionType">
```

```
92      <xs:sequence>
93        <xs:element name="AllowedForms" type="xs:string"
              maxOccurs="unbounded" minOccurs="0"></xs:element>
94        <xs:element name="inputRequirements" type="
              inputRequirementsType" minOccurs="0"></xs:element>
95        <xs:element name="outputRequirements" type="
              outputRequirementsType" minOccurs="0"></xs:element
              >
96        <xs:element name="executionEnvironment" type="
              ExecutionEnvironmentType" minOccurs="0"></xs:
              element>
97        <xs:element name="deadlines" type="deadlinesType"
              minOccurs="0"></xs:element>
98        <xs:element name="allowedSubmissions" type="xs:string
              " minOccurs="0"></xs:element>
99      </xs:sequence>
100   </xs:complexType>
101 </xs:schema>
```

Listing B.3: Challenge Problem Schema

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:xlink="http://www.w3.org/1999/xlink">
3  <xs:include schemaLocation="
       XmlSchema_Execution_Environment_Challenge_Problem_Version_6
       .xsd"/>
4  <!--Importing Common XMLSchema-->
5  <xs:include schemaLocation="D:\MEng_Docs\MEng_Docs\Project\
       Common_Schema\XmlSchema_Common.xsd"/>
6  <xs:import namespace="http://www.w3.org/1999/xlink"
       schemaLocation="http://www.w3.org/1999/xlink.xsd"/>
7    <!-- Main Schema -->
8    <xs:element name ="challenges">
9      <xs:complexType>
10       <xs:sequence>
11         <xs:element name="challenge" maxOccurs="unbounded">
12           <xs:complexType>
13             <xs:sequence>
14               <xs:element name="challengeName" type="xs:
                     string" />
15               <xs:element name="area" type="xs:string" />
16               <xs:element name="challengeDescription" type
                     ="challengeDescriptionType"></xs:element>
17               <xs:element name="rules" type="ruleType"
```

```
                    minOccurs="0"></xs:element>
18                  <xs:element name="supportingDocuments" type="
                        supportingDocumentsType" minOccurs="0"></
                        xs:element>
19                  <xs:element name="year" type="xs:gYear"
                        minOccurs="0"></xs:element>
20                  <xs:element name="assessmentDescription" type
                        ="assesmentDescriptionType" minOccurs
                        ="0"></xs:element>
21                  <xs:element name="participants" type="
                        participantsType" minOccurs="0"></xs:
                        element>
22                  <xs:element name="benchmarks" type="
                        benchmarksType" minOccurs="0"></xs:element
                        >
23                  <xs:element name="expectedSolution" type="
                        expectedSolutionType"></xs:element>
24                  <xs:element name="allowedTools" type="
                        toolsType" minOccurs="0"></xs:element>
25                  <xs:element name="solutions" type="
                        solutionsType" minOccurs="0"></xs:element>
26                  <xs:element name="contactDetails" type="
                        contactsType" minOccurs="0"></xs:element>
27                  <xs:element name="results" type="resultsType"
                         minOccurs="0"></xs:element>
28                  <xs:element name="changes" type="changesType"
                         minOccurs="0"></xs:element>
29              </xs:sequence>
30            </xs:complexType>
31          </xs:element>
32        </xs:sequence>
33      </xs:complexType>
34    </xs:element>
35  </xs:schema>
```

Listing B.4: Common Schema for Libraries and Tools

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:xlink="http://www.w3.org/1999/xlink">
3  <xs:import namespace="http://www.w3.org/1999/xlink"
       schemaLocation="http://www.w3.org/1999/xlink.xsd"/>
4  <!--Importing Common XMLSchema-->
5  <xs:include schemaLocation="D:\MEng_Docs\MEng_Docs\Project\
       Common_Schema\XmlSchema_Common.xsd"/>
```

```
<xs:complexType name="versionType">
  <xs:sequence>
    <xs:element name="versionName" type="xs:string"
        minOccurs="0"></xs:element>
    <xs:element name="year" type="xs:string" minOccurs
        ="0"></xs:element>
    <xs:element name="description" type="xs:string"
        minOccurs="0"></xs:element>
    <xs:element name="linkDescription" minOccurs="0"
        maxOccurs="unbounded">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:anyAttribute namespace="http://www.
                w3.org/1999/xlink"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="status" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="Current"/>
        <xs:enumeration value="Previous"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
<xs:complexType name="versionsType">
  <xs:sequence>
    <xs:element name="version" type="versionType" maxOccurs
        ="unbounded"></xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name ="exContentsType">
  <xs:sequence>
    <xs:element name="contentDescription" type="xs:string"
        minOccurs="0"></xs:element>
    <xs:element name="content" type="descriptionType"
        maxOccurs="unbounded"></xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="contentFilesType">
  <xs:sequence>
```

```
43      <xs:element name="contentDescription" type="xs:string"
            minOccurs="0"></xs:element>
44      <xs:element name="contentFile" type="descriptionType"
            maxOccurs="unbounded"></xs:element>
45    </xs:sequence>
46  </xs:complexType>
47  </xs:schema>
```

Listing B.5: Course Module Schema

```
1   <?xml version="1.0" encoding="iso-8859-1"?>
2   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xlink="http://www.w3.org/1999/xlink">
3   <xs:import namespace="http://www.w3.org/1999/xlink"
        schemaLocation="http://www.w3.org/1999/xlink.xsd"/>
4    <!--Importing Common XMLSchema-->
5   <xs:include schemaLocation="D:\MEng_Docs\MEng_Docs\Project\
        Common_Schema\XmlSchema_Common.xsd"/>
6   <xs:element name="modules">
7     <xs:complexType>
8       <xs:sequence>
9         <xs:element name="module" maxOccurs="unbounded">
10          <xs:complexType>
11            <xs:sequence>
12              <xs:element name="school" type="xs:string"
                    minOccurs="0"></xs:element>
13              <xs:element name="moduleName" type="xs:string
                    "></xs:element>
14              <xs:element name="moduleCode" type="xs:string
                    "></xs:element>
15              <xs:element name="year" type="xs:string"
                    minOccurs="0"></xs:element>
16              <xs:element name="professors" type="
                    contactsType" minOccurs="0"></xs:element>
17              <xs:element name="status" type="xs:string"
                    minOccurs="0"></xs:element>
18        <xs:element name="requiredFor" type="xs:string"
              minOccurs="0"></xs:element>
19        <xs:element name="allowedTracks" type="
              allowedTracksType" minOccurs="0"></xs:element>
20              <xs:element name="numberOfCredits" type="xs:
                    string" minOccurs="0"></xs:element>
21        <xs:element name="teachingTermsAllowed" type="
              termsType" minOccurs="0"></xs:element>
22        <xs:element name="preRequisites" type="
```

```
            preRequisitesType" minOccurs="0"></xs:element>
23        <xs:element name="courseDescription" type="
            descriptionType" minOccurs="0"></xs:element>
24            <xs:element name="aims" type="aimsType"
                minOccurs="0"></xs:element>
25            <xs:element name="learningOutcomes" type="
                learningOutcomesType" minOccurs="0"></xs:
                element>
26            <xs:element name="rules" type="ruleType"
                minOccurs="0"></xs:element>
27            <xs:element name="workload" type="workloadType"
                 minOccurs="0"></xs:element>
28            <xs:element name="feedback" type="feedbackType"
                 minOccurs="0"></xs:element>
29            <xs:element name="content" type="contentType"
                minOccurs="0"></xs:element>
30            <xs:element name="location" type="xs:string"
                minOccurs="0"></xs:element>
31            <xs:element name="results" type="resultsType"
                minOccurs="0"></xs:element>
32            <xs:element name="requiredLinks" type="
                relatedLinksType" minOccurs="0"></xs:element
                >
33          </xs:sequence>
34        </xs:complexType>
35      </xs:element>
36    </xs:sequence>
37  </xs:complexType>
38 </xs:element>
39 </xs:schema>
```

Listing B.6: Schema for Libraries

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xlink="http://www.w3.org/1999/xlink">
3 <xs:import namespace="http://www.w3.org/1999/xlink"
    schemaLocation="http://www.w3.org/1999/xlink.xsd"/>
4 <!--Importing Common XMLSchema-->
5 <xs:include schemaLocation="D:\MEng_Docs\MEng_Docs\Project\
    Common_Schema\XmlSchema_Common.xsd"/>
6 <xs:include schemaLocation="D:\MEng_Docs\MEng_Docs\Project\
    XmlSchema_Libraries\XMLSchema_Common_Libraries_Tools.xsd
    "/>
7 <xs:element name="libraries">
```

```
8    <xs:complexType>
9      <xs:sequence>
10       <xs:element name="library" maxOccurs="unbounded">
11         <xs:complexType>
12           <xs:sequence>
13             <xs:element name="libraryName" type="xs:string
                   "></xs:element>
14             <xs:element name="libraryOverview" type="
                   descriptionType" minOccurs="0"></xs:element>
15             <xs:element name="availableVersions" type="
                   versionsType" minOccurs="0"></xs:element>
16             <xs:element name="
                   experimentalContentsOrExtensions" type="
                   exContentsType" minOccurs="0"></xs:element>
17             <xs:element name="contributors" type="
                   contactsType" minOccurs="0"></xs:element>
18             <xs:element name="contentFiles" type="
                   contentFilesType" minOccurs="0"></xs:element
                   >
19             <xs:element name="downloads" type="
                   downloadsType" minOccurs="0"></xs:element>
20             <xs:element name="requiredToolsForCompilation"
                   type="toolsType" minOccurs="0"></xs:element>
21             <xs:element name="supportingDocuments" type="
                   supportingDocumentsType" minOccurs="0"></xs:
                   element>
22             <xs:element name="dependencyDetails" type="
                   dependencyDetailsType" minOccurs="0"></xs:
                   element>
23             <xs:element name="examples" type="examplesType"
                    minOccurs="0"></xs:element>
24             <xs:element name="relatedLinks" type="
                   relatedLinksType" minOccurs="0"></xs:element
                   >
25             <xs:element name="references" type="
                   referencesType" minOccurs="0"></xs:element>
26           </xs:sequence>
27         </xs:complexType>
28       </xs:element>
29     </xs:sequence>
30   </xs:complexType>
31 </xs:element>
32 </xs:schema>
```

Listing B.7: Schema for Tools

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:xlink="http://www.w3.org/1999/xlink">
3  <!--Importing Common XMLSchema-->
4  <xs:include schemaLocation="D:\MEng_Docs\MEng_Docs\Project\
       Common_Schema\XmlSchema_Common.xsd"/>
5  <xs:include schemaLocation="D:\MEng_Docs\MEng_Docs\Project\
       XmlSchema_Libraries\XMLSchema_Common_Libraries_Tools.xsd
       "/>
6  <xs:import namespace="http://www.w3.org/1999/xlink"
       schemaLocation="http://www.w3.org/1999/xlink.xsd"/>
7  <xs:element name="verificationTools">
8    <xs:complexType>
9      <xs:sequence>
10       <xs:element name="verificationTool"  maxOccurs="
             unbounded">
11         <xs:complexType>
12           <xs:sequence>
13             <xs:element name="toolName" type="xs:string"></
                   xs:element>
14             <xs:element name="toolOverview" type="
                   descriptionType" minOccurs="0"></xs:element>
15             <xs:element name="availableVersions" type="
                   versionsType" minOccurs="0"></xs:element>
16             <xs:element name="Functionalities" type="
                   functionalitiesType" minOccurs="0"></xs:
                   element>
17             <xs:element name="intendedUsers" type="
                   intendedUsersType" minOccurs="0"></xs:
                   element>
18             <xs:element name="
                   experimentalContentsOrExtensions" type="
                   exContentsType" minOccurs="0"></xs:element>
19             <xs:element name="contacts" type="contactsType"
                    minOccurs="0"></xs:element>
20             <xs:element name="contentFiles" type="
                   contentFilesType" minOccurs="0"></xs:element
                   >
21             <xs:element name="supportingDocuments" type="
                   supportingDocumentsType" minOccurs="0"></xs:
                   element>
22             <xs:element name="relatedLinks" type="
                   relatedLinksType" minOccurs="0"></xs:element
```

```
                    >
23                  <xs:element name="relatedTools" type="toolsType
                        " minOccurs="0"></xs:element>
24                  <xs:element name="howToObtain" type="
                        howToObtainType" minOccurs="0"></xs:element>
25                  <xs:element name="executionEnvironment" type="
                        executionEnvironmentType" minOccurs="0"></xs
                        :element>
26                </xs:sequence>
27              </xs:complexType>
28            </xs:element>
29          </xs:sequence>
30        </xs:complexType>
31  </xs:element>
32  </xs:schema>
```

Bibliography

[Ale13] Alea: a library for reasoning on randomized algorithms in Coq . https://www.lri.fr/~paulin/ALEA/, 2013.

[AW10] Mark Lawford Alan Wassyng, Tom Maibaum. On software certification: We need product-focused approaches. Number LNCS 6028. Springer-Verlag Berlin Heidelberg, 2010. C. Choppy and O. Sokolsky (Eds.): Monterey Workshop 2008.

[BCa] Legion of the Bouncy Castle Java cryptography APIs . http://www.bouncycastle.org/java.html.

[Ben11] Marc Bender. What is software certification? McMaster Centre for Software Certification (McSCert), April 13 2011. McSCert Seminar 2.

[Bri13a] Domenico Briganti. W3C XML Schema (XSD) Validation online. http://www.utilities-online.info/xsdvalidation/#.U0M3I_ldXwg, 2013.

[Bri13b] Domenico Briganti. XSLT (eXtensible Stylesheet Language Transformations) online transformations. http://www.utilities-online.info/xsltransformation/#.U0M6RfldXwg, 2013.

[BSh] Legion of the Bouncy Castle C sharp cryptography APIs . http://www.bouncycastle.org/csharp/.

[CAD] Construction and Analysis of Distributed Processes. http://cadp.inria.fr/. Toolbox for the design of asynchronous concurrent systems.

[CAD12] CADE - the Conference on Automated Deduction. http://www.cadeinc.org/, 2012.

[CAD13] The CADE ATP System Competition. http://www.cs.miami.edu/~tptp/CASC/, 2013. The World Championship for Automated Theorem Proving.

[Cla13] Clafer Notation. http://gsd.uwaterloo.ca/clafer, 2013.

[COQ] The Coq Proof Assistant. http://coq.inria.fr/.

[CW94] Per Runeson Claes Wohlin. Certification of software components. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 20, June 1994.

[DF05] Ewen Denney and Bernd Fischer. Software certification and software certificate management systems. In *2005 Automated Software Engineering Workshop on Software Certificate Management*, Long Beach, California, USA, November 8 2005. Association for Computing Machinery.

[DFC] A Coq Library on Floating-Point Arithmetic . http://lipforge.ens-lyon.fr/www/pff/.

[Gli] The GNU C Library (glibc). https://www.gnu.org/software/libc/index.html.

[IJC12] IJCAR 2012 The 6th International Joint Conference on Automated Reasoning. http://ijcar.cs.manchester.ac.uk/, 2012.

[MaC13] MECHTRON/SFWR ENG 4AA4/6GA3 Real-Time Systems , 2013. McMaster University.

[McC13] McMaster Centre for Software Certification. https://www.mcscert.ca/, 2013.

[MCH] Microchip Certified Class B Safety Software Library for 16 bit and PIC32 MCUs . http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&dDocName=en548988.

[NUS] NuSMV: a new symbolic model checker. http://nusmv.fbk.eu/.

[Pac07] Pacemaker Formal Methods Challenge. http://sqrl.mcmaster.ca/pacemaker.htm, 2007.

[SAT12a] Fifteenth International Conference on Theory and Applications of Satisfiability Testing (SAT 2012). http://sat2012.fbk.eu/, 2012.

[SAT12b] SAT Challenge 2012. http://baldur.iti.kit.edu/SAT-Challenge-2012/, 2012.

[SMT12] SMT-COMP 2012. http://smtcomp.sourceforge.net/2012/, 2012.

[SSL] OpenSSL. http://www.openssl.org/.

[STL] Introduction to the Standard Template Library. https://www.sgi.com/tech/stl/stl_introduction.html.

[SV13] Competition on Software Verification (SV-COMP). http://sv-comp.sosy-lab.org/2013/index.php, 2013.

[TAC13] TACAS 2013. http://www.etaps.org/index.php/2013/tacas, 2013.

[Vic] Embedded and Networked Systems . http://www.vu.edu.au/units/ENE3202. Victoria University.

[Voa14] Jeffrey Voas. Developing a usage-based software certification process. Reliable Software Technologies, 03 2014.

[Wat14] ECE423: Embedded Computer Systems , 2014. University of Waterloo.

[XML10] XML Database. http://www.rpbourret.com/xml/XMLDatabaseProds.htm, 2010.

[XPa99] XML Path Language (XPath)Version 1.0. http://www.w3.org/TR/xpath/, 1999.

[XQu01] W3C XML Query (XQuery). http://www.w3.org/XML/Query/, 2001.

[XSD11] XML Schema. http://www.w3.org/XML/Schema/, 2011.

[Yor13] Course Modules Offered at York University . https://www.cs.york.ac.uk/postgraduate/taught-courses/msc-scse/#tab-2, 2013. MSc in Safety Critical Systems Engineering.