Structured Repository of Information Related to Software Certification

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

May 9, 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 same data. For human understandability of the results, we also created some XSL to display the data, guided by the schema.

Contents

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 process follow various mechanisms.

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, 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 out 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 of 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 in order to index some available material in the respective areas. These areas were already chosen by the principals running the Software Certification project at McMaster. The latter chapters would introduce the idea of the software repository meant to store available material, schema designing and testing processes for Challenge problems, Course Modules and Certified Software Libraries finally concluding with future scope.

Chapter 2

Global Requirements

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. were selected as reference problems for challenge problem area. This completes the data gathering process for each area. However, the gathered data needs to be stored in a uniform structural format which makes it searchable and reusable. Thus, a schema one for each chosen area should be designed to classify the material related to the corresponding area.

In order to accomplish the schema designing task, the meta-data associated with all the reference material in each area is identified. The next task is to select attributes which can be organized in the schematic format using the meta-data. This is done by finding the commonalities between the meta-data associated with each area. This is the final step in the requirement gathering process. The attributes for each area are then converted into the corresponding schemas.

4

Chapter 3

Overview of Software Repository

The design of the repository is provided by the principals running the Software Certification project at McMaster. The central structure of the repository is shown in 3.1.
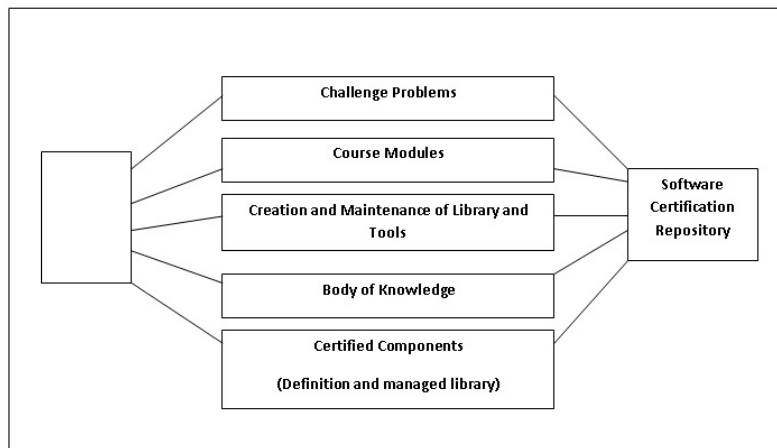

Figure 3.1: Design of Software Repository

The report considers first three components of this repository namely Challenge Problems, Course Modules and Creation and Maintenance of Library and Tools. 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 SMT COMP challenge, CADE ATP challenge, Pacemaker challenge etc. Most of the challenge problems are offered every year as a part of various conferences and workshops held for software communities. The goal is to design a schema which 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. Course modules intend to have all the information about the courses involving topics in software certification. The schema mainly targets all the time independent information like aim of the course, learning outcomes, course outlines, materials etc. The last component manages information about libraries and verification tools. Libraries are of two types. It can either be a library which is a part of a verification tool or it can be a library that is verified by a verification tool. For example, Coq is a tool used for verification of libraries and contains number of libraries. So, many libraries can be a part of a tool and can be verified by a tool. Libraries and tools are stored using different schemas.

Chapter 4

Methodology

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 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. Therefore, 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 [Bria]. 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 [Brib]. 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 meaning that 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, CADE ATP challenge, Pacemaker challenge, SMT COMP challenge, SV-COMP challenge [SAT, CADb, Pac07, SMT, SV13], etc. have been offered as a part of various conferences and workshops. Each challenge has different dimensions.

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.

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 various the 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 etc. is 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 etc. 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. The next attribute denotes the area of the challenge (SAT Solving, Automated Deduction, Object Oriented Systems etc.). 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 etc.), 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. 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. Specific information related

to benchmarks is can be 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.

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 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, it does not even impose the restriction on number of phones, emails or web page links and allows collecting any number of contacts. Similar restrictions are imposed on all other tags such as rules, supporting documents, assessment details, benchmarks, execution environment etc. This allows users to store maximum 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 [Bria].

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 code for the common schema.

Listing 5.1: 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>
```

```xml
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>
```

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 code for Execution Environment Schema.

Listing 5.2: 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>
```

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.

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

Listing 5.3: 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>
```

18

```
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>
```

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 [Brib]. 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/Schema` 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 [Yor, Wat, MaC, Vic]. 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 in order 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, one course module offered at Victoria University, Australia and one at McMaster University, Canada.

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. 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.

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, pre-requisites 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 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 [Bria]. The testing details are discussed later in the report. The Listing 6.1 shows the code for course module schema.

The complete code is found out at `https://github.com/ish1289/MEng_`

Listing 6.1: 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:
```

23

```
                    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>
```

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 [Brib]. As discussed earlier, the tool ac-
cepts 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.

Schema Design

The design process again begins with some sample libraries and tools [Ale13, DFC, MCH, BCa, BSh, Gli, COQ, CADa, STL, NUS, SMP]. 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.

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, STL etc. 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.

Attributes Listing: Tools

After analyzing some tools like COQ, CADP, NuSMv, etc [COQ, CADa, 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 at-

tributes 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.

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.

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 common schema code for libraries and tools.

Listing 7.1: 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"/>
6  <xs:complexType name="versionType">
7    <xs:sequence>
8      <xs:element name="versionName" type="xs:string"
           minOccurs="0"></xs:element>
9      <xs:element name="year" type="xs:string" minOccurs
           ="0"></xs:element>
```

```xml
10      <xs:element name="description" type="xs:string"
            minOccurs="0"></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:attribute name="status" use="optional">
22       <xs:simpleType>
23         <xs:restriction base="xs:string">
24           <xs:enumeration value="Current"/>
25           <xs:enumeration value="Previous"/>
26         </xs:restriction>
27       </xs:simpleType>
28     </xs:attribute>
29   </xs:complexType>
30   <xs:complexType name="versionsType">
31     <xs:sequence>
32       <xs:element name="version" type="versionType" maxOccurs
            ="unbounded"></xs:element>
33     </xs:sequence>
34   </xs:complexType>
35   <xs:complexType name ="exContentsType">
36     <xs:sequence>
37       <xs:element name="contentDescription" type="xs:string"
            minOccurs="0"></xs:element>
38       <xs:element name="content" type="descriptionType"
            maxOccurs="unbounded"></xs:element>
39     </xs:sequence>
40   </xs:complexType>
41   <xs:complexType name="contentFilesType">
42     <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>
```

Schema for Libraries

The Listing 7.2 contains the implementation for libraries' schema.

Listing 7.2: 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>
```

Schema for Tools

The Listing 7.3 gives the implementation for tools' schema.

Listing 7.3: 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   <!−−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>
```

```xml
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>
```

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.

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 are

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

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 were failed. For example, the first test case "information coverage" failed because, the detailed information about the rule wasn't stored by the schema. The schema could store only the high level description of the rules. The second and third test cases were 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, 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 ia 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.

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 reasonable number of XMLs in order 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 5.3 [SAT]. The XML is then validated using the online tool against the challenge problem schema [Bria].

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 viz. challenge problems, course modules and libraries and tools is proposed here.

Future tasks for the repository can be categorized into two streams. First one 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. Hence, the next job is to find some more relevant challenge problems, course modules, libraries and tools. The second part of the future task 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.

Finally, to conclude, the report provided 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

```
1   A.    Challenge Name
2   B.    Area
3   C.    Challenge Description
4     a.    Description
5     b.    Challenge Date
6        1.    To
7        2.    From
8     c.    Challenge Location
9     d.    Associated Conference
10    e.    Part of Series
11  D.    Rules
12    a.    Rule
13       1.    Rule Category
14       2.    Description
15       3.    Input Requirements
16          i.    Input Requirement
17       4.    Output Requirements
18          i.    Output Requirement
19       5.    Links
20  E.     Supporting Documents
21    a.    Document Name
22    b.    Description
23    c.    Link
24  F.    Year
25  G.    Assessment Description
26    a.    Description
27    b.    Link
28    c.    Jury
```

```
74      1.   Deadline
75        A.   Name
76        B.   Date
77   f.   Allowed Submissions
78 K.   Allowed Tools
79   a.   Tool
80      1.   Name
81      2.   Description
82      3.   Link
83 L.   Required Contacts
84   a.   Contact
85      1.   Name
86      2.   Description
87      3.   Phone
88      4.   Email
89      5.   Web page Link
90 M.   Results
91   a.   Link
92 N.   Changes from Previous Challenges
93   a.   Change
94      1.   Description
95      2.   Link
96 O.   Solutions
97   a.   Format
98   b.   Description
99   c.   Input Requirements
100   d.   Output Requirements
101   e.   Execution Environment
```

Listing A.2: Attributes for Course Modules Schema

```
1  Modules
2  a.   Module
3     1.    School/University
4     2.    Module Name
5     3.    Module Code
6     4.    Professors/Lecturers
7     5.    Status (Core/Optional)
8     6.    Required For (Full−time/part−time)
9     7.    Allowed Tracks
10    8.    Number of credits
11    9.    Teaching Term
12    10. Pre−requisites
13    11. Description
14    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      iv.  Deadlines
45      v.   Submissions
46        a.   Description
47        b.   Link
48        c.   Start Date
49        d.   End Date
50      vii.   Assessment
51        a.   Description
52        b.   Assignments
53          1.   Assignment
54            a.   Description
55            b.   weight
56            c.   Link
57        c.   Exams
58          2.   Exam
59            d.   Description
```

```
60          e.   weight
61          f.   Link
62       d.   Start Date
63       e.   End Date
64    viii.  Allowed Tools
65       a.   Tool Name
66       b.   Description
67       c.   Link
68    ix.  Location
69    x.   Results
70    18. Related Links
```

Listing A.3: Attributes for Libraries' Schema

```
1   A.   Libraries
2    i.   Library
3    1.   Name
4    2.   Overview
5       a.   Description
6       b.   Link
7    3.   Current Version
8    4.   Versions (Current/Previous descriptions)
9       a.   Version name
10      b.   Status (current/previous)
11      c.   Description
12      d.   Link
13   5.   Experimental Library Contents (Similar to Extensions)
14      a.   content
15        i.   Description
16        ii.  Link
17   6.   Contributors
18      a.   Contributor
19        i.   Name
20        ii.  Contribution Description
21        iii.   Email
22        iv.  Phone
23        v.   Web page link
24   7.    Content Files
25      a.   Content File
26        i.   Name
27        ii.  Description
28        iii.   Link
29   8.   Downloads
30      a.   Description
31      b.   Format (OS)
```

```
32    c.   Size
33    d.   Link
34   9.   Required Tools for compilation
35    a.   Tool
36      i.   Tool name
37      ii.  Description
38      iii.  Link
39   10.  Documentation
40    a.   Document
41      i.   Name
42      ii.  Description
43      iii.  Link
44   11.  Dependency details
45    a.   Description
46    b.   Link
47   12.  Examples
48    a.   Example
49      i.   Name
50      ii.  Description
51      iii.  Link
52   13.  Related Links
53    a.   Link
54   14.  References
55    a.   Description
56    b.   Link
```

Listing A.4: Attributes for Tools' Schema

```
1    Tools
2    o Tool
3    1.   Name
4    2.   Overview
5     a.   Description
6     b.   Link
7    3.   Current Status
8     a.   Latest available version
9     b.   Description
10    c.   Link
11   4.   Functionalities/Features
12    a.   Function/Feature
13      i.   Description
14      ii.  Link
15   5.   Intended Users
16    a.   User
17   6.   Extensions
```

```
63          I.   Name
64          II.  Version
65          III.   Distribution
66    11. Related Tools
67      a.   Tool
68        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
```

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.

[Bria] Domenico Briganti. W3C XML Schema (XSD) Validation online. `http://www.utilities-online.info/xsdvalidation/#.UOM3I_ldXwg`.

[Brib] Domenico Briganti. XSLT (eXtensible Stylesheet Language Transformations) online transformations. `http://www.utilities-online.info/xsltransformation/#.UOM6RfldXwg`.

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

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

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

[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`.

[MaC] MECHTRON/SFWR ENG 4AA4/6GA3   Real-Time Systems . `http://www.cas.mcmaster.ca/~downd/courses/4aa4/4aaout.htm`. McMaster University.

[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.

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

[SMP] SMPS Control Library. `http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&dDocName=en567449`.

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

[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.

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

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

[Wat]   ECE423:   Embedded   Computer   Systems .   `https://ece.`
        `uwaterloo.ca/~rpellizz/ECE423.php`. University of Waterloo.

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