Schema Designing to Index Some Available Material Related to Software Certification

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

April 20, 2014

Abstract

Software Certification is the process of certifying the systems scoped to their software part. The report emphasizes on the task of indexing some available material around the various areas of software certification and thus explains the idea of the software repository. However, it revolves around Challenge Problems, Course Modules and Certified Software Libraries and Tools which are the three areas related to Software Certification and also are the three components of the repository. The scope of the report is to plan and implement schemas using XSDs to index some important information for the above three areas. The report starts with an introduction to the software certification and some approaches followed to perform software certification. It then describes the approach of the software repository and the need for schema design to store the data in the repository. The schemas are implemented using XSDs. Thus, the latter chapters explain the process of schema design and present implementations developed by the corresponding XSLs for challenge problems, course modules, libraries and tools respectively. Further part provides an overview of the testing process and a representative case study of challenge problem category.

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 encompassesthewide range of formal, semi-formal and informal assurance techniques which includeeven formal verification of securitypolicies, 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. The latter chapters in the report would introduce the idea of the software repository meant to store available materials, schema designing and testing processes for Challenge problems, Course Modules and Certified Software Libraries finally concluding with future scope.

Chapter 2

Golbal Requirements

As mentioned earlier, the main idea behind this report is to create a software repository which stores material from various areas around software certification in a schematic format. The first requirement comprises of choosing the areas to be included in the software repository and method of schema designing process. The areas considered here are chosen based upon the intended use of the repository. As the repository would be mainly designed for the university domain, the areas such as Challenge problems, Course Modules, etc. which are relevant to the university topics are selected. The structure of the repository is explained in detail later.

Next requirements are to decide the format of the schema, data gathering and schema creating process. Most of the material to be stored is loosely bound making it easy to store material in XML format. So, instead of using relational schema designing, Xml schema designing is used. The data gathering process includes finding some data samples which would help to design the schema as well as to test the schema afterwards. These requirements are applicable to every area that has been included in the repository. The next chapter explains the schema designing method followed in the report.

Chapter 3

Overview of Software Repository

The prime components of the repository are Challenge Problems, Course Modules, Creation and Maintenance of Libraries, Body of Knowledge and Certified Components which include definitions and managed libraries. The central structure of the repository is shown in 3.1.

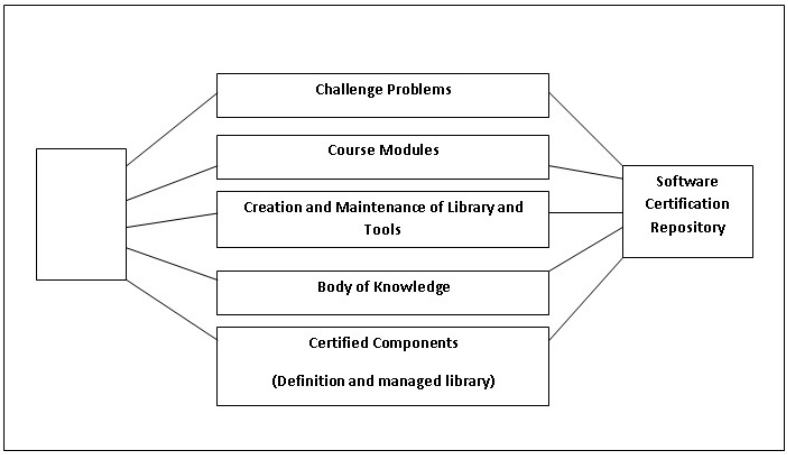

Figure 3.1: Design of Software Repository

This 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. XMLs added to the repository would contain all the information regarding a particular challenge and would be created according to the general schema designed for challenge problems. Course modules intend to have all the information about the courses involving topics in software certification. This information is again in XML format following schema designed for Course Module part of Software Repository. The last component manages information about libraries and verification tools. Libraries comprise 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. XMLs representing libraries and tools are validated against two separate schemas for libraries and tools respectively.

3.0.1 Methodology

The repository would have a schema design for each of the components of the repository. The schema would be created as XSD 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. 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 4

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 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 part of various conferences and workshops. Each challenge has different dimensions.

The requirements and specifications depend on the actual challenge problem and their organization meaning 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.

4.1 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 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. The Listing 4.1 shows all the attributes derived after analyzing some challenge problems.

Listing 4.1: 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

```
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
```

Attributes' names indicate the purpose of the attribute. Thus, it covers all the required parameters regarding the challenge such as its name, area, its description, associated conference, rules to solve the challenge, documentation, tools allowed, expected solution format, available solutions, assessment details, deadlines and contacts. Some challenge problems also have a set of benchmark problems. Benchmark problems are smaller sets of problems related to main challenge problem. Their formats, rules and solutions vary from challenge to challenge. Some attributes are used to tag this relevant information about benchmarks, as well. 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.

4.2 Implementation

The above schema design is implemented using XSD. 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].

4.2.1 Common Schema

As described earlier, this schema represents all the common attributes to the three areas covered in this report. This is done to avoid redundancy in implementing the entire schema. This schema would be included in all the main schemas for challenge problems, course modules, libraries and tools. The Listing 4.2 displays the code for the common schema.

Listing 4.2: 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>
```

```
<xs:complexType name="toolsType">
    <xs:sequence>
      <xs:element name="tool" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="toolName" type="xs:string"></xs:element>
            <xs:element name="toolDescription" type="xs:string" minOccurs="0"></xs:element>
            <xs:element name="toolLinkDescription" 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:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
<xs:complexType name="relatedLinksType">
    <xs:sequence>
     <xs:element name="relatedLinkDescription" 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:complexType>
<xs:complexType name="documentType">
    <xs:sequence>
      <xs:element name="documentName" type="xs:string" minOccurs="0"></xs:element>
      <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>
```

4.2.2 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 4.3 shows the code for Execution Environment Schema.

Listing 4.3: 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>
```

4.2.3 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 4.4 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 4.4: 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>
```

4.3 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 [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 5

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 providing a schematic structure in order to be able to tag all the information related to various courses. Some examples of program 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.

5.1 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.

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 5.1 shows all the attributes used to implement course module schema.

Listing 5.1: Attributes for Course Modules Schema

```
1 Modules
2 a. Module
3   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
    a. Aim
        i. Description
        ii. Link
13. Learning Outcomes
14. Rules
15. Workload
    i. Total lecture hours
    ii. Each lecture time
    iii. Total Private study time
    iv. Assessment Time
16. Feedback
    i. Description
    ii. Link
17. Content
    i. Topics covered
        a. Description
        b. Link
    ii. Teaching Material
        a. Name
        b. description
        c. Type (Slides/Case Studies/exercise)
        d. Link
    iii. Books
        a. Name
        b. Type (Required/Recommended)
        c. Author
        d. Title
        e. Publisher
        f. Year
    iv. Deadlines
    v. Submissions
        a. Description
        b. Link
        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
```

5.2 Implementation

The schema implementation follows the schema discussed in an above section. It also includes the common schema implemented given in Listing 4.2. This schema again covers the general structure to tag all the attributes of course modules in diverse areas while allowing to save 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 5.2 shows the code for course module schema.

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

Listing 5.2: 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>
```

5.3 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 [Brib]. 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 6

Libraries and Tools

The third component of the repository saves the data related to libraries and tools. The repository requires a schema 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. Tools are referred as a certification mechanism for libraries. The report presents two separate schemas for libraries and tools respectively.

6.1 Schema Design

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

6.1.1 Attributes Listing: Libraries

The analysis of some libraries such as Microchip certified libraries, COQ libraries and a PRL math library, some attributes regarding the libraries are noted down in a hierarchical format. The Listing 6.1 shows the elements

for the libraries' schema.

Listing 6.1: 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
```

```
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
```

6.1.2 Attributes Listing: Tools

After analyzing some tools like COQ, CADP, NuSMv, etc [COQ, CADa]., some elements that constitute the tool's schema are found out.The Listing 6.2 shows the elements for the tools' schema.

Listing 6.2: 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
18      a. Extension
19          i. Description
20          ii. Link
21  7. Contacts
22      a. Contact
```

23 i. Name
24 ii. Email
25 iii. Phone
26 iv. Web page link
27 8. Contents
28 a. Content
29 i. Name
30 ii. Description
31 iii. Link
32 9. How ToObtain
33 a. Source
34 i. Size
35 ii. Description
36 iii. Link
37 b. Binaries
38 i. Binary
39 I. Description
40 II. Format (OS)
41 III. Size
42 IV. Link
43 c. Others
44 i. Other forms
45 I. Description
46 II. Link
47 10. Execution Environment
48 a. Languages Used
49 i. Language
50 b. Input Requirements
51 i. Input Requirement
52 c. Compatible Compilers
53 i. Compiler
54 I. Name
55 II. Description
56 d. Compatible Processors
57 i. Processor
58 I. Name
59 II. Memory
60 III. Description
61 e. Compatible OS
62 i. OS
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
```

6.2 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 4.2. The latter chapter talks about the testing of these schemas in detail.

6.2.1 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 6.3 shows the common schema code for libraries and tools.

Listing 6.3: 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>
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>
```

6.2.2 Schema for Libraries

The Listing 6.4 contains the implementation for libraries' schema.

Listing 6.4: Schema for Libraries

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xlink="http://www.w3.org/1999/xlink">
<xs:import namespace="http://www.w3.org/1999/xlink"
    schemaLocation="http://www.w3.org/1999/xlink.xsd"/>
<!--Importing Common XMLSchema-->
<xs:include schemaLocation="D:\MEng_Docs\MEng_Docs\Project\
    Common_Schema\XmlSchema_Common.xsd"/>
<xs:include schemaLocation="D:\MEng_Docs\MEng_Docs\Project\
    XmlSchema_Libraries\XMLSchema_Common_Libraries_Tools.xsd
    "/>
<xs:element name="libraries">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="library" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="libraryName" type="xs:string
                "></xs:element>
            <xs:element name="libraryOverview" type="
                descriptionType" minOccurs="0"></xs:element>
            <xs:element name="availableVersions" type="
                versionsType" minOccurs="0"></xs:element>
            <xs:element name="
                experimentalContentsOrExtensions" type="
                exContentsType" minOccurs="0"></xs:element>
            <xs:element name="contributors" type="
                contactsType" minOccurs="0"></xs:element>
            <xs:element name="contentFiles" type="
                contentFilesType" minOccurs="0"></xs:element
                >
            <xs:element name="downloads" type="
                downloadsType" minOccurs="0"></xs:element>
            <xs:element name="requiredToolsForCompilation"
                type="toolsType" minOccurs="0"></xs:element>
            <xs:element name="supportingDocuments" type="
                supportingDocumentsType" minOccurs="0"></xs:
                element>
            <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>
```

6.2.3 Schema for Tools

The Listing 6.5 gives the implementation for tools' schema.

Listing 6.5: 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>
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`

6.3 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 7

Testing

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

7.1 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 7.1 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 7.1. This again follows the iterative approach till all the test cases are satisfied.

The table 7.1 lists down the test cases used for both XSDs and XSLs of challenge problems, course modules, libraries and tools.

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 7.1: Test Cases

7.2 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 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 4.4 [SAT]. The XML is then validated using the online tool against the challenge problem schema [Bria].

Chapter 8

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 report proposed the overall 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.

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 areas such as challenge problems, course modules, libraries and tools. Hence, the next job for the three areas in the report 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 three areas around software certification.

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/#.U0M3I_ldXwg.

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

[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.

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

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

[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 process. 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.