Clean Sweep Robotic Vacuum Cleaner

Control System & Sensor Simulator

SE-359/459 Team Project

STUDENT GUIDE

School of Computing
College of Computing and Digital Media
DePaul University
Chicago, IL 60604, USA

Fall 2014-2015

1 Control System

Your company has assembled a small team of developers and business people to work on the implementation of the Clean Sweep robotic vacuum cleaner control system. The control system integrates the Clean Sweep hardware components so that the vacuum can successfully navigate and clean a typical home.

1.1 Navigation

The Clean Sweep roams around the home sweeping up any dirt that it finds. Therefore it must be able to discover the basic floor plan of a home based on the input from its sensors. The Clean Sweep has four sensors around its perimeter that detect when it is about to encounter an obstacle. The control system must interpret the output from these sensors and stop the vacuum from running into any obstacles. Upon encountering an obstacle, the Clean Sweep must determine a new direction that prevents it from running into another obstacle.

Another goal is to minimize the amount of work done by the Clean Sweep. This means that while a random walk around the home would work, it is better if the machine had some intelligence.

Under no circumstances should the Clean Sweep attempt to traverse stairs. A sensor on the bottom of the Clean Sweep can detect the edge of stairs or other declines. If this sensor detects anything, the control system should treat it as though it had encountered any other kind of obstacle.

The Clean Sweep can move in four different directions. If you envision the Clean Sweep as traversing a grid, and assume that it is located at position (x, y), then it has the ability to move to positions: $(x+1, y)$, $(x-1, y)$, $(x, y+1)$, and $(x, y-1)$. There is no need for the vacuum to change its orientation or turn around in order to navigate.

1.2 Dirt Detection and Cleaning

The Clean Sweep has a sophisticated dirt detection sensor. As the vacuum moves, the sensor determines if there is dirt to collect. If so, the control system should engage the vacuum and sweep up the dirt. The sensor is not able to tell how much dirt is present, only whether or not the current location is considered "clean". Thus multiple uses of the vacuum may be required to clean a particularly dirty area.

The Clean Sweep can detect whether or not the surface it is currently traversing is bare floor, low-pile carpet, or high-pile carpet. It should automatically shift its cleaning apparatus between the surfaces accordingly.

Each use of the vacuum removes 1 unit of dirt. The Clean Sweep has a dirt-carrying capacity of 50 units. Once that capacity has been reached, it should return to its charging base and turn on its *Empty Me* indicator.

The Clean Sweep doesn't stop cleaning until:

- It has visited every accessible location on the current floor.

- It's 50-unit dirt capacity has been met.
- It only has enough power remaining to return to its charging station. See *Power Management* for more details.

Each time the Clean Sweep begins a cleaning cycle, it automatically assumes that each location is dirty. A cleaning cycle is defined as the start of the next cleaning after all reachable areas have been cleaned. So, if the Clean Sweep is partially done cleaning a floor and has returned to its base station to recharge, it will assume that the current cleaning cycle is not yet complete.

1.3 Power Management

The Clean Sweep has a limited battery life of 50 units of charge. Each movement and vacuum operation requires units of charge depending on the surfaces being traversed:

- Bare floor - 1 unit
- Low-pile carpet - 2 units
- High-pile carpet - 3 units

The charge required to clean the current location depends solely on the surface of that location. The charge required for the Clean Sweep to move from location A to location B is the average of the required charge costs for the surfaces at the two locations. For example, if the Clean Sweep moves from a bare floor to low-pile carpet, then the charge required is 1.5 units.

The Clean Sweep should always return to its charging station before it runs out of power. Homeowners that come home and discover their robotic vacuum out of power in the middle of a room tend to be dissatisfied customers. Upon returning to its charging station, the Clean Sweep will automatically re-charge to full capacity. Once re-charged, it will resume cleaning until it detects that it has visited every accessible location on the current floor.

1.4 Diagnostics and Troubleshooting

Since the Clean Sweep is a new product, it is important that technical support personnel are able to troubleshoot its operation.

1.4.1 Floor Plan Dump

On request, the Clean Sweep should be able to dump its memory of the floor plan of the home. While not something a homeowner will generally need, this information can be used for diagnostics and troubleshooting. During development this will allow you to compare the Clean Sweep's perception of the home's layout to that which was being referenced by the Sensor Simulator.

See the *Sensor Simulator* section for more details.

1.4.2 Activity Log

On request, the Clean Sweep should be able to dump a log of its activity for the latest cleaning. This log should provide relevant information including:

- Sensor checks.
- Movement.
- Cleaning.

2 Sensor Simulator

The Clean Sweep must maintain an internal map of the home, similar to that shown in Figure 1. While this picture contains more details than the Clean Sweep requires, it should help you visualize the floor plan.

In order to write the control software, it is necessary to simulate the input of the Clean Sweep's sensor array as it might respond within actual homes. The Sensor Simulator provides this capability and acts as a virtual sensor array. These simulations will act as test data against which you can test your control software.

2.1 Layouts

A floor plan is a grid comprised of a series of cells. Each cell has a position, defined as a pair of coordinates. Position (0,0) is always the location of the charging station. Each cell can thus connect to at most 8 other cells. Because the placement of the charging station, and thus the placement of cell (0,0), is arbitrary, coordinates may be negative.

A	B	C
D	E	F
G	H	I

The Clean Sweep can move 90° in any direction. Consider the figure to the right. If the Clean Sweep is on cell E, it can move to cells B, D, F and H. Similarly, if it is on cell A, it can only move to cells B and D.

2.2 Floor Plan Representation

This section defines the basic XML structure that is used to represent a floor plan. The XML is shown by the snippet in Listing 1.

```
1 <floorplan>
2   <floor level='1'>
3     <cell xs='0' ys='0' ss='2' ps='1212' ds='0' cs='1' />
4     <cell xs='1' ys='0' ss='1' ps='1112' ds='1' cs='0' />
5     <cell xs='0' ys='1' ss='1' ps='1211' ds='1' cs='0' />
6     ...
7   </floor>
8 </floorplan>
```

Listing 1: Clean Sweep Layout XML.

The root of the XML is the `home` element. Each `home` may have multiple `floor` elements. The `floor` element consists of multiple `cell` elements, each of which represents an area roughly the size of the Clean Sweep's footprint.

A `cell` completely describes the configuration of the Clean Sweep's current location. In other words, we are assuming that the Clean Sweep is always com-

pletely within a `cell`. The `cell's` *x-sensor (`xs`)* and *y-sensor (`ys`)* attributes define the `cell's` position within the `floor's` grid.

The *surface sensor (`ss`)* attribute represents the surface of the cell. The value of this enumeration determines how much power is required to move into and clean that cell. The `ss` attribute can be comprised following values:

1 The cell is bare floor.

2 The cell is covered in low-pile carpet.

4 The cell is covered in high-pile carpet.

The *dirt sensor (`ds`)* indicates how many units of dirt are at the current location. While the sensor itself cannot tell exactly how many units are present, this value allows the simulator to continuously register the `cell` as dirty until the appropriate number of vacuum operations have been performed.

The `paths` attribute is a byte-array that represents the Clean Sweep's ability to move along with the x- or y-axis based on the feedback from its sensors. The first and second positions represent the increasing and decreasing directions along the x-axis respectively. The third and fourth position represent the increasing and decreasing directions along the y-axis respectively. Each value can be one of the following:

0 Unknown. The Clean Sweep doesn't yet know what's in the specified direction.

1 Open. The sensor indicates that the path is open.

2 Obstacle. The sensor indicates that the path is obstructed.

4 Stairs. The sensor indicates that the path in that direction is blocked by stairs.

Thus, for the example cell shown in Line 3 of Listing 1, the `paths` attribute value of `'1212'` indicates that movement is open in the positive `x` and `y` directions, but obstructed in the negative `x` and `y` directions.

The *charging station (`cs`)* attribute is a simple boolean that indicates whether or not the location contains the charging station.

2.2.1 Layout Files

To facilitate the testing of the Clean Sweep, the Sensor Simulator must be able to "play back" pre-defined home floor plans. Since the floor plans are stored as XML, this means that the Sensor Simulator should be able to read a floor plan file and use that file to respond to requests from the Clean Sweep's control software.

The control software itself should have no knowledge of the floor plan layout. This file is only intended to allow the sensor simulator to emit controlled output based on interactions with the control software. The fact that the entire floor

plan is available to the sensor simulator, does not mean that the Clean Sweep has *a priori* knowledge of the home's layout. The Clean Sweep control system must discover the home's floor plan just as the physical device would.

2.3 Sample Floor Plan

This section provides a more complex example of a floor plan. Figure 1 is an example of a basic floor plan. In this plan there are 3 bedrooms and a main hallway that will be cleaned. The closets and bathrooms will not be cleaned.

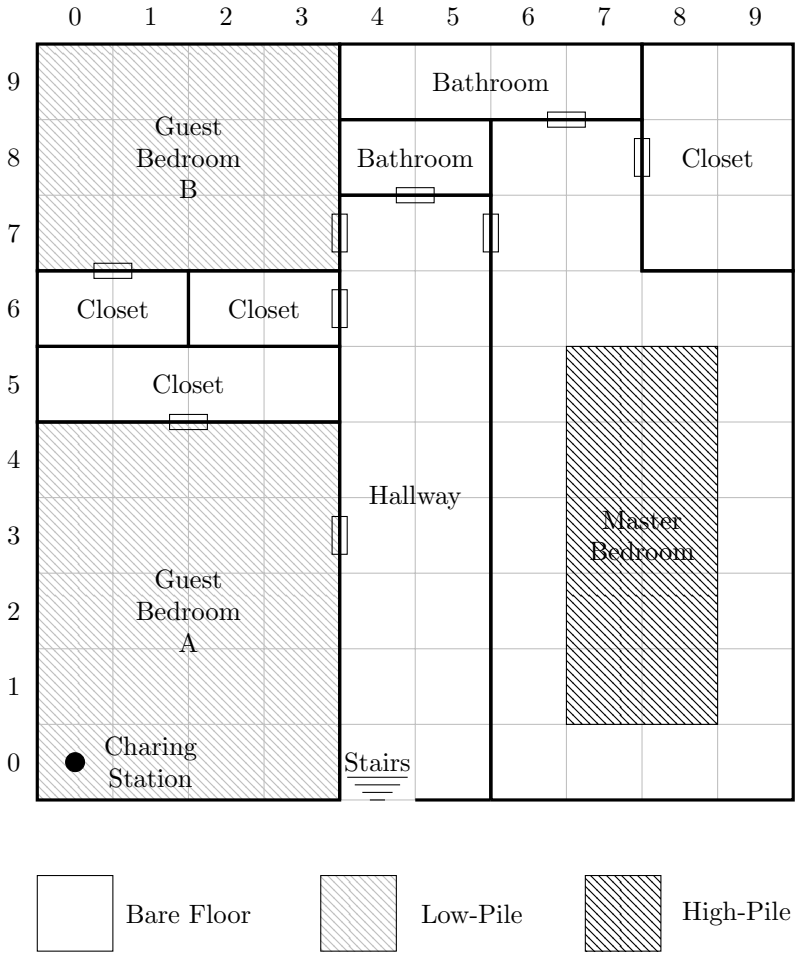

Figure 1: Sample Floor Plan

The complete XML is shown in Listing 2. In particular you should notice that there is no automatic detection of doorways. If a door is closed, then it is

treated as an obstacle. To simulate an open door, simply treat it as though the obstacle is not present.

```
<home>
  <floor level='1'>
    <!-- Guest Bedroom A -->
    <cell xs='0' ys='0' ss='2' ps='1212' ds='1' cs='1' />
    <cell xs='1' ys='0' ss='2' ps='1112' ds='1' cs='0' />
    <cell xs='2' ys='0' ss='2' ps='1112' ds='1' cs='0' />
    <cell xs='3' ys='0' ss='2' ps='2112' ds='1' cs='0' />

    <cell xs='0' ys='1' ss='2' ps='1211' ds='1' cs='0' />
    <cell xs='1' ys='1' ss='2' ps='1111' ds='1' cs='0' />
    <cell xs='2' ys='1' ss='2' ps='1111' ds='1' cs='0' />
    <cell xs='3' ys='1' ss='2' ps='2111' ds='1' cs='0' />

    <cell xs='0' ys='2' ss='2' ps='1211' ds='1' cs='0' />
    <cell xs='1' ys='2' ss='2' ps='1111' ds='1' cs='0' />
    <cell xs='2' ys='2' ss='2' ps='1111' ds='1' cs='0' />
    <cell xs='3' ys='2' ss='2' ps='2111' ds='1' cs='0' />

    <cell xs='0' ys='3' ss='2' ps='1211' ds='1' cs='0' />
    <cell xs='1' ys='3' ss='2' ps='1111' ds='1' cs='0' />
    <cell xs='2' ys='3' ss='2' ps='1111' ds='1' cs='0' />
    <cell xs='3' ys='3' ss='2' ps='1111' ds='1' cs='0' />

    <cell xs='0' ys='4' ss='2' ps='1221' ds='1' cs='0' />
    <cell xs='1' ys='4' ss='2' ps='1121' ds='1' cs='0' />
    <cell xs='2' ys='4' ss='2' ps='1121' ds='1' cs='0' />
    <cell xs='3' ys='4' ss='2' ps='2121' ds='1' cs='0' />

    <!-- Hallway -->
    <cell xs='4' ys='0' ss='1' ps='1214' ds='2' cs='0' />
    <cell xs='5' ys='0' ss='1' ps='2112' ds='1' cs='0' />
    <cell xs='4' ys='1' ss='1' ps='1211' ds='2' cs='0' />
    <cell xs='5' ys='1' ss='1' ps='2111' ds='1' cs='0' />
    <cell xs='4' ys='2' ss='1' ps='1211' ds='2' cs='0' />
    <cell xs='5' ys='2' ss='1' ps='2111' ds='1' cs='0' />
    <cell xs='4' ys='3' ss='1' ps='1111' ds='2' cs='0' />
    <cell xs='5' ys='3' ss='1' ps='2111' ds='1' cs='0' />
    <cell xs='4' ys='4' ss='1' ps='1211' ds='2' cs='0' />
    <cell xs='5' ys='4' ss='1' ps='2111' ds='1' cs='0' />
    <cell xs='4' ys='5' ss='1' ps='1211' ds='2' cs='0' />
    <cell xs='5' ys='5' ss='1' ps='2111' ds='1' cs='0' />
    <cell xs='4' ys='6' ss='1' ps='1211' ds='2' cs='0' />
    <cell xs='5' ys='6' ss='1' ps='2111' ds='1' cs='0' />
    <cell xs='4' ys='7' ss='1' ps='1121' ds='2' cs='0' />
    <cell xs='5' ys='7' ss='1' ps='1121' ds='3' cs='0' />

    <!-- Guest Bedroom B -->
    <cell xs='0' ys='7' ss='2' ps='1212' ds='1' cs='0' />
    <cell xs='1' ys='7' ss='2' ps='1112' ds='1' cs='0' />
```

```
50      <cell xs='2' ys='7' ss='2' ps='1112' ds='1' cs='0' />
51      <cell xs='3' ys='7' ss='2' ps='1122' ds='1' cs='0' />
52      <cell xs='0' ys='8' ss='2' ps='1211' ds='1' cs='0' />
53      <cell xs='1' ys='8' ss='2' ps='1111' ds='1' cs='0' />
54      <cell xs='2' ys='8' ss='2' ps='1111' ds='1' cs='0' />
55      <cell xs='3' ys='8' ss='2' ps='2111' ds='1' cs='0' />
56      <cell xs='0' ys='9' ss='2' ps='1221' ds='1' cs='0' />
57      <cell xs='1' ys='9' ss='2' ps='1121' ds='1' cs='0' />
58      <cell xs='2' ys='9' ss='2' ps='1121' ds='1' cs='0' />
59      <cell xs='3' ys='9' ss='2' ps='2121' ds='1' cs='0' />
60
61      <!-- Master Bedroom -->
62      <cell xs='6' ys='0' ss='1' ps='1212' ds='1' cs='0' />
63      <cell xs='7' ys='0' ss='1' ps='1112' ds='1' cs='0' />
64      <cell xs='8' ys='0' ss='1' ps='1112' ds='1' cs='0' />
65      <cell xs='9' ys='0' ss='1' ps='2112' ds='1' cs='0' />
66      <cell xs='6' ys='1' ss='1' ps='1211' ds='1' cs='0' />
67      <cell xs='7' ys='1' ss='4' ps='1111' ds='2' cs='0' />
68      <cell xs='8' ys='1' ss='4' ps='1111' ds='2' cs='0' />
69      <cell xs='9' ys='1' ss='1' ps='2111' ds='1' cs='0' />
70      <cell xs='6' ys='2' ss='1' ps='1211' ds='1' cs='0' />
71      <cell xs='7' ys='2' ss='4' ps='1111' ds='2' cs='0' />
72      <cell xs='8' ys='2' ss='4' ps='1111' ds='2' cs='0' />
73      <cell xs='9' ys='2' ss='1' ps='2111' ds='1' cs='0' />
74      <cell xs='6' ys='3' ss='1' ps='1211' ds='1' cs='0' />
75      <cell xs='7' ys='3' ss='4' ps='1111' ds='2' cs='0' />
76      <cell xs='8' ys='3' ss='4' ps='1111' ds='2' cs='0' />
77      <cell xs='9' ys='3' ss='1' ps='2111' ds='1' cs='0' />
78      <cell xs='6' ys='4' ss='1' ps='1211' ds='1' cs='0' />
79      <cell xs='7' ys='4' ss='4' ps='1111' ds='2' cs='0' />
80      <cell xs='8' ys='4' ss='4' ps='1111' ds='2' cs='0' />
81      <cell xs='9' ys='4' ss='1' ps='2111' ds='1' cs='0' />
82      <cell xs='6' ys='5' ss='1' ps='1211' ds='1' cs='0' />
83      <cell xs='7' ys='5' ss='4' ps='1111' ds='1' cs='0' />
84      <cell xs='8' ys='5' ss='4' ps='1121' ds='1' cs='0' />
85      <cell xs='9' ys='5' ss='1' ps='2121' ds='1' cs='0' />
86      <cell xs='6' ys='6' ss='1' ps='1211' ds='1' cs='0' />
87      <cell xs='7' ys='6' ss='1' ps='1111' ds='1' cs='0' />
88      <cell xs='8' ys='6' ss='1' ps='1121' ds='1' cs='0' />
89      <cell xs='9' ys='6' ss='1' ps='2121' ds='1' cs='0' />
90      <cell xs='6' ys='7' ss='1' ps='1211' ds='1' cs='0' />
91      <cell xs='7' ys='7' ss='1' ps='2111' ds='1' cs='0' />
92      <cell xs='6' ys='8' ss='1' ps='1221' ds='1' cs='0' />
93      <cell xs='7' ys='8' ss='1' ps='2121' ds='1' cs='0' />
94    </floor>
95  </home>
```

Listing 2: Sample Floor Plan XML.

3 The Task

Your team will play the role of both "business" and "development". The instructor will play the role of "management". Your task is to develop the Clean Sweep's control system and sensor simulator.

3.1 Development

You are responsible for following Agile practices including working with the business to write user stories, estimating the development time, planning your releases, tracking your velocity and updating management's expectations accordingly, incorporating change requests, writing the various automated tests, and following development best practices including using source code control, continuous integration, constant code refactoring and automated build scripts.

If there are any questions about project functionality, management will provide the final determination.

3.1.1 Constraints

There are some constraints on how you may implement the solution. First, it must be written entirely in Java. Second, it must not require the installation of any other software like databases. Because the Clean Sweep's memory is limited, any third-party software or libraries must be carefully evaluated before they will be allowed. Management has the final say on the use of all such libraries.

There are no restrictions on the libraries that can be used for the Sensor Simulator. Since that software will not be installed within the actual Clean Sweep vacuum cleaner, there is greater flexibility.

You must develop the code as two separate modules: one for the Control System and the other for the Sensor Simulator. Remember that the control system must not have perfect knowledge of the floor plan; it derives all of its information from interactions with the sensors and those are driven by the information encoded in the floor plan layout document.

3.1.2 User Stories & Tracking

Your team is responsible for creating and maintaining a list of user stories. You may use whatever means you wish, but you must be able to submit the final list to me as a document; I will not log in to some external account to review your work.

You are responsible for tracking which stories were completed within each iteration so that you can construct the project burn down chart for the final deliverable. You are also responsible for tracking the estimates and actual time required for each story as well as its priority.

3.1.3 SCM & Git

Your team is responsible for storing your source code in the Git repository that has been provided for you:

`http://selab.cdm.depaul.edu/gitbucket/`

Make sure that you commit your code constantly so that you get the most advantage from the environment. You can also use your team repository to store other working documents. Those documents will not be evaluated.

3.1.4 Builds & Automation

Your team must also provide appropriate build scripts that will be used by the Jenkins continuous integration server that has been provided:

`http://selab.cdm.depaul.edu/jenkins/`

The build scripts must be written in Ant or Maven.

3.1.5 Sonar & Code Quality

The build script must publish the results of the build to the Sonar server that has been provided:

`http://selab.cdm.depaul.edu/sonar/`

You should take advantage of the information that Sonar provides you to improve the quality of your code.

3.2 Iteration Retrospectives

Your team is required to prepare a 15-minute retrospective for iterations 1-3. The retrospective for iteration 4 will be incorporated into the project retrospective during the final week of class. Each of these iteration retrospectives should cover the following topics:

- A brief demonstration of new the functionality.
- The project's schedule, with the stories that you expected to complete during the iteration.
- A breakdown of the high-level activities that the team worked on and the time spent on those activities.
- Show the iteration's project velocity and burn down charts. Discuss any times you needed to readjust your velocity, why that was necessary, and the impact it had on your ability to deliver the functionality for the iteration.

- Highlight any stories that needed to be carried forward to the next iteration and explain the impact to the deliverables.

3.3 Project Retrospective

Your team is required to prepare a 30-45 minute retrospective of your project. The retrospective should cover the following topics:

- Demonstrate the complete functionality.
- Describe how your team used Agile processes.
- Provide a summary breakdown of the high-level activities that the team worked on and the time spent on those activities.
- Provide charts that show week-by-week project velocity and burn down. Discuss any times you needed to readjust your velocity, why that was necessary, and the impact it had on your ability to deliver the required functionality.
- Describe 2-3 of your most important lessons learned. What would you do differently? What would you do the same?
- Describe which Agile practices you found most helpful, least helpful, and why.
- Review the current output of Sonar and describe how your team used it during your project.

The purpose of these retrospectives is two-fold. The first is to show off your team's work. The second is the encourage dialog across the different teams with respect to what Agile techniques worked and didn't work and to discuss why.

4 Schedule

For this project, we will use the following tentative schedule:

Week 1: Review the project.

Week 2: Identify the teams and review the project.
Define user roles.
Write epics and user stories.

Week 3: Prepare a 3-iteration schedule for story implementation.
Define, prioritize and estimate user stories for iteration 1.
Estimate velocity for iteration 1.
Prepare technical infrastructure.

Week 4: Present your team's schedule.
Implement iteration 1 user stories.
Calculate weekly velocity.

Week 5: Implement iteration 1 user stories.
Prepare iteration 1 retrospective.
Calculate weekly velocity and overall velocity for iteration 1.
Estimate velocity for iteration 2.
Define, prioritize and estimate user stories for iteration 2.

Week 6: Iteration 1 retrospective.
Implement iteration 2 user stories.
Calculate weekly velocity.

Week 7: Implement iteration 2 user stories.
Prepare iteration 2 retrospective.
Calculate weekly velocity and overall velocity for iteration 2.
Estimate velocity for iteration 3.
Define, prioritize and estimate user stories for iteration 3.

Week 8: Iteration 2 retrospective.
Implement iteration 3 user stories.
Calculate weekly velocity.

Week 9: Implement iteration 3 user stories.
Prepare iteration 3 retrospective.
Calculate weekly velocity and overall velocity for iteration 3.

Week 9: Iteration 3 retrospective.

Week 10: Prepare project retrospective.

Week 11: Project retrospective.

5 Evaluation

Your project will be evaluated based on the following deliverables:

20% Presentation of bi-weekly iteration progress that includes: demonstrations of new functionality, calculations of project velocity and its impact on subsequent iterations, and burndown charts.

15% Documented epics, user stories, and tasks as well as their schedule and estimates.

15% Working code and associated automated, working, passing tests.

10% Final, comprehensive presentation.

10% Ongoing use of Git for source code control.

10% Automated build scripts using Ant or Maven.

10% Continuous integration using Jenkins.

10% Code quality reports from Sonar including: PMD, FindBugs, and Cobertura.

A Useful Resources

The following are some references that may prove useful to you in working with the various tools that comprise this project. The list isn't comprehensive and if you find a good resource that you think might help other students, please send it to me and I'll review it to see if it should be included.

A.1 Git

`http://git-scm.com`

A.2 Ant & Maven

`http://ant.apache.org/`
`http://maven.apache.org/`

A.3 Continuous Integration

`http://jenkins-ci.org/`

A.4 Code Quality

`http://cobertura.sourceforge.net/`
`http://pmd.sourceforge.net/`
`http://findbugs.sourceforge.net/`

A.5 Sonar

`http://www.sonarsource.org/`