A Prototyping Discipline in OpenUP to Satisfy Wireless Sensor Networks Requirements

Gian Ricardo Berkenbrock
Software/Hardware Integration Lab.
Federal University of Santa Catarina (UFSC)
Joinville, SC, Brazil
Email: gian.rb@ufsc.br

Carla Diacui Medeiros Berkenbrock
Santa Catarina State University (UDESC)
Department of Computer Science
Joinville, SC, Brazil
Email: carla.berkenbrock@udesc.br

Celso Massaki Hirata
Department of Computer Science
Aeronautics Institute of Technology (ITA)
S.J.Campos - SP, Brazil
Email: hirata@ita.br

***Abstract*—Wireless Sensor Networks (WSNs) are used to collect data from different sources and they can be applied in monitoring and instrumentation areas. WSN are highly dependent on application requirements, then one application is hardly equal to another. There is not a specific process to address the development of WSN applications. Open Unified Process is an iterative software development process that is intended to be minimal, complete, and extensible, and because of these features it is a good candidate for WSN application development. However, OpenUP does not support the challenges and requirements of WSN systems, because it does not have specific tasks that consider such requirements. Then, in order to address this lack of support, this paper proposes a prototype discipline that can be integrated into software development process for WSN applications.**

Keywords–Software Engineering; Prototype Discipline; Discrete Simulation.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) are used in environmental monitoring, surveillance of installations or areas, such as, home, border, or building, object tracking, precision agriculture, bridge monitoring, hospital monitoring, and herd monitoring. WSNs are composed of nodes that have capability to sense and communicate over-the-air to each other. The nodes have also processing capability and local storage. In order to deliver the collected data to a base station in a multi-hop scenario, nodes transmit data using ad hoc communication. The nodes have the capability to create the wireless interconnection network, which is needed to delivery the data.

The number of nodes of WSNs can change from a few nodes to hundreds of thousand nodes. And they can be static or mobile. In order to aggregate all the aforementioned characteristics, the node platform has some drawbacks, e.g., short communication range, low bandwidth, small memory, and limited battery. These drawbacks are restrictions that are inherent to any WSN application.

The feasibility of WSN is highly dependent on application requirements mainly due to the restrictions aforementioned. Even when the application has similar requirements to other previously deployed application, a complete application verification and validation must be performed again in the new environment. The main cause is that the behavior of a WSN can differ from the other previously known application, and the target environment can be different.

The development of WSN applications is challenging and demands a peculiar effort. The difficulties include requirements satisfaction, gap between model and implementation, specific hardware platform, system validation, verification, testing [1] [2]. Then, during the WSN application development the team members inform the project management about some concerns related to prototyping. For example, hardware test and approval, application test, and third-party system integration evaluation are related to prototyping. So, we argue that one should consider the use of specific software development process to improve the development organization, deal with such concerns, and to enable a better project management. When the organization uses a process for software development, it can have the opportunity to reproduce the process in the following projects and to enhance it with the feedback of the previous team. Software development process [3] is a set of activities whose goal is the development or evolution of software. An example of software development process is Open Unified Process (OpenUP) [4].

OpenUP is an open iterative software development process that is intended to be minimal, complete, and extensible [5]. It can be used to develop software of different purposes, from small and embedded to desktop enterprise application. For example, some WSNs projects are Aquila Tower Monitoring [6], and Aqua WSN [7]. Nevertheless, OpenUP does not address the specific requirements for development of WSN systems. Thus, it is difficult to achieve a predictable system behavior that complies with the WSN application requirements.

There are some studies reporting the extension and integration of OpenUP, such as [5] for specifying capacity requirements, and [8] for security. Yet complex products, such as WSN, need special handling of requirements as well. However, there is a lack of definition in the existing software development process to address the development of a WSN application project. The main reason why development

Thanks to CAPES process nr. 6804-14-4.

 ISBN: 978-1-61208-367-4

processes do not fit to WSN development is the inability to meet important requirements of WSN satisfactorily and they do not provide a way to manage activities that are needed through the development, such as, simulation and prototyping activities.

In addition, other characteristics that need to be addressed during the development process of the WSN applications are the data integration with third party systems, employment of verification techniques, and different viewpoints used for analysis. Yet, due to the development characteristics of WSN systems, it is important to aid the development team with a disciplined way to perform their tasks, including prototyping activities.

This paper is organized as follows. Section II investigates some related works. Section III introduces the Prototype Disciplines that can be integrated into software development process for WSN applications. Section IV presents details of work products integration from the Prototype discipline to the OpenUP development process. Section V illustrates the use of proposed discipline. Finally, Section VII concludes this paper.

II. Related Work

There are several studies reporting processes for software development in constrained environments [9] [10] [11] [12]. However, there is not a specific process to address the development of WSN applications. Developers should be aware of restrictions such as limited storage, battery consumption, low accuracy sensor, and short transmission range. WSN are highly dependent on application requirements, then one application is hardly equal to another.

Carvalho et al. [9] conducted a comparative investigation between applications of two software development processes: Scrum[13] and Rational Unified Process [14]. The authors concluded that it is necessary high effort in the traditional method compared to the agile software development.

According to Marincic et al. [10], when the next generation of a system is designed, the new system will have common elements with its previous version. Then, the authors propose a framework for identifying the non-formal elements of knowledge which can support modelling of the next system generation. The authors presented the application their framework modelling mechanical parts of a paper-inserting machine on an action research case.

The development process proposed by Nosseir et al. [11], called Mobile Development Process Spiral, is a usability driven model designed to integrate usability into existing application development processes. It also recommends usability techniques for assessing mobile applications. The proposed method aims identifying a set of usability techniques and incorporating these techniques into each iteration to assess the mobile applications.

According to Berrani [12] WSN specification is a complex task due to their embedded and distributed nature as well as the strong interaction between their hardware and software parts. In order to improve the verification of the WSN properties the authors propose an approach called Model Driven Architecture (MDA). This approach aims to promote the reusability and improve the development process. The authors mentioned that their approach promoted the reusability of modeled components and it also facilited the modeling task decreasing relative costs.

III. Prototyping Discipline

The use of prototypes during project development helps to improve the knowledge about the system, the network, and the nodes. In this research a prototype has at least one real node and the working code is deployable to the node's hardware. The scope of discipline is restricted to software prototypes. Then the prototype discipline can be used at any process phase of the OpenUP.

The prototyping discipline considers specific tasks to build a prototype during the software development process for WSN application. The workflow is depicted in Figure 1. The workflow begins by defining the objective for the prototype study, it drives all the further decisions. After that, a specification regarding the prototype requirements is elaborated. Afterwards, the prototype design based on information from previous activities performed can be started. Then, in parallel, the development and the calibration activities are performed, followed by the test activity, which verifies if the code complies with the objectives. So, the code is compiled and deployed in the nodes to perform the experiments which can begin. After running the experiments, the results are evaluated and depending on them it might be needed a code review and new experiments or it proceeds to document the generated results. In the remainder of this section details of each activity of this discipline are given.

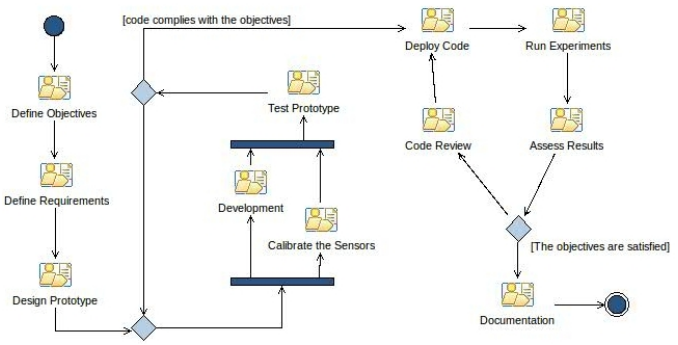

Figure 1. Workflow for prototype discipline

A. Define Objectives

This activity defines which is the objective for the prototype discipline. The prototyping model used to perform this discipline relies directly on the chosen objective. For example, one can create a prototype only to test the sensors and after the study has finished, the prototype is not needed any longer. This activity is performed by *Prototype Analyst* and the main output work product is the *Prototyping Objectives*. This activity has two tasks: *Define Objectives*, and *Plan Prototyping*. The first task aims to define the overall prototyping objective and the second one aims to plan the prototyping activities. The *Prototyping Plan* is used as an input work product of the design task performed later.

 ISBN: 978-1-61208-367-4

B. Define Requirements

After the objectives are defined, the definition of the prototype requirements is produced. The requirements drive the prototype development. It is performed by the *Prototype Analyst*. This activity produces the *Prototype Requirements Specification* work product.

C. Design Prototype

This activity consists of describing the blueprints that guide the development. Some information in this activity describes, for example, the detailed communication model, topology to be followed, which programming approach to be used and a detailed software architecture, and even the base station behavior if it is applied for the study. This activity is performed by the *Prototype Analyst* role and he can be assisted by the *Prototype Developer* role. The *Software Specification*, *Protocols Definition*, and *Base Station Behavior Description* are the main output work products of this activity.

D. Develop Prototype

In this activity, the code is implemented. This activity is performed by the *Prototype Developer* role and it produces the *Code* that is compiled and deployed to the real nodes. In addition, the *Prototype Developer* needs to deal with the WSN restrictions during the coding tasks. He/She also needs to deal with the specified middleware, OS, and hardware platform details. The role has to manage the code size, because of the memory size available in the hardware platform. It is expected that this activity demands a considerable effort. Finally, in order to assist the coding, one can use the simulation model for, if it is applied, developing the WSN application.

E. Calibrate Sensors

This activity is performed by the *Phenomenon Specialist* and he can be assisted by the *Prototype Developer*. It consists of verifying the values of the prototype and then calibrates it closer to the real values as possible. So, during the experiments the values obtained from the prototype are more reliable. This activity has the *Sensor Calibrated* as an outcome.

F. Test Prototype

Tests are performed in order to verify if the code complies with the prototyping objectives of the current iteration and if there is no error in the code. This activity is performed by the *Prototype Tester* and it is expected as results the *Test Log* work product. The *Test Log* contains the results of performed tests and if some test fails the process resumes back in the *Development* activity otherwise it proceeds to *Deploy the Code* activity.

G. Review Code

The *Review Code* activity is only performed if the experiment results do not fulfill the prototype objectives and requirements. It consists of performing a review of the code deployed after running the experiments and assess its results. The *Prototype Developer* does the solicited changes in *Code Review* task and then the *Prototype Tester* performs again the tests in the code reviewed. The process resumes in the *Deploy Code* activity only when the complete review is finished.

H. Deploy Code

Deploy the code in the nodes is an activity performed by the *Prototype Developer*. It is performed using the tools available from the chosen OS. In some cases, it can be made via over-the-air communication, but commonly the node is plugged to the development station and then the binary code is deployed to the connected node through the cable. The outcome from this activity is *Prototype Nodes with Code* loaded. The task *Deploy* is performed with the nodes that are considered for the experiment.

I. Run Experiments

This activity is executed by the *Prototype Developer* with the aid of the *Prototype Analyst*. *Run Experiments* activity aims to obtain the experiments results that are used in the *Assess the Results* activity in order to evaluate the current study. The main output work product is the *Experiment Results*.

J. Assess Results

After the experiments results are available, further analysis are performed. The *Prototype Analyst* role is responsible for performing the analysis and he can have assistance from the *Phenomenon Specialist* role. The analysis can be made using statistics techniques and tools. In addition, the activity provides information, via its main output work product *Experiment Results Evaluated*, to make a decision if the experiments results satisfy the prototype's objective. If the results do not fulfill the objective then the process resumes in the *Code Review* activity. Otherwise, the next activity is the *Documentation* activity.

K. Document Prototype

This activity consists of evaluating the results provided by the *Assess Results* activity. The *Prototype Report* is generated. In this report some information is available, such as solicitation for requirements, parameter, and sensors review, or it can provide information how close the prototype is to reach the quality of the final system. The report can have also decisions resulting from the experiments evaluated regarding the prototyping objectives. The *Prototype Analyst* role performs the *Documentation* task.

IV. Integration with OpenUP

During the development process, the team members inform the project management some concerns that include: further requirement analysis, debugging results, communication analysis, system performance analysis, design evaluation, scalability validation, hardware test and approval, application test, third-party system integration evaluation, requirements refinement, and training results. Some concerns of hardware test and approval, application test, third-party system integration evaluation, requirements refinement are related to prototyping. The concerns are reported to project management and then a decision regarding of which approach is used in sequence is taken.

The project team performs an assessment of the raised concerns and then the related information is updated to the following work products: *Iteration Plan*, *Project Plan*, *System Wide Requirements*. These work products are the main inputs

 ISBN: 978-1-61208-367-4

to start the prototyping iteration. After the selected iteration is executed, it provides a report with the answers and recommendations to the project. The prototyping discipline provides the *Prototype Report*. It is important to mention that this report is not the only work product outputs, the iteration also generates other work products of interest, for example, a validated model, experiments results assessed, and application code evaluated.

Table I presents details of work products integration from the Prototype discipline to the OpenUP development process. Table I also describes how the work product is used by the proposed disciplines: as input or as output. It indicates which tasks use the work product as input and which tasks produce the work product.

TABLE I. Work products interaction of Prototyping Discipline

Work Product (WP)	**Type**	**Task that produces the WP**	**Task that uses the WP**
Iteration Plan	Input	*Plan Iteration/ Manage Iteration*	*Define Objectives*
Project Plan	Input	*Plan Project*	*Define Objectives/ Plan Prototyping*
Vision	Input	*Develop Technical Vision*	*Requirements Specification/ Define Objectives*
System-Wide Requirements	Input	*Identify and Outline Requirements/ Detail System-Wide Requirements*	*Requirements Specification*
Middleware Definition	Output	*Design Prototype*	*Implement Solution*
Protocols Definition	Output	*Design Prototype*	*Implement Solution*
Message Behavior Description	Output	*Design Prototype*	*Implement Solution*
Software Specification	Output	*Design Prototype*	*Implement Solution*
Base Station Behavior Description	Output	*Design Prototype*	*Implement Solution*
Code	Output	*Prototype Building*	*Implement Solution*
Test Case	Input	*Create Test Cases*	*Test*
Prototype Report	Output	*Documentation*	*Assess Results/ Detail System Requirements*

The following comments are related to the tasks that use the work products of the Table I. The *Define Objectives*, *Plan Prototyping*, and *Requirements Specification* tasks use the *Vision*, *Iteration Plan*, *Project Plan*, and *System-wide Requirements* work products that provide essential information to perform the tasks. The *Implement Solution* task is performed after the *Inception* phase is executed. It uses the following work products: *Middleware Definition*, *Protocols Definition*, *Message Behavior Description*, *Software Specification*, *Base Station Behavior Description*, and *Code*. The next work product, *Test Case*, is used in *Test* task. The input work product is the *Test Case* - it has the specification of a set of test inputs, execution conditions, and expected results related to some scenario. The work product *Prototype Report* is the most useful to the development process because it provides a feedback of the prototype iteration to the *Assess Results* task and the *Detail System-Wide Requirements* task. For example, the *Prototype Report* contents are details about the network protocol performance, and if the employed network protocol complies with the application requirements.

Therefore, if the integration requirements are satisfied, e.g., the process analyst can map all input and output work products, then our proposed extension could be integrated to other software development process. Thus, it can be used for development of WSN applications as needed.

A. Publishing

In this paper, Prototyping Discipline is described using the Eclipse Process Framework (EPF) – version 1.5.1. EPF enables the process manager/analyst to update all components of the process in use and also enables to publish the desired process. The process can be available directly inside the tool or it can be published for web access. The availability inside the EPF helps the process analyst to preview the web site structure before publishing it. For authoring, the EPF has available a perspective which provides the necessary set of solutions for method composing.

If the organization needs to provide access via web, then the process analyst just needs to generate the web pages, and then they can be published, for example, in the intranet of the organization or in the public web site. So, the published web pages are available via the web browser. For example, Figures 2 and 3 depict our proposed extensions available using the web browser.

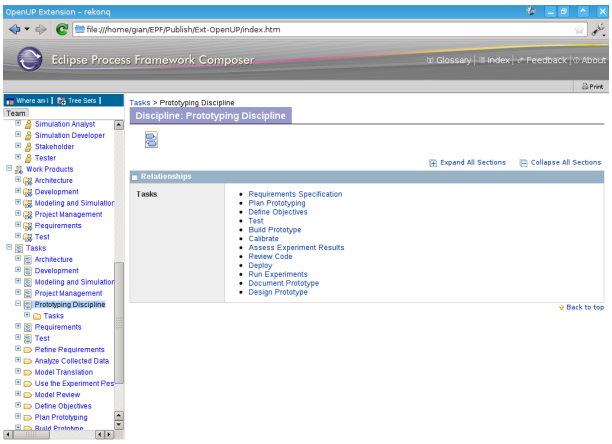

Figure 2. Consulting the extension via web browser

V. Case Study

This section aims to illustrate the use of prototype discipline for the software development process for WSN application. The case study is about monitoring a cellar used to age Brazilian sugar cane spirits. A description of the application, some requirements, and the achieved results are given.

A. Description

The sugar cane spirits (Cachaça) is a genuine Brazilian drink, known worldwide. Its production began in the sixteenth century. According to Brazilian laws, which standardize and rank drinks, cachaça is defined as a typical beverage produced

 ISBN: 978-1-61208-367-4

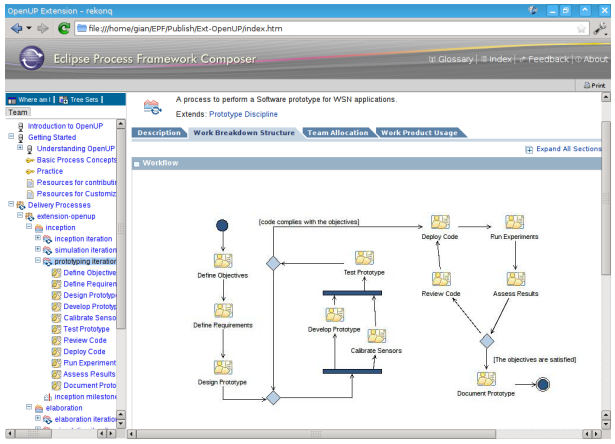

Figure 3. Consulting the prototyping workflow via web browser

in Brazil, with an alcoholic strength of 38 % vol (thirty-eight percent by volume) to 48 % vol (forty-eight percent by volume) at a temperature of 20^o C (twenty degrees Celsius). It is obtained by distilling the fermented juice of sugar cane with peculiar sensory features.

The production cycle of sugar cane spirits starts with the milling of sugar cane, through the preparation of the wort, fermentation, distillation, filtration and dilution. After filtering and resting, the sugar cane spirits can be bottled or stored in wooden barrels for aging.

The longer the aging of the sugar cane spirits, the higher is the value of the drink. In the aging process, the characteristics of the sugar cane spirits change, improving their qualities with new flavors, new tastes and new coloring.

B. User Environment

The main environment place is a cellar for aging drinks. The tasks related with the application in the environment are: deployment task, monitoring task, maintenance task. The considered platform is TelosB. TelosB is a WSN platform with a TI-MSP430 micro controller, 128 kbytes of memory, supported by TinyOS and ContikiOS, and it has a light, temperature, and humidity sensor. The maintenance task occurs only once a year. The monitoring task must inform the control station when the cellar is out of its environmental specification. So, in order to keep the cellar with the appropriate condition, a notification must be delivered. The previous notification occurs when the cellar environment reaches a critical temperature or humidity, close to the upper or lower bounds. Another notification must be sent if the cellar is out of its upper or lower bound for any given environment condition. The control station must store all the received events.

C. Functional Requirements

The sugar cane spirits age in a cellar with dimension of 8 meters wide per 80 meters long and 5 meters high. The cellar can store 800 barrels with capacity of 250 liters each or 200,000 liters in total. The barrel dimensions are 95 cm of height, 72 cm of diameter at middle, and 58 cm of diameter at top/bottom. During the aging time (from 1 year up to 5 years) the temperature must be within the range of 15^o C (fifteen degrees Celsius) to 20^o C (twenty degrees Celsius) and the air relative humidity must be from 70% to 90%. Figure 4 shows the arrangement of barrels in the cellar and the circles represent the nodes' positions.

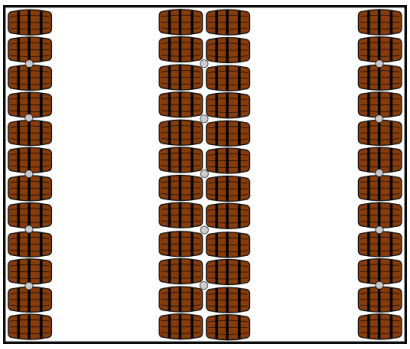

Figure 4. Cellar illustration with the barrels and the position of the nodes

In WSN application, the nodes must be able to set upper and lower bounds for the environmental variables (temperature and humidity). If the sensed value gets closer to the bounds, the node must send a message to the control station. It only stops sending the message when the control station notifies that the message was received. The node sends the message again at a 5 time intervals. Nodes should be able to connect to the network and it must be able to perform the following activities: sensing, routing, disseminating, aggregating. A threshold should be in two levels, either for upper and lower bounds, one soft threshold 10% lower than the hard threshold. All the parameters must be changeable and be retrievable through the control station. At least, once a day, the node must inform its sensing values to the sink node. All information received in the sink node is retransmitted it to the control station. When all nodes send the alarm message to the control station, they indicate their battery levels. The nodes are deployed one meter above the barrels and two meters from each other in the same row.

The Control Station communicates to the network via the node sink, i.e., it is directly connected to the sink node. It must also indicate the battery level of all nodes that perform any communication with it. Additionally, it must indicate the upper and lower threshold used by each node. The Control Station must be able to set a parameter of a single node or the whole network, and it must indicate all alarms received and notify back to the source node.

VI. Results

The case study was performed by only one person. He performed all the roles. During the execution of one simulation study, the time needed to execute each activity was measured. The whole simulation discipline was completed in fourteen days, including weekends and the prototype discipline was performed in 21 days.

Figure 5 depictes the relative time spent for each activity. The activities with zero time were not executed. And before, after, and between the execution of the proposed discipline, an Inception iteration from OpenUP is executed. Then, the iteration executions are Inception, Simulation, Inception, Prototyping, and Inception.

 ISBN: 978-1-61208-367-4

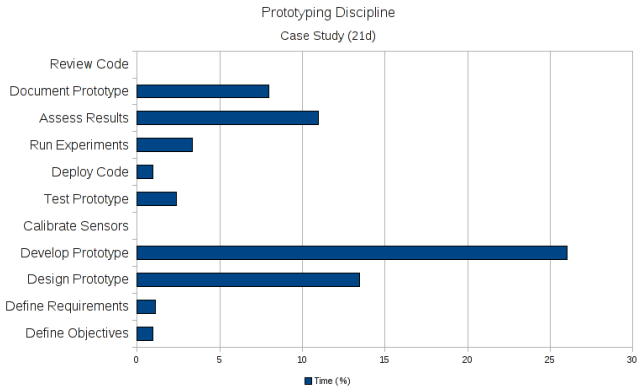

Figure 5. Relative time to perform each activity

After the entire process execution, the results are compiled and they show as expected. In each experiment, we changed different MAC protocol parameters, such as, duty cycle, listening interval, and transmission power.

Furthermore, the activities that demanded more effort to complete in the prototype discipline was the *Development* followed by *Design Prototype* and then by *Assess Results*. The longest activity is *Run Experiments* that took more than eight days to complete.

Using this discipline to perform the prototyping aided the developer to organize the work and to be more objective in each step of the process.

VII. Conclusion

This paper presented an extension for OpenUP. The purpose of this paper is to introduce the discipline Prototyping for developing WSN applications. The prototyping discipline was created in order to improve the knowledge about the system as well as to refine information about the system. This paper detailed the discipline and described the discipline activities. In addition, there is a description about how this discipline integrate with OpenUP and we argue that if the process analyst maps all input and output work products in a new software development process then the discipline can be used in that software development process. The EPF publishes the discipline by making available via web.

References

[1] A. Hasler, I. Talzi, J. Beutel, C. Tschudin, and S. Gruber, "Wireless sensor networks in permafrost research: Concept, requirements, implementation, and challenges," in Proceedings... International Conference on Permafrost (NICOP), 2008, pp. 669–674.

[2] A. Lédeczi, P. Völgyesi, M. Maróti, G. Simon, G. Balogh, A. Nádas, B. Kusy, S. Dóra, and G. Pap, "Multiple simultaneous acoustic source localization in urban terrain," in Proceedings..., International Symposium on Information Processing in Sensor Networks. Piscataway, NJ, USA: IEEE Press, 2005, p. 69.

[3] I. Sommerville, Software Engineering, 7th ed. Addison-Wesley, 2004.

[4] IBM, "Openup," 2009.

[5] A. Borg, K. Sandahl, and M. Patel, "Extending the openup/basic requirements discipline to specify capacity requirements," in Proceedings..., IEEE International Conference on Requirements Engineering. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 328–333.

[6] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon, "Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment," in Proceedings..., International Conference on Information Processing in Sensor Networks. Washington, DC, USA: IEEE Computer Society, 2009, pp. 277–288.

[7] H. Ntareme, M. Zarifi, A. Ergawy, S. Rathore, K. Wang, and A. Strikos, "Aqua wireless sensor networks," January 2008. [Online]. Available: http://www.online.kth.se/csd/projects/0726/

[8] S. Ardi and N. Shahmehri, "Integrating a security plug-in with the openup/basic development process," in Proceedings..., International Conference on Availability, Reliability and Security. Washington, DC, USA: IEEE Computer Society, 2008, pp. 284–291. [Online]. Available: http://portal.acm.org/citation.cfm?id=1371602.1371921

[9] S. Carvalho, F. Motta Cardoso, A. Da Cunha, and L. Zanetti, "A comparative research between scrum and rup using real time embedded software development," in Information Technology: New Generations (ITNG), 2013 Tenth International Conference on, April 2013, pp. 734–735.

[10] J. Marincic, A. Mader, R. Wieringa, and Y. Lucas, "Reusing knowledge in embedded systems modelling," Expert Systems, vol. 30, no. 3, 2013, pp. 185–199. [Online]. Available: http://dx.doi.org/10.1111/j.1468-0394.2012.00631.x

[11] A. Nosseir, D. Flood, R. Harrison, and O. Ibrahim, "Mobile development process spiral," in Computer Engineering Systems (ICCES), 2012 Seventh International Conference on, Nov 2012, pp. 281–286.

[12] S. Berrani, A. Hammad, and H. Mountassir, "Mapping sysml to modelica to validate wireless sensor networks non-functional requirements," in Programming and Systems (ISPS), 2013 11th International Symposium on, April 2013, pp. 177–186.

[13] K. Beck and C. Andres, Extreme programming explained. Embrace change. Addison–Wesley Boston, 2005, vol. 2.

[14] P. Kruchten, The Rational Unified Process: An Introduction, 3rd ed. Addison Wesley, 2003.

 ISBN: 978-1-61208-367-4