Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 9 (2013) 1245 – 1251

Procedia Technology

CENTERIS 2013 - Conference on ENTERprise Information Systems / ProjMAN 2013 - International Conference on Project MANagement / HCIST 2013 - International Conference on Health and Social Care Information Systems and Technologies

Utilization of Software Product Lines for generation of Patient Monitoring Systems and Sending Alerts

Bruno Gomes de Araújo[a,b,*], Philippi Sedir Grilo de Morais[b], Jailton Carlos de Paiva[b], Ricardo Alexsandro de Medeiros Valentim[b], José Diniz Júnior[b], Francis Solange Vieira Tourinho[b]

[a]*Federal Institute of Education, Science and Technology of Rio Grande do Norte, São Braz Street, 304, Santa Cruz and Postcode: 59200-000, Brazil*

[b]*Federal University of Rio Grande do Norte, Campus Universitário, Lagoa Nova, Natal and Postcode: 59078-970, Brazil*

Abstract

Software Product Lines (SPL) are responsible for the generation of systems based on a selection of its features. Despite the ease of handling ordinary systems in the hospital environment, there are processes which handle some critical information that are related to time constraints (Real-Time). In this context, this work implements an SPL focused on the generation of patient monitoring systems. A tool named Captor was used in this work, in order to provide forms to control the variability of the system that will be generated. So as to validate the SPL, two systems were generated by the tool through the selection of a combination of the available features.

Keywords: Software Product Lines; Real-Time; Patient Monitoring; Alerts; Features

* Corresponding author. Tel.: +55-84-96745248.
E-mail address: bruno.gomes@ifrn.edu.br

1. Introduction

One of the main goals in software development is high quality development, low cost and short-term production. Due to the large amount of systems in the market, a practice that is gaining prominence is software reuse, which allows the use of concepts or products previously acquired or built as part of a new development process [1][2].

Software reuse facilitates and speeds up the development of new software and troubleshooting. Among the used techniques are frameworks, software components, among others. The evolution of these ideas led to the formulation of the concept of Software Product Lines (SPL), responsible for producing a collection of products belonging to a "family". The concept of family systems has emerged because of some software that shared a set of common characteristics, differing from each other only by variable characteristics, known as variabilities[3][1].

Howerver, some applications present a higher degree of complexity during its development and in the daily usage, which makes it hard for the creation of a software product line, such as hospital systems, particularly in sectors that need to perform monitoring in hospitalized patients. These systems are run periodically and must follow strictly certain time constraints and because of this, they are considered critical systems. This kind of system is becoming increasingly common in the market, and its applicability range from smart controllers embedded in home appliances, to military defense systems, and air traffic and railroad controls [4][5].

This paper aims to demonstrate how to design a Software Product Line for a patient monitoring system and alert sending, using some of the existing practices in the literature and mentioning the necessary care needed for the generation of systems before the family they belong to. A tool named Captor is used for the elaboration of the product line and systems generation.

2. Software Product Lines for Patient Monitoring Systems and Alert Sending

Software Product Lines can be applied to systems in various domains. The concern is to select the features of each system, thereby identifying common and variables characteristics among them and in this way elaborate the System Product Line.

For the system development, a monitoring system for patients in a hospital was used as a case study. The system allows the monitoring of body temperature and pressure signals of a particular patient, and, in case of detection of any abnormality in this data, an alert is generated and sent to the responsible health professional. It can be easily integrated into any medical monitoring equipment through a Web Service interface, which provides access through the computer network and internet using the SOAP (Simple Object Access Protocol) communication protocol . Through this access interface, the system receives from the monitoring equipment a message identifying what is being monitored and the monitoring status that corresponds to the normal or abnormal stage (when an abnormality is detected). It is also possible to send the name of the detected abnormality. In this way, several advantages are added to the system, such as communication transparency, security and robustness.

The system can also be generated for monitoring only one signal, depending on the restrictions and needs of the location that it will be installed. For the warning alert it is possible to choose among three options: Onscreen display, audible warning or even sending the message to the doctor's mobile device.

And finally, the system provides a mechanism to manage the generated alerts by selecting one of two ways: One for simpler hardware with less processing power and another one for more complex and powerful hardware. The first consists of a data list using FIFO (First In First Out) that keep sending alerts as soon as new ones are received. The second consists of a priority list that before submitting, it checks if there are other warnings, and depending on the alert priority, the queue is reordered. Fig 1 presents an overview of the system operation.

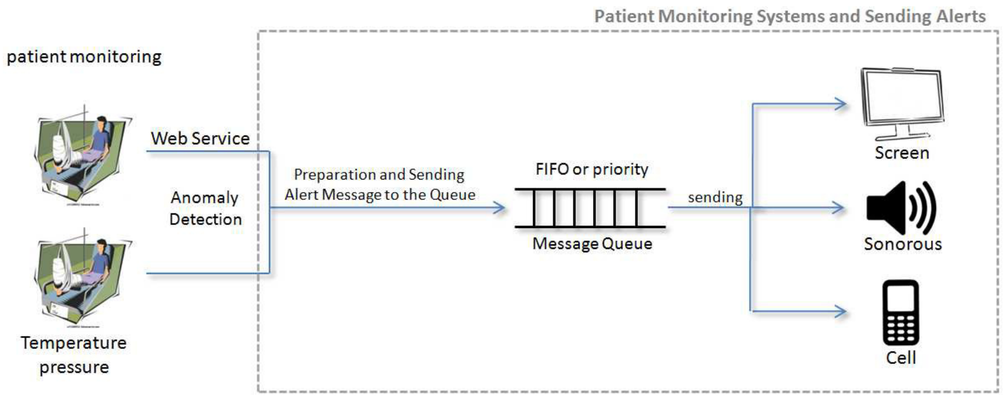


Fig. 1. System Overview

Thus, a feature model based on the Czarnecki model [6] was used for a better view of the system that will be generated by the Software Product Line. This model allows listing all the features present in the system, and their dependencies, as shown in the Fig 2.

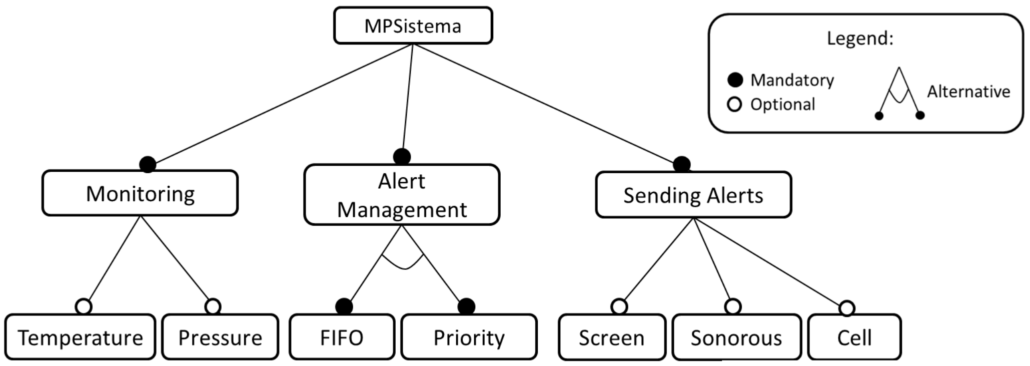


Fig. 2. Features Model for Case Study

The first feature of the system corresponds to the monitoring which is responsible for querying and analyzing the data related to the health condition of a patient in the hospital. Thus, it contains two optional features, the Temperature and Pressure. In the system feature selection, the user can choose to have only temperature monitoring, only pressure monitoring, or even both in parallel.

The second feature corresponds to alert management, which stores the alerts in a row, scheduling them based on a FIFO queue or on a priority list, and after that it makes the alerts available for sending. Thus, there are two types of schedulers, FIFO and priority, which are mutually exclusive, which means the user can only select one of the two.

And for the third feature, it corresponds to the sending of alerts, which can be made to three devices: A speaker, the screen of a computer, or even a mobile device. These three options correspond to optional features, in which the user has the freedom to choose which items to use.

Systems that are part of the real-time domain demand major concerns in the selection of related features, because besides common features, temporal issues of the system must be taken in consideration, such as temporal parametrization and process concurrency in the system.

In the context of Software Product Lines, the generation of products is carried out with the aid of instantiation tools, which are responsible for automating the process of derivation of systems based on the feature model. [7] Thus, there are several tools in the literature, including: GenArch, pure::variants and Captor [8].

GenArch is a tool based on three models: feature, architecture and configuration. Despite being a very efficient tool in the derivation of products, you must have a good knowledge prior to its use on the Java programming language, the Eclipse development environment and its plugins (including the FMP - Feature Modeling Plugin).

Pure::variants is also a tool for model-based derivation, and its approach considers two main models: feature and family models. It is a complex tool, heavy and is not widely used by developers unfamiliar with the area.

Finally, Captor the chosen tool is a Configurable Application Generator (CAG) that enables developers to configure and derive systems comprising the steps of Domain and Application Engineering. It also has a friendly user interface using forms at the time of instantiation of the system [9]. Another advantage of Captor is in variability mapping, which can be done directly in the application code, as well as in external application, by configuring the component templates and XSL (EXtensible Stylesheet Language) templates transformation [10].

By presenting a simple methodology for configuration and system mapping, the Captor tool was used in the development of the Product Line. This tool eases the creation of real-time systems as it allows full configuration of its variability, as it uses forms for the creation and configuration of the new system.

2.1. Methods: System Preparation

The first concern is the preparation of variabilities that are based on the parameter and number of tasks. Such variability can produce one or more tasks on the final system that can be varied according their temporal parameters. In the case study, is the case of features that represent the selection of the type of monitoring, in which the user can choose temperature or pressure, and, shortly thereafter, pass values to execution.

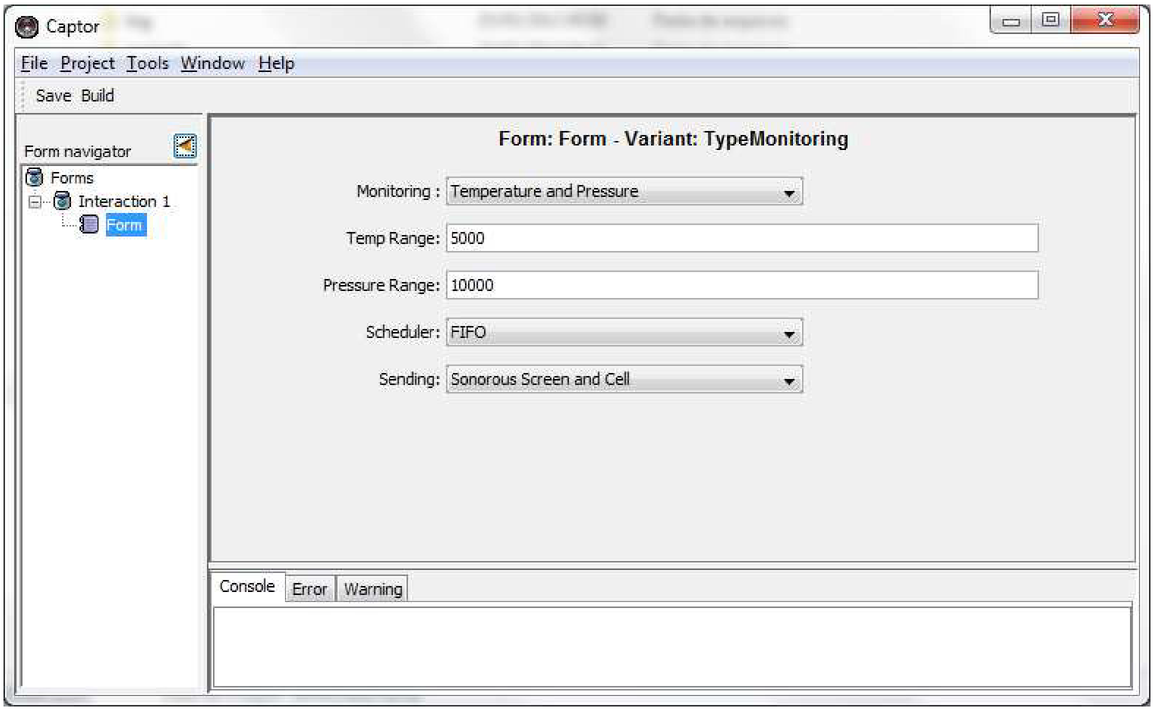
The Captor uses an Application Modeling Language (AML) based on form filling as a way of developing domain [11][12].

Captor's graphical user interface is configured to display the changeable behaviors of systems that can be generated, and save them to an XML (Extensible Markup Language) file. Inicially, must be configured a set of files responsible for transforming the specification in the forms into software artifacts called templates. These templates should be built using the XSL (Extensible Stylesheet Language) template language.

For each domain, a template transformation mapping file called MTL (Mapping Transformation Language) must be created. In the code, the template language (XSLT - eXtensible Stylesheet Language Transformation) is used for configuration and modification of the related parts to the configured domains. The concern in the configuration code is in the generation of the corresponding processes, modifying the variables that represent the time constraints, and generating custom code and ready for operation.

In the code, the mechanisms of the real-time library RTSJ (Real-Time Specification for Java) are used for task synchronization, which allows blocking features and functionality when accessed by a task, allowing access only when this task finishes processing [13].

These synchronization mechanisms are important for use in a product line, because the amount of tasks that will be generated and how many of them will run in parallel and accessing common resources is undefined, depending mainly on the selection of features that the system contains.
Thereafter, there is a form configuration phase. The screen displayed by Captor is shown in Fig 3.



Form: Form - Variant: TypeMonitoring

Monitoring : Temperature and Pressure
Temp Range: 5000
Pressure Range: 10000
Scheduler: FIFO
Sending: Sonorous Screen and Cell

Fig. 3. Captor selection screen of system features

The first feature selection matches the kind of monitoring that will be performed. It is possible to choose the monitoring option between temperature, pressure, or even both. At the end, just select the generation system option in Captor, then it will be ready to run on the computer with all the selected features and elaborated restrictions, allowing it do work properly.

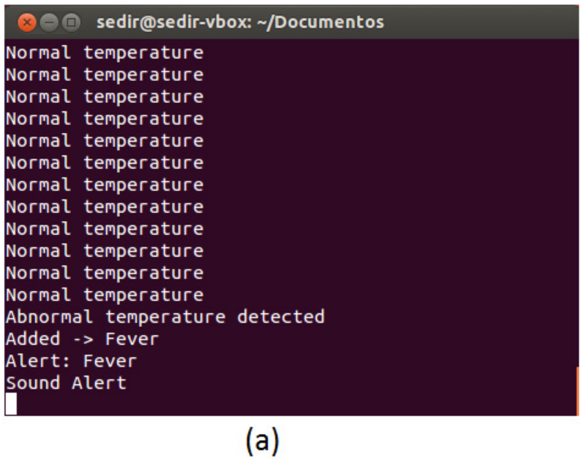
3. Tests and Results

Two tests were made in order to validate the tool. A controlled test environment was used, in which the values of monitoring and identification of abnormalities had been previously defined and configured on a system developed and integrated into the tool generated by the Software Product Line, simulating a medical equipment for patient monitoring in a ICU at a hospital. Tests were performed on the Ubuntu 10.10 GNU/Linux Operating System, with the Real Time Specification library for Java installed.

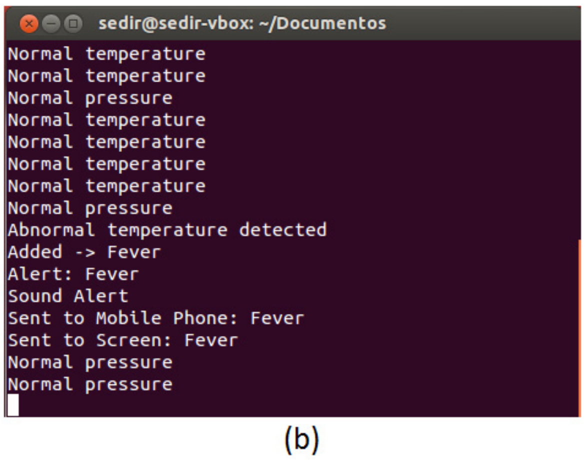
The first one consisted by selecting only some features of the system such as temperature monitoring, priority type scheduler and the alert as audible only. In this one, the developed system for testing was responsible for monitoring only the temperature of a patient by sending to the generated tool a message that the patient had a normal temperature every thirty (30) seconds. The system was programmed to detect and send an abnormality after five (5) minutes of monitoring, identifying fever in the patient.

The second test was more comprehensive by selecting various features, increasing thus, the level of concurrency in the system. The following features were selected in this last test: Monitoring of temperature and pressure, the FIFO scheduler, and all three systems alerts. In this test, the monitoring system simulated temperature and pressure checking of the patient every thirty (30) seconds. After five (5) minutes, an abnormal message was sent, representing the detection of fever in the patient.

In Fig 4 (a), the resulting system log from the first test execution is represented, and the Fig 4 (b) shows the system log from the second test.



```
Normal temperature
Normal temperature
Normal temperature
Normal temperature
Normal temperature
Normal temperature
Normal temperature
Normal temperature
Normal temperature
Normal temperature
Normal temperature
Normal temperature
Abnormal temperature detected
Added -> Fever
Alert: Fever
Sound Alert
```



```
Normal temperature
Normal temperature
Normal pressure
Normal temperature
Normal temperature
Normal temperature
Normal temperature
Normal pressure
Abnormal temperature detected
Added -> Fever
Alert: Fever
Sound Alert
Sent to Mobile Phone: Fever
Sent to Screen: Fever
Normal pressure
Normal pressure
```

Fig. 4. Captor selection screen of system features

Results show that the system generated through the selection of features was executed normally without any errors, and all the functionalities of both the generated systems worked as expected, demonstrating that the tools were effective for development.

4. Discussion

A simple Patient Monitoring System can be generated using a software product line by following the steps mentioned in this paper. In case the system presents greater complexity, the project must use new mechanisms available for real-time machine or even the programming language, such as new scheduling algorithms, task priorities, or even managing shared memory computer.

Captor has proven to be an efficient tool for the preparation of Product Line, since it reaches all stages of development, make use of forms in the assembly and selection of system features (such structures are present in the daily lives of users who will use the system) and also provide all the mechanisms for systems generation.

5. Conclusion

This work made possible the development and generation of a Software Product Line for Patient Monitoring System in practice, allowing the analysis and obtainment of experimental results about the used domain.

The preparation of the SPL made it easier to develop systems for a hospital environment, since it abstracts this early stage and allows a health care professional to play the role of software assembler, and then generate new systems by selecting desired features. This is very important, as professionals in this field have little knowledge related to system development and now they can generate new systems in a much shorter time than would professional software developers.

The system used as a case study is an important solution for patient monitoring environments in hospitals, since they require efficient and variable mechanisms to notify the medical team in charge, allowing, thus, to be possible the decision-making in an appropriate skillful time. Software Product Lines facilitate the generation and use of these systems in hospitals, due to the wide variety of the equipment used in monitoring.

As future works, test the system in a real world production environment by health professionals and after that, apply question forms about the needs fulfilled or not in the context of the created SPL. In this test, we intend to perform system integration with medical equipment located in the ICU (Intensive Care Unit) of the Onofre Lopes University Hospital, so that it can generate alert messages for real patients.

Acknowledgements

The authors thank the Laboratory for Technological Innovation in Healthcare (LAIS), the Onofre Lopes University Hospital (HUOL), and the Federal University of Rio Grande do Norte (UFRN), for providing proper environment and facilities to execute this research.

References

[1] Gurp JV, Bosch J, Svahnberg M. On the Notion of Variability in Software Product Lines". In WICSA '01: Proceedings of the Working IEEE/IFIP Conference on Software Architecture. Amsterdam, Netherlands; 2001.
[2] Prieto-Diaz R. Classification of reusable modules, in Software Reusability: Concepts and Models, T.J. Biggerstaff and A.J. Perlis, Editors. Addison-Wesley Pub. Co.: New York, NY; 1989. 99-123.
[3] Clements P, Northrop L. Software Product Lines: Practices and Patterns. Boston, MA, USA: Addison-Wesley Longman Publishing; 2001.
[4] Brandt SA, Banachowski S, Caixue L, Bisson T. Dynamic integrated scheduling of hard real-time, soft real-time, and non-real-time processes. In RTSS 2003: 24th IEEE Real-Time Systems Symposium. Cancun, Mexic; 2003.
[5] Krishna, C. M. Real-Time Systems. Wiley Encyclopedia of Electrical and Electronics Engineering. 1999.
[6] Czarnecki K. Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models. Technical University of Ilmenau. 1998.
[7] Sybren D, Marco S, Jan B. Product derivation in software product families: a case study. Journal of Systems and Software 2005; 74, 2; 173-194
[8] Torres M, Kulesza U, Sousa M, Batista T, Teixeira L, Borba P, Cirilo E, Lucena C, Braga R, Masiero P. Assessment of product derivation tools in the evolution of software product lines: an empirical study. In Proceedings of the 2nd International Workshop on Feature-Oriented Software Development. New York; 2010. 10-17.
[9] Shimabukuro EK. Um Gerador de aplicações configurável. Dissertação de Mestrado, ICMC/USP, São Carlos, SP; 2006.
[10] W3C. The Extensible Stylesheet Language Family
[11] Pereira Junior CAF, Braga RTV, Masieiro PC. Captor-ao: Gerador de aplicações apoiado pela programação orientada a aspectos. In: Anais da Seção de Ferramentas do II Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS'2008), Porto Alegre, Brazil; 2008. 1–8.
[12] Shimabukuro EK, Masiero P, Braga R. Captor: um gerador de aplicações configurável. In: XX Simpósio Brasileiro de Engenharia de Software. Florianópolis, SC, Brazil; 2006.
[13] Bollella, G; Gosling, J. The real-time specification for Java. Computer 2000,33,47-54.