Productivity Evaluation of Self-Adaptive Software Model Driven Architecture

Basel Magableh, Trinity College Dublin, Ireland
Stephen Barrett, Trinity College Dublin, Ireland

ABSTRACT
Anticipating context changes using a model-based approach requires a formal procedure for analysing and modelling context-dependent functionality and stable description of the architecture which supports dynamic decision-making and architecture evolution. This article demonstrates the capabilities of the context-oriented component-based application-model-driven architecture (COCA-MDA) to support the development of self-adaptive applications; the authors describe a state-of-the-art case study and evaluate the development effort involved in adopting the COCA-MDA in constructing the application. An intensive analysis of the application requirements simplified the process of modelling the application’s behavioural model; therefore, instead of modelling several variation models, the developers modelled an extra-functionality model. COCA-MDA reduces the development effort because it maintains a clear separation of concerns and employs a decomposition mechanism to produce a context-oriented component model which decouples the applications’ core functionality from the context-dependent functionality. Estimating the MDA approach’s productivity can help the software developers select the best MDA-based methodology from the available solutions. Thus, counting the source line of code is not adequate for evaluating the development effort of the MDA-based methodology. Quantifying the maintenance adjustment factor of the new, adapted, and reused code is a better estimate of the development effort of the MDA approaches.

Keywords: Constructive Cost Model II, Context-Dependent Behavioural Variations, Context-Oriented Component-Based Application-Model-Driven Architecture (COCA-MDA), Model Driven Architecture (MDA), Self-Adaptive Software

INTRODUCTION
Mobile computing environments are heterogeneous and dynamic. Everything from the devices used and resources available to network bandwidths and user context can change drastically at runtime (Belaramani, Wang, & Lau, 2003). This presents the software developers with the challenge of tailoring behavioural variations both to each specific user need and to the context information. Context-dependent behavioural variations can be seen as a collaboration of individual features expressed in requirements, design, and implementation. Before encapsulating the crosscutting context-dependent behaviours into a software module,
the developers must first identify them both in the requirements document and in the software model. This is difficult to achieve because, by their nature, context-dependent behaviours are entangled with other behaviours, and are likely to be included in multiple parts (scattered) of the software modules. Using intuition or even domain knowledge is not necessarily sufficient for identifying the behavioural variations; instead, it requires a formal analysis procedure for the software requirements and a separation of their individual concerns. Moreover, a formal procedure for modelling these variations is needed. This kind of analysis and modelling procedure can reduce the complexity in modelling self-adaptive applications and encapsulate the context-dependent part of the distinct architecture module (component).

In this sense, a context oriented component model (Coca-component) (Magableh & Barrett, 2009) is used to encapsulate behavioural variations and decouple them from the application’s core functionality. In this way, dynamic component composition is achieved. Additionally, from the software developer’s perspective, it is vital to know the productivity of the development paradigm which might be used in constructing the self-adaptive application. Productivity evaluation of model-driven approaches can assist the developers in selecting among the proposed methodologies in the literature which approach dynamic behavioural variations of self-adaptive software. Context-oriented component-based application-model-driven architecture (Coca-MDA) emerged as a development paradigm which facilitates the development of self-adaptive context-oriented software (Magableh & Barrett, 2011b, 2011c).

This article evaluates the development effort involved in adopting Coca-MDA when constructing a self-adaptive application for an indoor wayfinding application (IWayfinder) targeting individuals with cognitive impairments. The development effort of Coca-MDA is compared to other model-driven approaches proposed in the literature.

The remainder of the article is structured as follows. The next section provides a comparative analysis of related studies. A case study of a self-adaptive application is then demonstrated. The Coca-MDA phases are also described. The following section provides a Coca-MDA evaluation using constructive cost model II (COCOMO II). The last section summarizes the research findings and describes directions for future work.

RELATED WORK

In the literature, there are several MDA approaches which target the development of self-adaptive applications for mobile computing environments which produce component-based applications; this study borrows from the following methodologies: MUSIC, proposed by Wagner, Reichle, Khan, and Geihs (2011); U-MUSIC (Khan, 2010); and PasPallis MDA (PasPallis, 2009).

The MUSIC development methodology (Wagner et al., 2011) adopts a model-driven approach to constructing the application variability model. The applications are built as a component framework with component types as variation points. Middleware is used to resolve the variation points, which involves the election of a concrete component as a realization of the component type. Using this method, a number of application variants can automatically be derived.

The U-MUSIC methodology, proposed by Khan (2010), adopts the model-driven approach to constructing self-adaptive applications and enabling dynamic unanticipated adaptation based on a component model. The U-MUSIC system enables the developers to specify the application variability model, context elements, and data structure. The developers are able to model the component functionalities and quality of service (QoS) properties using an abstract, platform-independent model (PIM).

PasPallis (2009) proposes another MDA-based methodology which considers the context providers for the application. For each context provider, a plug-in is proposed during the design phase. At runtime, a utility function is
E-Portfolio to Promote the Virtual Learning Group Communities on the Grid
[www.igi-global.com/article/portfolio-promote-virtual-learning-group/2606?camid=4v1a](www.igi-global.com/article/portfolio-promote-virtual-learning-group/2606?camid=4v1a)

Experiences in Developing a Micro-payment System for Peer-to-Peer Networks
[www.igi-global.com/article/experiences-developing-micro-payment-system/41726?camid=4v1a](www.igi-global.com/article/experiences-developing-micro-payment-system/41726?camid=4v1a)