A Software Product Line Architecture for Distributed Real-time and Embedded Systems: A Separation of Concerns Approach

Shih-Hsi Liu
Department of Computer and Information Sciences, University of Alabama at Birmingham
Birmingham, AL, 35294, U.S.A.
{liush}@cis.uab.edu

ABSTRACT
This paper presents a separation of concerns approach to solve the tangling problem of functional and Quality of Service (QoS) concerns in traditional Component-based Software Engineering (CBSE) and Software Product Line (SPL) technologies applied to Distributed Real-time and Embedded (DRE) systems. This problem originates from the interchangeability for fulfilling functional and QoS concerns during composition. The approach utilizes the perspective of QoS to design and analyze a set of software systems represented by a collection of QoS systemic paths, which determine how well functional tasks perform in terms of flows of application-specific and functionality-determined information between components. Our approach not only reserves the virtues of reusability, changeability, productivity and expeditiousness that traditional CBSE and SPL technologies possess, but also dedicates the contributions in terms of separation of concerns, design space exploration, fine-grained commonality and reusability evaluation and less subjective feasibility analyses for a component-based SPL.

Categories and Subject Descriptors
C.3 [Special-purpose and Application-based Systems]: Real-time and embedded systems.; D.2.13 [Software Engineering]: Reusable Software; D.3.1 [Programming Languages]: Formal Definitions and Theory

General Terms
Design, Languages.

Keywords
Software Product Line Architecture, Quality of Service, Two-Level Grammar++, UniFrame, Real-Time.

1. PROBLEM STATEMENTS
Component-Based Software Engineering (CBSE) technologies benefit software development by reusing and replacing components as needed. Software Product Line (SPL) technologies enrich the reusability merit of CBSE by analyzing a specific domain and constructing a set of software products that share the commonality of particular features. The integration of CBSE and software product line concepts expedites the pace of software development and proliferates the productivity of software products. Recent CBSE technologies have involved separation of functional and QoS concerns that facilitates component reusability at the component abstraction level [2]. For DRE systems, however, QoS transcends functional properties to include non-functional (i.e., QoS) properties such as real-time issues and plays a critical role in determining the quality of systems. It reduces the virtues of CBSE and software product line technologies and raises new problems in assembling a system from components, namely: (a) the component perspective of CBSE reveals difficulties in evaluating abundant QoS properties possessed in many DRE systems (i.e., QoS-sensitive systems), (b) numerous design alternatives will be generated by composition and permutation decisions among components, (c) composition semantics influencing the magnitude of QoS performance still lack appropriate specifications, and (d) all QoS properties and the margins of QoS satisfaction require less subjective evaluation at system assembly time.

2. PROPOSED APPROACH
This paper presents a separation of concerns approach for a component-based DRE SPL in the vision of the UniFrame project [6]. By separating functional and non-functional concerns in the requirements and design phases and using the perspective of QoS, a DRE system is regarded as a collection of QoS systemic paths [7]. Such paths determine how well functional tasks perform in terms of flows of application-specific and functionality-determined information between components. The following paragraphs show the procedures of the proposed approach. Domain analysis: As a SPL involves thousands of QoS requirements to be evaluated (e.g., an avionics SPL), it is inevitable to explore the common features among software products and to eliminate infeasible ones in terms of the requirements. The QoS perspective approach regarding domain analysis utilizes Two-Level Grammar++ (TLG++) [1] as a representation and evaluation notation [5]. TLG++ is an object-oriented specification language that consists of two Context Free Grammars (CFGs) defining the set of parameters and the set of function definitions over the parameters, respectively. For the first CFG, a set of production rules specifies the flows of the QoS systemic paths of one QoS requirement (i.e., a QoS systemic path family). In such a case, a syntax tree derived from the first CFG represents an alter-
native of a QoS systemic path family containing the stream directions and involved components. Parsing all possible syntax trees of a QoS requirement facilitates commonality and variability analyses in terms of the QoS requirement at the QoS systemic path abstraction level. The commonality and variability analyses among different QoS requirements (i.e., systematical abstraction level) can be examined by a meta-symbol table consisting of the knowledge of the symbol tables derived from all of the first CFGs of QoS requirements. The graphical information of systematical common and variable features can be also captured in the next phase.

In the second CFG, semantics describes the margins of QoS satisfaction and the composition influences regarding the QoS requirement of the QoS systemic path family. The margins of QoS satisfaction serve as selection metrics while collecting satisfactory QoS systemic paths for constructing a DRE SPL. The composition influences state the possible upgrade or degradation caused by glue or wrapper code. A Jess rule and knowledge base [3] are involved to offer an inference mechanism to reason about the feasibility of composition in terms of QoS requirements and component dependencies.

A Colored Petri Net-Based Modeling Approach: The suitable QoS systemic paths are accumulated and rendered by a Colored Petri Net (CPN)-based modeling approach, called QoS-UniFrame [4], in the design phase. CPNs are formalisms similar to dataflow analysis, but comprise extra abstractions beneficial in simulating concurrent and asynchronous systems. The Generic Modeling Environment (GME)\(^1\), which consists of metamodel, model and interpreter levels, is the modeling environment chosen. At the metamodel level, a modeling paradigm specifies the abstractions and corresponding behaviors of CPNs. Software engineers depict flows of satisfactory QoS systemic paths to simulate the behaviors of a DRE SPL in compliance with the modeling paradigm at the model level. At the interpreter level, three dimensions are considered: design space exploration and elimination and systematical commonality and variability analyses for the DRE SPL and systematical QoS requirement evaluations for individual DRE systems. Design space exploration and elimination diagnoses the feasibility of software products (i.e., alternatives) of the DRE SPL based on CPN’s reachability tree. Systematical commonality and variability analyses offer the additional SPL information at the design stage. After design space exploration and elimination, systematical QoS requirement evaluations for feasible alternatives are processed. The survival alternatives not only share the common functional features, but also achieve the QoS satisfaction with different margins.

3. OBJECTIVES

As DRE systems are omni-present, traditional component-based software engineering and SPL technologies are insufficient. The novel QoS systemic path perspective and granularity facilitates the design and evaluation of DRE systems in consideration of all kinds of QoS properties. They also assist in analyzing a domain for common features, constructing a SPL for productivity, and optimizing QoS properties for quality guarantees. In addition to the virtues of productivity, reusability, changeability and expeditiousness that CBSE and SPL techniques possess, the proposed approach will dedicate four contributions to the DRE community: (a) the separation of concerns solves the problem of tangling functional and non-functional concerns in CBSE and SPL that reveal difficulties in designing and evaluating QoS-sensitive systems, (b) the design space exploration and elimination reduces the design space, (c) QoS-driven domain analysis captures fine-grained common features and solves the QoS composition semantics problem, and (d) the different QoS satisfactory margins of feasible alternatives are evaluated by a less subjective approach and hence offer selectivity for software engineers to construct various software systems under application-specific circumstances.

4. STATUS AND FUTURE WORK

The prototypes of the domain analysis and modeling approaches have been constructed. For domain analysis, our prototype is capable of analyzing the satisfactions of QoS systemic paths defined by CFGs in TLG++. A DRE system can be designed and analyzed by simulating flows of QoS systemic paths in QoS-UniFrame. Currently, our research concentrates on solving the scalability and commonality analysis issues in both phases. This research should be validated via constructing a DRE SPL. After construction, an evaluation process should be applied to verify if all software products of the DRE SPL satisfy all QoS requirements and if the optimal alternative is within the DRE SPL.

5. ACKNOWLEDGEMENTS

This research is mentored by Dr. Barrett R. Bryant.

6. REFERENCES