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

Shih-Hsi “Alex” Liu, Barrett R. Bryant, Jeff Gray
University of California at Los Angeles (luis, bryant, gray)@ics.ucla.edu

Mikhail Auguston
Naval Postgraduate School
mauguston@nps.navy.mil

Project Objective
This project presents a novel software product line architecture for component-based Distributed Real-time and Embedded (DRE) systems. The project concentrates on the phases of domain engineering and application engineering to achieve the following objectives:

- Every member of a software product line satisfies its functional and non-functional requirements synergistically.
- Every member possesses an architectural design.
- All members of a software product line share a number of common features.
- Members possessing various margins of QoS satisfaction are differentiated by variable features.
- Each member satisfies its functional and non-functional requirements at requirements and design workflows.

Key Challenges

- Challenge 1: QoS sensitive
  DRE systems are sensitive to the availability of system resources, which directly or indirectly affect the QoS properties of the system. The magnitudes of such properties influence the feasibility and performance of a DRE system. More precise and less subjective QoS property measurements are required.

- Challenge 2: Component Composition
  Evaluation after composition of hundreds of QoS properties requires satisfaction. It is difficult for the QoS tuning approach to balance and obtain the optimal solution after system composition. In addition, effort is wasted on many infeasible design alternatives after composition.

- Challenge 3: Abundant alternatives
  Abundant design alternatives generated from the combination and permutation of selected components are infeasible in terms of functional and non-functional requirements.

- Challenge 4: Costly DRE systems
  Many DRE systems are costly and hard to modify. A software product line, which consists of a set of software products sharing common features, will solve the problem.

Key Contributions

- The DRE software product line constructed by the project possesses three major contributions:
  - The advantages of applying component-based software engineering and product line engineering to software development.
  - The feasibility design alternatives are pruned off, which reduces the extra workload stated.
  - Each member satisfies its functional and non-functional requirements at requirements and design workflows.

Quality of Service (QoS)
- Functional Path: Flows of application-specific and functionality-driven information between components
- QoS Systematic Method: determines how well a functional path behaves in terms of a specific QoS property of a QoS systemic path
- QoS Classification:
  - Static: parameters are design-related.
  - Dynamic: parameters are influenced by the deployment environment.
  - Strict: parameters must satisfy requirements.
  - Non-strict: parameters allow margins of error when meeting requirements.
  - Orthogonal: two parameters have no mutual effects regarding a specific resource.
  - Non-orthogonal: two parameters have mutual influence regarding a specific resource.

Domain Engineering
- Select DRE components by functional and non-functional requirements (not the core procedure in the post).
- Analyze the combinability, variability, and satisfaction of QoS systemic path.
- Define a grammatical QoS-driven approach.
- The Java Rule Engine and Knowledge Rules facilitate the composition of components.
- Construct a set of software products that share common features by QoS-driven frame, a Context-driven Non-based approach.
- Stores the functional and non-functional requirements of each existing component, component dependencies, and composition rules.
- Component dependencies: the relationships between components in terms of function-determined and application-specific tasks.
- Composition rules: verify interface consistency between components and pre- and post-conditions of component interactions.
- QoS satisfaction formula: quantitatively estimate the satisfaction of the QoS property of a QoS systemic path.
- TLO++: as an Architecture Description Language (ADL), describes the reference architecture.

Application Engineering
- TLO++ syntactically and semantically expresses QoS systemic paths.
  - The first CFG defines rules for selecting a set of components.
  - The second CFG defines a set of function definitions.
  - The first CFG defines syntactic production rules.
  - The second CFG defines semantic production rules.
  - An example:
    - Entry: select QoS system with Sensor, Connect, Comp, Coord, and Present.
    - Query: if query then application of components with Sensor, Connect, Comp, Coord, and Present.

Mobile Augmented Reality Systems
- A DRE system concentrating on enriching the user environment by merging real and virtual objects
  - Six subsystems:
    - A Capture: performs specific functionalities for the application
    - A Presentation: exhibits virtual multimedia objects
    - A Tracking and registration: tracks user’s position and orientation and registers virtual objects
    - An Environment model: stores the geometrical and detailed hierarchical 3D information
    - Interaction: a user friendly interface for input and output
    - Visualization: provides mobile communications
  - Examples:
    - Battlefield training system (shown at right)

A Case Study: The Battlefield Training System
- The Battlefield Training System (BTS) assists in training soldiers in different scenarios, strategies, and battles.
- The BTS consists of:
  - Real objects: buildings and obstacles in the battlefield
  - Virtual objects: enemies and a hostage displayed on a Head Mounted Display (HMD)
  - Sensors/Trackers: track the position and orientation of the soldiers
  - Scenario: rescue the hostage from the enemy

- The advantages of BTS:
  - Adaptable scenarios: trains soldier to react and respond property in different exercises.
  - Less cost: simulates highly cost battlefield devices (e.g., tanks and weapons).
  - Less wounded: reduces the possibilities that soldiers being wounded in the real battlefield.

Background
- Two-Level Grammar++ (TLO++)
- A formalism beneficial in modeling concurrent and asynchronous systems.
- Transition: determines what, when and how QoS parameters are to be processed with associated predicates and functions for time, priorities and event triggers.
- Arc: control the timing direction of QoS parameters.
- QoS parameter: consists of priority, type, and range.
- Event: triggers transition.
- Time: transition is triggered at specific time.
- Place: represents a component in a DRE system.

Timed Colored Petri Nets (TCPNs)
- A formalism beneficial in modeling concurrent and asynchronous systems.
- Transition: determines what, when and how QoS parameters are to be processed with associated predicates and functions for time, priorities and event triggers.
- Arc: control the timing direction of QoS parameters.
- QoS parameter: consists of priority, type, and range.
- Event: triggers transition.
- Time: transition is triggered at specific time.
- Place: represents a component in a DRE system.

Application Engineering
- TLO++ syntactically and semantically expresses QoS systemic paths.
  - The first CFG defines rules for selecting a set of components.
  - The second CFG defines a set of function definitions.
  - The first CFG defines syntactic production rules.
  - The second CFG defines semantic production rules.
  - An example:
    - Entry: select QoS system with Sensor, Connect, Comp, Coord, and Present.
    - Query: if query then application of components with Sensor, Connect, Comp, Coord, and Present.

The Separation of Concerns Approach

An Overview
- Select DRE components by functional and non-functional requirements (not the core procedure in the post).