QoSPL: A QoS-Driven Software Product Line Engineering Framework for Distributed Real-time and Embedded Systems

Shih-Hsi “Alex” Liu¹, Barrett R. Bryant¹, Jeff Gray¹, Rajeev Raje², Mihran Tuceryan², Andrew Olson², and Mikhail Auguston³

1. University of Alabama at Birmingham
2. Indiana University-Purdue University Indianapolis
3. Naval Postgraduate School
Outline

- Problem Statements
- Background: UniFrame
- QoSPL: a QoS-driven Product Line framework
  - A Case Study: a battlefield training system
  - Domain Engineering (a TLG++ approach)
  - Application Engineering (a Petri Net approach)
- Related Work
- Conclusion
Problem Statements

Requirements Tangling: Func. and nonfunc. req. require evaluation interchangeably

Abundant Alternatives: Numerous design alternatives generated

QoS Sensitive: QoS Satisfaction affects the correctness and performance of systems
Overview of the QoSPL

Select DRE components by their spec.

Analyze the commonality, variability, and satisfaction of QoS paths by the TLG++ approach

Component Selection

Stores the functional and nonfunctional spec. of existing components, component dependencies, and composition rules

Construct a set of software products that share common features by QoS-UniFrame

A Rule Engine and Knowledge Base

Including facts, queries and rules for inferring component composition

Domain Engineering

Application Engineering

Overview of the QoSPL
Mobile Augmented Reality Systems

- A DRE system concentrating on enriching the user environment by merging *real* and *virtual* objects

- Six subsystems:
  - **Computation**: performs specific functionalities for the application
  - **Presentation**: computes virtual multimedia objects
  - **Tracking and registration**: tracks user’s position and orientation and registers virtual objects
  - **Environmental model**: store the geometrical and detailed hierarchical 3D information
  - **Interaction**: coordinates virtual multimedia objects
  - **Wireless communication**: provides mobile communications
A Case Study: A battlefield training system

Interaction subsystem shows text and visual information
A battlefield strategist/trainer assigns strategies

The view displayed by HMD

Send/Receive Trackers Signals

GPS PDA

GPS Satellites

Geographical Model

Presentation subsystem

Wireless Devices

Computation subsystem

Tracking and Registration subsystem

Send strategies

The Battlefield

Interaction subsystem shows text and visual information
A battlefield strategist/trainer assigns strategies

Send strategies

The Battlefield

Interaction subsystem shows text and visual information
A battlefield strategist/trainer assigns strategies
A soldier is to rescue a virtual hostage in a battlefield. The position and orientation sensors on his body send back the 6 Degrees Of Freedom (6DOF) data to the tracking subsystem every half second via wireless communication. As the soldier is standing on specific positions with specific orientations in some buildings derived from a predefined tactical scenario stored in the computation subsystem, his HMD displays the enemies registered by the tracking system, rendered by the presentation subsystem and coordinated by the interaction subsystem. The soldier communicates with the command center via his headphone. The information of the soldier's current position is displayed on the HMD by text.
Domain Engineering: The Two-Level Grammar++ approach

- **Goals:**
  - Analyze common and variable requirements
  - Design a prescribed reference architecture for its product line
  - Implement a set of reusable core assets
  - Verify and validate the core assets
Domain Engineering: The Two-Level Grammar++ approach

Original TLG++

- Two Context Free Grammars (CFGs):
  - The 1\textsuperscript{st} CFG: Define a set of Parameters
  - The 2\textsuperscript{nd} CFG: Define a set of Function Definitions

- A Grammatical Concept:
  - Define Syntax and Semantics of Programming Languages
    - The 1\textsuperscript{st} CFG: Define the Syntax by Production Rules
    - The 2\textsuperscript{nd} CFG: Define the Semantics of the Production Rules
QoS-driven TLG++

- QoS-driven TLG++ specifies and analyzes QoS paths at the service level

- The first CFG utilizes Extended Backus-Naur Form (EBNF) to define the components and direction of a QoS systemic path
  - EBNF represents *mandatory, alternative, optional,* and *OR* features
class TextDeadlineFromClient.

Syntax :: Sensor WC TkP.
Sensor :: OS PS ; PS ; OS.
PS :: ps1 ; ps2 ; ps3.
OS :: os1 ; os2 ; os3 ; os4.
WC :: wc1 ; wc2.
TkP :: trackProcessing.
Sum :: Double.

semantics of sendPositionFromClient :
//coming up……..

end class.

Domain Engineering
The second CFG defines component dependencies, composition rules, and QoS satisfaction formula.

- 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 composition by inferences.
- QoS satisfaction formula: quantitatively estimate the satisfaction of the QoS property of a QoS systemic path.
class TextDeadlineFromClient.
Syntax ::= Sensor WC TkP.
....

semantics of sendPositionFromClient :
PreCondition ::= semantics of queryComponent with OS PS WC and TkP,
if PreCondition then
    Sum ::= semantics of sumOfMTAT with OS PS WC and TkP,
else ErrorMessage, end if,
PostCondition ::= semantics of queryPattern with Sum.
Double semantics of sumOfMTAT with OS PS WC and TkP :
    return OS semantics of getMTAT + ......./[SEKE’05]
end class.

Pre-Condition and Post-Condition can be used for validating QoS analysis and eliminating infeasible ones

Symbol tables are utilized for analyzing the rate of commonality and reusability of QoS systemic path families.

Intuition: More satisfactory QoS paths have higher probabilities to be selected.

- [ps2, wc2, tkp]
- [os1, ps2, wc2, tkp] or [os4, ps2, wc2, tkp]

<table>
<thead>
<tr>
<th>ID</th>
<th>os1</th>
<th>os2</th>
<th>os3</th>
<th>os4</th>
<th>ps1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>OS</td>
<td>OS</td>
<td>OS</td>
<td>OS</td>
<td>PS</td>
</tr>
<tr>
<td>MTAT (ms)</td>
<td>0.2</td>
<td>0.3</td>
<td>0.45</td>
<td>0.2</td>
<td>0.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>ps2</th>
<th>ps3</th>
<th>wc1</th>
<th>wc2</th>
<th>tkp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>PS</td>
<td>PS</td>
<td>WC</td>
<td>WC</td>
<td>Tkp</td>
</tr>
<tr>
<td>MTAT (ms)</td>
<td>0.15</td>
<td>0.2</td>
<td>0.5</td>
<td>0.4</td>
<td>0.9</td>
</tr>
</tbody>
</table>
Overview of the QoSPL

An Incremental-and-iterative lifecycle model

Domain Eng.

QoS-driven TLG++
Domain Analysis
ADL in TLG++
Domain Design
Glue/wrapper code
Domain Impl.

Common and variable features

Appl. Eng.

QoS-UniFrame
Appl. Analysis
TCPNs
Appl. Design
Glue/wrapper code
Appl. Impl.

A Software Product Line
Application Engineering: The Timed Colored Petri Nets Approach

Goals:

- Reuse core assets in each workflow in the domain engineering process
- Exploit variable features of each workflow on individual applications
- In our approach: assemble all common QoS paths and specific variable paths for different product line members
Timed Colored Petri Nets

Original Timed Colored Petri Nets (TCPNs)
- Places: states
- Transitions: actions or events
- Arcs: directions (P->T or T->P)
- Tokens: notations specifying configurations of places
- Weights: notations specifying configurations of transitions
- Colors: identities of places
- Time: timer
- Markings: configurations comprising tokens

QoS-driven Timed Colored Petri Nets
- Places: components or sub-components
- Transitions: design decisions (when, what, how) or method calls
- Arcs: directions (P->T or T->P)
- Tokens: QoS parameters or events
- Weights, Colors, Time, Markings: same as above
The diagram illustrates a system for tracking and communication in a battlefield scenario. It includes:

- **Trackers** connected to a **Soldier** wearing a **HMD** (Head-Mounted Display).
- A **GPS PDA** and a **Hand Tracker** are used for navigation and data collection.
- A **Rifle** is part of the tracking system.
- **GPS Satellites** provide location data.
- **Geographical Model** is used for situational awareness.
- **Presentation subsystem** displays information on a **Wireless Devices**.
- The **Computation subsystem** processes data and sends strategies.
- **Tracking and Registration subsystem** manages the system's interactions.

The view displayed by the HMD shows text and visual information, and an **Interaction subsystem** ensures smooth communication between the battlefield strategist/trainer and the soldiers.

A battlefield strategist/trainer assigns strategies and oversees the system's operations.
The reachability tree analysis

- Dynamic analysis: evaluate every intermediate node of the tree.
- Discard the entire branch if an intermediate node is not satisfied

- Marking

\[
\begin{align*}
M_0 & : (3,3,3,1,0,0,0,0,0,0,0,0,0,0,0) \\
M_1 & : (1,1,3,0,2,0,0,1,0,0,0,0,0,0,0) \\
M_2 & : (0,0,3,0,3,0,0,0,0,0,0,0,0,0,0) \\
M_3 & : (0,0,3,0,0,0,0,0,0,0,3,0,0,0,0) \\
M_4 & : (0,0,0,0,0,1,1,0,1,0,2,0,0,0,0) \\
M_5 & : (0,0,0,0,0,0,0,0,0,0,2,1,1,1,0) \\
M_6 & : (0,0,0,0,1,0,0,0,0,0,0,0,0,0,2) \\
M_7 & : (0,0,0,0,3,0,0,0,0,0,0,0,0,0,0) \\
M_8 & : (0,0,0,1,0,0,0,0,0,2,0,0,0,0,0)
\end{align*}
\]
Overview of the QoSPL

An Incremental-and-iterative lifecycle model

Domain Eng.

QoS-driven TLG++

ADL in TLG++

Glue/wrapper code

Domain Analysis
Domain Design
Domain Impl.

Common and variable features

Appl. Eng.

QoS-UniFrame

TCPNs

Glue/wrapper code

Appl. Analysis
Appl. Design
Appl. Impl.

A Software Product Line
Related Work

- FORM: extension of FODA
- KobrA: MDA+ CBSE + SPLE
- QADA: scenario based quality analysis
- Mini-Middleware: customized middleware based on QoS demands
Conclusion

- A separation of concerns approach to focus on QoS properties during the construction of DRE systems
- A DRE product line is assembled by combining different QoS paths
- Solve the three challenges inherent in DRE system construction using traditional CBSE and SPLE
  - QoS sensitive, Requirements tangling, Abundant alternatives
- Two formalisms to facilitate high confidence system construction
- TODO:
  - commonality and variability management utilized by the symbol table
  - Finer-grained QoS analysis
  - Implementation and testing workflows + ADL
Questions

More information
- UniFrame: www.cs.iupui.edu/uniFrame
- QoSPL: www.cis.uab.edu/liush/QoSPL.htm