QoS-UniFrame: A Petri Net-based Modeling Approach to Assure QoS Requirements of Distributed Real-time and Embedded Systems

Shih-Hsi Liu¹, Barrett R. Bryant¹, Jeffrey G. Gray¹, Rajeev R. Raje², Andrew M. Olson², and Mikhail Auguston³

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

Outline

- Motivation
- Challenge and characteristics
- QoS-UniFrame
- A case study
- Related work
- Conclusion
- Future work
Motivation

- Construction of Distributed Real-time and Embedded (DRE) Systems from scratch
  - time consuming
  - expensive
  - less modifiable
- A solution for constructing DRE systems: construct a DRE system by composing components from a repository
  - Reusability and Changeability

Challenge and Characteristics

- The Abundant Alternatives Problem
  - design decisions and permutation → abundant alternatives
- Quality of Service (QoS) Sensitive: pertains to the usage of system resources
  - QoS requirements (i.e., non-functional requirements): system resource specification
  - QoS parameters (i.e., QoS attributes): evaluation units for QoS requirements
  - QoS utility function: represents the measurement function for QoS req.
  - QoS constraints: limitation of system resources for QoS req.
- Unpredictable behaviors (e.g., glitches) occur in DRE systems, degrading the confidence of verification and validation
QoS-UniFrame: project objectives

- A QoS-Based semi-automatic design space exploration and analysis toolkit for constructing DRE systems at system assembly time
  - Explore possible alternatives based on different design and deployment decisions and permutation
  - Eliminate infeasible and less probable alternatives based on the evaluation of QoS requirements (i.e., the utility functions the corresponding constraints)
  - Assure feasible alternatives and obtain the optimal one based on the QoS utility functions
  - Evaluate the statistical results of alternatives as the references for substitutions when unpredictable behaviors occur
QoS-UniFrame

- QoS Parameter Classification
- Generic Modeling Environment (GME)
- Petri Nets
- AspectJ
- Backtracking and branch-and-bound algorithms
- Evolutionary Algorithms and PPCEA

![Diagram of QoS-UniFrame](image)
**QoS Parameter Classification**

- Static: parameters are design-related
- Dynamic: parameters are influenced by the deployment environment
- Strict: the system must satisfy the requirements regarding the strict QoS parameters
- Non-strict: the system allows margins of error when evaluating requirements regarding the non-strict parameters
- Orthogonal: two parameters have no mutual effects regarding specific resource
- Non-orthogonal: two parameters have mutual influence regarding specific resource
Generic Modeling Environment (GME)

- A metaconfigurable modeling tool that permits the customization of visual domain languages that are capable of code generation
- Metamodel: defines the modeling paradigm, including the syntax, semantics, constraints & presentation of the domain - Define the modeling paradigm of the Petri Net graph (defined later)
- Model: defines modeling case by the definitions of its metamodel - Define a Petri Net graph
- Interpreter: generates source code based on the model created by the modeler – Provide execution semantics for a Petri Net, including model analysis and reachability tree construction (defined later)
Petri Nets: A formalism beneficial in modeling concurrent and asynchronous systems (1/2)

Petri Nets: reachability tree (2/2)
AspectJ (1/2)

- AspectJ: an aspect-oriented extension to Java that enables modular implementation of crosscutting concerns
- Helps in identification of reachability tree nodes (through a pointcut), and defines the analyses methods on each node (through advice)
- Permits the isolation of constraint analyses in the nodes of the reachability tree: provide good modularity for analyzing individual QoS requirements
  - one QoS constraint analysis is written in an aspect
  - precedence of aspect decides the order of constraint analyses
  - reuse the reachability tree construction code: one applies to all
AspectJ (2/2)

Reachability Tree

- AspectJ’s Pointcut
- The aspect for the constraint analyses

Functional and Nonfunctional Specification

- A Use Case Scenario
- Static QoS: Strict & Orthogonal
- Static QoS: non-strict & non-orthogonal
- Dynamic QoS

- Petri Net Tree Constructor by Java
- Constraint Analysis by AspectJ Backtracking and Branch-and-bound
- PPCEA Interpreter
- Evolutionary Algorithms with PPCEA

- Discard - Strict Static QoS Requirements Do Not Meet
- Statistical Results of Non-strict Static QoS of Design Spaces

-PPCEA Code

Component Repository and Knowledge Base of UniFrame

The second level assurance

Statistical Results of Dynamic QoS of Design Spaces
Backtracking and branch-and-bound (B/B) algorithms (1/2)

Algorithms for the problems that search a set of solutions or ask for an optimal solution satisfying some constraint under the tree structure

<table>
<thead>
<tr>
<th></th>
<th>Backtracking</th>
<th>Branch-and-bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Algorithm</td>
<td>Depth-first</td>
<td>Breath-first search</td>
</tr>
<tr>
<td></td>
<td>Search</td>
<td>Least Cost search</td>
</tr>
<tr>
<td>QoS Utility Functions</td>
<td>Bounding</td>
<td>Bounding Function</td>
</tr>
<tr>
<td>QoS Constraints</td>
<td>Function</td>
<td></td>
</tr>
</tbody>
</table>

Backtracking and branch-and-bound (B/B) algorithms (2/2)

- **Dynamically** eliminates (i.e., stops generating) the infeasible nodes as the QoS utility function does not satisfy its constraint
- **Simultaneously** eliminates descendant leaves: because the QoS requirements are not satisfied, the following descendant leaves are not necessary to go through
Evolutionary Algorithms and PPCEA (1/3): the second level assurance

- Evolutionary Algorithms (EAs): a joint concept of computer science and Darwin’s theory of evolution
- Selection, recombination (e.g., crossover and mutation) and evaluation
- Obtain the optimal fitness values of fitness functions by exploring domain spaces under specific constraints
Evaluating dynamic QoS \textit{at system assembly time}:
- access the previous state information of a component in the knowledge base
- eliminate less probable assembled cases by statistics using PPCEA (next slide)
- statistical results of dynamic QoS of feasible alternatives may serve as excellent estimates and as substitutions as unpredictable behaviors occur at runtime.

Programmable Parameter Control for Evolutionary Algorithms (PPCEA): a domain-specific scripting language for Evolutionary Algorithms:
- Compute the statistical results for the fitness functions (e.g., best, average, worst and standard deviation of fitness values)
- \textit{Discarding policy} and the \textit{discard rate} decide the alternatives to be eliminated by statistics.
A Case Study: the water treatment plant (1/4)

3 Treatment Units (TUs) are selected

Each TU's battery life => 15Hrs

Total water treatment volume of selected TUs => 35M gal

Total processor usage => 70%

Total max flow processing capacity => 50M gal

A Case Study: Petri Net model (2/4)
The first level assurance applies backtracking algorithm.

Two aspects are applied to evaluate the strict static requirements “battery life” and “total maximum flow processing capacity”.

The second level assurance applies EA with the discard rate 1.1.
- If the averaged worst fitness value of the dynamic QoS is greater than 1.1 times this dynamic QoS req., the assembled case can be eliminated.

The alternatives that violate “battery life” requirement.

The alternative eliminated by PPCEA based on statistics.
Related Work

- DESERT: exploits Ordered Binary Decision Diagrams (OBDD) to perform design space exploration and analysis
  - pros: rapid design space exploration and analysis for functional requirements (i.e., explore implementation alternatives)
  - cons: static pruning approach: dataflow and constraints
  - issues on non-functional requirements:
    1. mostly focus on functional requirements
    2. QoS parameters need to be discretized
    3. probabilistic analysis: values must be power of two

Conclusion

- A QoS-Driven approach to deal with abundant QoS parameters
- Petri Nets as a formalism including predicates, time and event constraints
- GME as a configurable and customizable tool for Petri Nets
- AspectJ provides modular merit for constraint analyses
- Dynamic and parallel approach for eliminating infeasible alternatives by backtracking and branch-and-bound algorithms
- Statistical method to delete less probable alternatives by PPCEA
Future Work

- Termination criterion of reachability tree construction under specific QoS constraints
- Investigation on the priorities and affectation degree of non-orthogonal QoS
- A comprehensive automatic toolkit for the design space exploration and analyses

Questions?

- More research information http://www.cis.uab.edu/liush
- Acknowledgements
  This research is supported in part by U. S. Office of Naval Research award N00014-01-1-0746