Introduction

- Component based software methodology is widely used
  - software components with predefined interfaces to facilitate software reuse
  - requires middleware for location transparency and concurrency control
  - lots of configuration options to provide software flexibility
Problem Statement

- Each component has its own environmental assumptions
- If configuration settings do not meet the assumptions could cause problems
  - such as deadlocks, race conditions, priority inversions.
- Representation and analysis are needed to check the compatibility between components
- The analysis should
  - support the modeling of
    - Component interactions
    - Component timing constraints
    - Influences of preemptive scheduling algorithms
    - Influences of different component concurrency strategies
  - Automatically check the safety and liveness properties of a system

Previous Works about Modeling Theory

<table>
<thead>
<tr>
<th></th>
<th>Component Interaction</th>
<th>Timing Constraints</th>
<th>Preemptive System</th>
<th>Concurrency</th>
<th>Property Checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Checking</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Timed Automata</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Interface Automata</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>safety only</td>
</tr>
<tr>
<td>Hybrid Automata</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>generally unsolvable</td>
</tr>
</tbody>
</table>

- Tractable checking of all 4 areas is needed
**My Proposal**

- Develop a new component model
  - Combining Interface Automata, Timed Automata and Preemptive Scheduling Algorithms
  - It integrates the modeling of
    - component interactions
    - component timing constraints
    - the influences of preemptive scheduling algorithms
    - the influences of different component concurrency strategies
- Algorithms to convert the proposed model into a checkable timed automata model
- Build a tool to realize and check the proposed model

---

**Prior Works on Model Checker and Domain Specific Model Checking**

- Modeling Environments
  - GME, Ptolemy, ...
- Untimed Model Checkers
  - SPIN, Bogor, ...
- Model Checkers for Timed or Hybrid Automata
  - UPPAAL, IF toolset, HYTECH, ...
- Model Libraries or Domain Specific Modeling Techniques
  - Cadena, Madl et al, Subramonian et al, PTIDES, ...
- None of above integrate component modeling with timing constraints, preemptive scheduling and choice of concurrency strategies.
Structure of the Research

- Interface Automata
- Middleware concurrency control
- Timed and Hybrid Automata
- Proposed Component Model
- Research and Evaluation Plan
- Conclusions

Interface Automata

Alfaro and Henzinger (2001)

- Models component interactions
- Actions
  - Output (!)
  - Input (?)
  - Internal (;)
- Finite state machine
- Component composition
  - Needed by model checker to search a single state space
  - Assumes every component is active object
- Matching actions become internalized actions after composition
Deficiencies with Interface Automata

- No timing constraint support
- Every component is an active object
  - No concurrency strategy support
- Middleware concurrency strategies
  - Single Thread
    - Wait on connection
    - Wait on reactor
  - Multiple Threads
    - Thread pool

Wait on Connection Strategy
One request at a time, does not service others until current one finish
Interface Automata

Wait on Reactor Strategy
One request at a time, allows to service others while waiting

Interface Automata

Thread Pool Strategy
More than one request at a time
- Aims for applications written directly on top of middleware such as CORBA, nORB
- Use timed automata to model network and middleware entities.
- Problems:
  - Lack of abstraction
  - Requires explicit specification on network connection management
  - Requires explicit control to simulate thread execution
  - Prone to state space explosion
  - Domain specific optimization helps, but must be applied explicitly.
Prior Work on Modeling Middleware Concurrency
The Solution of Subramonian et al. (2006)

- Aims for applications written directly on top of middleware such as CORBA, nORB
- Use timed automata to model network and middleware entities.
- Problems:
  - Lack of abstraction
  - Requires explicit specification on network connection management
  - Requires explicit control to simulate thread execution
  - Prone to state space explosion
  - Domain specific optimization helps, but must be applied explicitly.

Component Composition

Modeling Component with Middleware Abstractions
My proposed solution

- Use component composition schemes to represent middleware concurrency strategies
  - Multi-threaded Composition
    - Parallel Composition: the only scheme used by Interface Automata
  - Single-threaded Compositions: my new additions
    - Atomic Composition
    - Monitor Composition
Parallel Composition

Atomic Composition
Monitors Composition

How to model time?
A Timed Automaton $A = (S, S_0, X, I, T)$

- $S$ is a finite set of locations
- $S_0 \subseteq S$: a set of starting locations
- $X$: a set of clocks
- $I : S \rightarrow C(X)$ location invariants
- $T \subseteq S \times C(X) \times 2^X \times S$ a set of transitions

Timed Automata models can’t express preemption semantics

- All clocks progress at the same speed
- During preemption, the clocks for representing execution time of the preempted task should be stopped
  - Not possible in Timed Automata model
Hybrid Automata
Henzinger (1996)

- Each location contains rate conditions for each variable
- Timed Automata is a special case of Hybrid Automata
- Model Checking with Hybrid Automata is generally undecidable
  - Some certain special cases are decidable but the complexities are NP-hard.

\[
\begin{align*}
& p_0 q_0 \\
& t_1 = 0 \\
& t_2 = 0
\end{align*}
\]

\[
\begin{align*}
& p_1 q_0 \\
& t_1 = 1 \\
& t_2 = 0
\end{align*}
\]

\[
\begin{align*}
& a? \\
& t_1 := 0
\end{align*}
\]

\[
\begin{align*}
& b? \\
& t_2 := 0
\end{align*}
\]


Proposed Component Model

My Solution for Preemption

- Use task response time instead of maximum execution time in Timed Automata Models
  - Task response time is subject to the changes of CPU allocation and scheduling algorithms
  - Timed Automata model does not have enough information for scheduling analysis
- Combine Interface Automata, Timed Automata and Scheduling Analysis
- What is needed for scheduling analysis?
  - Tasks and their worst case execution times
  - Task periodicity
  - The components share the same scheduling resources (Node)
A New Component Model

- **Actions**
  - Input Actions
  - Output Actions
  - Internal Actions

- **Locations**
- **Clocks**
- **Location Invariants**
- **Transitions**
  - Timed Transition: Guards, Resets
  - Preemptible Transitions
  - Output Rate Relations

---

Component Composition and Node

- **Component Composition Schemes**
  - Parallel Composition \( \otimes \)
  - Atomic Composition \( \odot \)
  - Monitor Composition \( \oplus \)

- **Composition Rules**
  - Symmetric: \( A \otimes B \equiv B \otimes A \)
  - Parallel composition cannot be nested inside atomic and monitor composition
    - Legal: \( A \otimes (B \odot C) \)
    - Illegal: \( A \odot (B \times C) \)

- **Node**: the mapping of components to resources
  - Used to calculate CPU utilization and preemption overhead
Rate Deduction and CPU utilization Calculation of a Node

- Given the input rate of all external actions, the rate of internalized input actions can be deduced.
- The rate of preemptible transitions is associated with their closest input actions.
- The CPU utilization of a node is bounded by $\sum e_i / p_i$
  - $e_i$: worst case transition time of the preemptive transition $i$
  - $p_i$: period of the preemptive transition $i$
Scheduling Algorithm and Preemptible Transitions

- **An Example with Rate Monotonic Scheduling (RMS) Algorithm:**
  - Higher rate tasks have higher priorities
  - Liu and Layland (1973)

- **CPU Utilization:** \( U = \sum e_i p_i \)

- **Task response time**
  \( t_i = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t_i}{p_k} \right\rceil e_k \)

Proposed Component Model

Composition Constraints

- Modeling communication delay between components
Research and Evaluation Plan

- Develop a tool to realize the proposed model
- Base on the Bogor Model checker
  - Bogor is open source
  - Bogor supports several key abstractions such as classes, objects, threads
- What I need to do
  - Develop a textual language for the specification of proposed model
  - Define an algorithm to convert the proposed model into a checkable Timed Automata model
  - Implement timed automata model checking in Bogor
- Model realistic component based applications and compare with
  - The approach of Subramonian et al.
  - Hybrid automata

Conclusions

- Component based software hides some complexities that may impact the safety and liveness properties of a system
- Model checking technology can help to verify those properties during composition
- My proposed component model deals with the following issues
  - Component compositions with different concurrency strategies
  - Timed automata don’t support systems with preemptive scheduling
  - Hybrid automata are generally unsolvable
- Tool need to be developed for checking the proposed component model