Case Study: Dependability Analysis of Embedded Control Systems using SystemC and Statistical Model Checking

Van-Chan Ngo

INRIA Rennes, France

March, 2015
Our SMC-based verification framework analyzes the dependability of large embedded control systems modeling in SystemC.
Agenda

- Modeling Dependability in SystemC
- Analyzing Dependability with SMC
- Some Results
- Demo
An Embedded Control System

- Characterized by control algorithms (i.e. feedback control loops), sensors, and actuators
- Are computer-based (i.e. processors run the control software)
- Effected by the failures of the sensors, the actuators, and the processors
- Examples: Process plants, medical device, disaster management, and flight control,...
## Metric Used for Dependability

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reliability</strong></td>
<td>the number of times the system fails to deliver the services</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>how likely the system is available to meet a demand for service</td>
</tr>
<tr>
<td><strong>Maintainability</strong></td>
<td>the repairing the system after a failure</td>
</tr>
<tr>
<td><strong>Mean time to failure</strong></td>
<td>time between observed system failures</td>
</tr>
<tr>
<td><strong>Rate of occurrence of failures</strong></td>
<td>number of failures in a time period</td>
</tr>
<tr>
<td><strong>Probability that system will fail</strong></td>
<td>to complete a service request</td>
</tr>
<tr>
<td><strong>How likely the system will be available</strong></td>
<td>for use</td>
</tr>
</tbody>
</table>
Dependability of ECS

• The reliability of the system is affected by the failures of the sensors, actuators, and processors (the probability of bus failure is negligible)

• The sensors and actuators implement M-of-N modular redundancy, meaning that the system can operate with M or more functional elements

• The I/O can have a transient fault, meaning that it can be refined automatically by the processor reboot

• The system shuts down when communication (between the processors through the bus to synchronize the actions) time-out occurs
## Mean Time to Failure and for Delays

<table>
<thead>
<tr>
<th>Components</th>
<th>Mean time to failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor</td>
<td>1 month</td>
</tr>
<tr>
<td>Actuator</td>
<td>2 months</td>
</tr>
<tr>
<td>Processor transient</td>
<td>1 day</td>
</tr>
<tr>
<td>Processor</td>
<td>1 year</td>
</tr>
<tr>
<td>Processor reboot</td>
<td>30 seconds (1 time unit)</td>
</tr>
<tr>
<td>Timer cycle</td>
<td>1 minute</td>
</tr>
</tbody>
</table>
CTMC: Model of Component Dependability

- **State-based** models such as Markov chains are capable of capturing various kinds of dependency, reliability, and availability.

- The mean time to failure and the delays are exponentially distributed.

- The failure of a component can be modeled as a Continuous Time Markov Chain with guards.

- The system is modeled as a set of CTMCs in which their transitions can be synchronized.

- Our case study: The size of the ECS model is $\sim 2^{155}$. 
CTMC Model of a Sensor Group

• A sensor group has 3 sensors in which a single one fails with rate $\lambda$

• The CTMC has 4 states: $s = 0 \ldots 3$ (the number of functional sensors)

• When $s > 0$, it spends the holding time

$$H \sim Exp(s \times \lambda)$$

• When the holding time ends, it makes a transition into

$$s = s - 1$$
Implementation with GNU Scientific Library (GSL)

- Each component is modeled as a module in SystemC.
- The module has some processes that implement a CTMC.
- These modules run concurrently and communicate to synchronized their actions (i.e., using buffer, signals).

GSL is a collection of routines for numerical computing consisting the functions of random number generation, random distribution, and statistics.

GSL can be used in multi-threaded programs in both C and C++.

1 Full source code: [https://project.inria.fr/plasma-lab/embedded-control-system/]
Execution Trace and Model

• An **observed variable** is a typed variable (i.e., usual scalar types, enumerated types)

• The evaluation of observed variables specifies the **state** of the SystemC model execution

• A state consists of:
  - the **kernel phases**, **event notifications**,
  - the **values of variables**, **location of program counter**, **call stack**, **processes status**,
  - and the **state of external libraries**

• The time points in which the observed variables are evaluated are determined by a set of events (**temporal resolution**)  

• An **execution trace** is a sequence of observed variable evaluations along with time
Overview of the Verification Process

• Define the set of observed variables and the temporal resolution

• Use MAG to generate the monitor and the aspect-advises

• Use AspectC++ with the aspect-advises to instrument the SystemC model

• Compile the instrumented model with the generated monitor to obtain the executable model
Analysis Result: Running
## Analysis Result: Observed Variables

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_sensors</td>
<td>Number of working sensor groups</td>
</tr>
<tr>
<td>number_actuators</td>
<td>Number of working actuator groups</td>
</tr>
<tr>
<td>proci_status</td>
<td>Current input processor's state</td>
</tr>
<tr>
<td>proco_status</td>
<td>Current output processor's state</td>
</tr>
<tr>
<td>procm_status</td>
<td>Current main processor's state</td>
</tr>
<tr>
<td>timeout_counts</td>
<td>Number of skipped cycles</td>
</tr>
<tr>
<td>reward_up</td>
<td>Time spent in “up” state</td>
</tr>
<tr>
<td>reward_danger</td>
<td>Time spent in “danger” state</td>
</tr>
<tr>
<td>reward_shutdown</td>
<td>Time spent in “shutdown” state</td>
</tr>
<tr>
<td>proci_reboot</td>
<td>Number of I's reboots</td>
</tr>
<tr>
<td>proco_reboot</td>
<td>Number of O's reboots</td>
</tr>
</tbody>
</table>
## Analysis Result: Temporal Resolution

<table>
<thead>
<tr>
<th>Event names</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMED_NOTIFY_PHASE_END</td>
<td>Observed variables are evaluated at the timed-cycle boundary or every time unit</td>
</tr>
</tbody>
</table>
Analysis Result: Expressing the Failures over Observed Variables

\[
\begin{align*}
\text{failure}_1 & := (\text{number}_\text{sensors} < 37) \land (\text{proci}_\text{status} = 2) \\
\text{failure}_2 & := (\text{number}_\text{actuators} < 27) \land (\text{proco}_\text{status} = 2) \\
\text{failure}_3 & := \text{timeout}_\text{counts} > 4 \\
\text{failure}_4 & := \text{procm}_\text{status} = 0 \\
\text{shutdown} & := \text{failure}_1 \lor \text{failure}_2 \lor \text{failure}_3 \lor \text{failure}_4 \\
\text{danger} & := (\text{proci}_\text{status} = 1 \lor \text{proco}_\text{status} = 1) \\
& \quad \land \neg \text{shutdown} \\
\text{up} & := \neg \text{shutdown} \land \neg \text{danger}
\end{align*}
\]
Analysis Result: Probability of Each Type of Failure

\[ F_{\leq T} (\text{failure}_i) \]
Analysis Result: Probability of Each Type of Failure Occurring First

$(\neg \text{shutdown}) \ U_{\leq T} \ (\text{failure}_i)$
Analysis Result: Expected Time Spent in Each Class of States

\[ X_{\leq T} \text{ (reward\_up)} \quad X_{\leq T} \text{ (reward\_danger)} \quad X_{\leq T} \text{ (reward\_shutdown)} \]
Analysis Result: Expected Number of Processor Reboots

$X_{\leq T} (\text{proci\_reboot}) \quad X_{\leq T} (\text{proco\_reboot})$
Analysis Result: Expected Number of Working Sensors and Actuators

\[ X_{\leq T} \text{ (number\_sensors)} \quad X_{\leq T} \text{ (number\_actuators)} \]
Demo!