Model-Based Development

Models for Reactive Software and its Environment
  • Discrete Models
  • Continuous Models

Models for Analysis
  • Building Requirements: Conceptual Model
  • Building Solutions: Behavioral Model

Models for Design
  • Building Sound Systems: Conceptual Model
  • Building Correct Systems: Behavioral Model

Models for Construction
  • Describing Implementation: Complete Computation
  • Ensuring Functionality: Test Case
Building Requirements

Models for Reactive Software and its Environment

Models for Analysis

• Building Requirements: Conceptual Model
  – Formalizing Requirements
  – Structuring Requirements
  – Classifying Requirements

• Building Solutions: Behavioral Model

Models for Design

Models for Construction
Structuring Requirements: Unstructured and Structured Models

As the map Figure 1.1 shows, the lines feed from points north, south, and east into San Francisco, with the critical link being the section from Oakland to San Francisco. This is a key point where fast linear segments are joined together, and a gate is placed under the hourglass.

The BT needs to utilize this section of the line between points north and south. The BT needs to ensure that it can maintain a safe speed and a safe distance from other trains. The BT needs to be able to respond to any unexpected events, such as a sudden stop or a derailment, and ensure the safety of passengers and the general public.

Secondly, the Violent Station Computer (VSC), which is a slower but reliable enough to ensure safety-related MBBS goals.

The NVSC performs speed and acceleration commands, considering both safety and performance goals. The NVSC checks that they do not exceed maximum bounds for safety. For smooth operations, the NVSC generally will have to propose speeds and accelerations lower than the absolute maximum against which the VSC checks. For this case study, you may partition functions among the two machines as you wish, subject to meeting overall safety goals, the fact that the NVSC-handlers alone cannot support these goals, and the VSC cannot do more than essentially one computation of a bounding speed and acceleration.

(Note: This description of the NVSC and VSC does not precisely represent the actual interactions between the two machines being designed for the AATC system. The discussion above captures the basic concepts, but omit some real [and proprietary] system design complications.)

The following sections provide some additional details about the control system.

5. Inputs and Outputs to the Control Algorithm

The control algorithm receives the following information, updated every 1/2 second:

- The outputs of the position algorithm - speed and standard deviations on both position and velocity - for all trains in the area. Normal performance for this system is for the standard deviation to be 2 or 3 feet. Note that both the mean and standard deviations will vary a little (e.g., a few feet) from each 1/second-time step to the next.
- The Message Origin Time Tag (MOTT). This is the time a given train sent its most recent report. Information from this report is folded into the position tracking system. Thus, the MOTT is a measure of information currency. After receiving a MOTT from a train, the control system then attaches the same MOTT to acceleration and velocity commands that are then sent back out to the train, and the commands are only valid until MOTT + 2.5 seconds.
- Gate information, input, closest to the interlocking system.
- Any special speed restrictions in either the whole system or individual track segments.

It is the responsibility of the control system to separate trains sufficiently so that if a train is in front instantaneously stopped (e.g., by a derailment) then the following train could stop without hitting it (assuming worst-case conditions). Additionally, if a "deadlock" state occurs in front of the train, for which a speed/acceleration
Structuring Requirements: Informal Specification

As the map Figure 1.1 shows, the lines feed from points north, south and east into San Francisco, with the critical link being the section from Oakland to

BART needs to utilize this segment - in a tube underground - would be prohibitively expensive. AATC will allow.

The VSC checks that they do not exceed maximum bounds for safety. For smooth operations, the VSC generally will have to propose speeds and accelerations lower than the absolute minimum against which the VSC checks. For this case study, you may partition functions among the two machines as you wish, subject to meeting overall safety goals, the fact that the NVSC hardware alone cannot support these goals, and the VSC cannot do more than essentially one computation of a bounding speed and acceleration.

(Note: This description of the NVS and VSC does not precisely represent the actual interaction between the two machines being designed for the AATC system. The discussion above captures the basic concepts, but omits some real [and proprietary] system design complexities.)

The following sections provide some additional details about the control system.

5. Inputs and Outputs to the Control Algorithm

The control algorithm receives the following information, updated every 1/2 second:

- The outputs of the position algorithm – mean and standard deviation on both position and velocity - for all trains in the area. Nominal performance for the system is for the standard deviation to be 2 to 3 feet. Note that both the mean and standard deviation will vary a little (e.g., a few feet) from each 1/2-second-time step to the next.

- The Message Origin Time Tag (MOTT). This is the time a given train sent its most recent report. Information from this report is folded into the position tracking system. Thus the MOTT is a measure of information currency. After receiving a MOTT from a train, the control system then attaches the same MOTT to acceleration and velocity commands that are then sent back out to the train, and the commands are only valid until MOTT + 2 seconds.

- Gate information (open, closed) from the interlocking system.

- Any special speed restrictions on either the whole system or individual track segments.

6. Analysis

It is the responsibility of the control system to separate trains sufficiently so that if a train in front instantaneously stopped (e.g., it derailed) then the following train could stop without hitting it (assuming worst-case conditions). Additionally, if a "leading train" (one in front of the train for which a speed/acceleration
Extended conceptual model: Explicit representation of requirements
Structuring Requirements: **Conceptual Model**

- Define requirements from informal specification
- Refine requirements
- Classify requirements into constraints
- Formalize constraints into model views
Building Requirements

Models for Reactive Software and its Environment

Models for Analysis
• Building Requirements: Conceptual Model
  – Formalizing Requirements
  – **Structuring Requirements**
  – Classifiying Requirements

• Building Solutions: Behavioral Model

Models for Design

Models for Construction
Define Requirements: Grouping information into requirements

- Structuring Requirements: Identify Requirements

- Following train would hit it. A train should stay below the maximum speed that segment of track can handle.

- Commands are time stamped and become invalid 2 seconds after the delayed time. This time stamping (or time hopping) is done due to what is called a Message Originiation Time Tag (MOTT). When a train sends performance data back to the station, it attaches the time that it sends (originates) the message. When that information is used to update the position estimates, the MOTT is associated with that position estimates. The time stamp provides a measure of the currency of a position estimate. When that position estimate is used to compute a speed/acceleration command, the MOTT is attached to the command. The train then checks the MOTT before exercising a command. A train will continue to exercise that command until a new one arrives at which that command expires, 7 seconds after the originating time.

- If the train does not have a currently valid command, it goes into maximum braking. The central algorithm then have to be designed so that if all communications are lost, then when commands arrive and trains come to a stop, no safe violation will have occurred. That is, the stopping location of any train after last communication, has timed out, and has come to a stop will be (1) behind any closed gates and (2) behind the rear end of any trains ahead. This sequence of events, along with a very pessimistic definition of how long it physically takes to stop a train once braking begins, defines what is called the Worst Case Stopping Profile. Similarly, the central algorithms have to be designed so that whether or not it happens, the train must not exceed any track speed limits.
Refine Requirements: Identifying (common) sub-requirements
Building Requirements

Models for Reactive Software and its Environment

Models for Analysis
• Building Requirements: Conceptual Model
  – Formalizing Requirements
  – Structuring Requirements
  – **Classifying Requirements**

• Building Solutions: Behavioral Model

Models for Design

Models for Construction
Classify Requirements: **Classify Constraint**

Classify requirements: Introduce architectural, modal, type constraints
Classify Requirements: **Motivate Models**

Formalize Constraints: Define conceptual elements
Models for Analysis

Models for Reactive Software and its Environment

Models for Analysis
• Building Requirements: Conceptual Model

• Building Solutions: Behavioral Model
  – Validate Models
  – Formalize Requirements
  – Find Solutions

Models for Design

Models for Construction
Building Solutions

Models for Reactive Software and its Environment

Models for Analysis
• Building Requirements: Conceptual Model
• Building Solutions: Behavioral Model
  – Validate Models
  – Formalize Requirements
  – Find Solutions

Models for Design

Models for Construction
Validate Models: Critical and accessible information

Environment and Interface: Critical vs. accessible information:

- **Environment variables:**
  - Controlled and monitored process variables
  - Examples: Position, speed of train

- **Interface variables:**
  - Input and output ports of the environment (system)
  - Examples: Reports, commands
Modeling the environment: Avoiding implicit assumptions

- Explicit approximations
  - Simplifications of environment
  - Example: Laws of physics

- Explicit limitations
  - Restrictions on environment
  - Example: No backward movement
Validate Models: Validation

Developing a precise description of environment and interface:

- describe the critical and accessable information of the environment
- explicitly formalize assumptions about the environment
Building Solutions

Models for Reactive Software and its Environment

Models for Analysis
• Building Requirements: Conceptual Model

• Building Solutions: Behavioral Model
  – Validate Models
  – **Formalize Requirements**
  – Find Solutions

Models for Design

Models for Construction
Formalize Requirements: **Defining Safety Properties**

**Safety Properties:** Properties of the environment
- Defined in terms of variables of the environment
- Ensured by variables of the interface
- Requires models of system and environment for verification
Formalize Requirements: Defining Safety Properties

Requirement: Collision avoidance

- Definition: “A train must not hit a leading train at any time”
- Formalization: \( \forall \tau \in T. \forall t_1, t_2 \in \text{Trains. } t_1.\text{pos}_\tau + \text{Nose} \neq t_2.\text{pos}_\tau + \text{Tail} \)
Building Solutions

Models for Reactive Software and its Environment

Models for Analysis
• Building Requirements: Conceptual Model

• Building Solutions: Behavioral Model
  – Validate Models
  – Formalize Requirements
  – Find Solutions

Models for Design

Models for Construction
System Requirements: Properties of the System and the Environment

- Defined in terms of variables of interface and environment
- Invariant Property: If enforced when system is in charge, it will also hold in later steps
- Consequence: Restriction of the current command
Finding Solutions: **Invariant Safety Properties**

**Invariant safety property:**
- System: Respects property when issuing commands
- Environment: Respects property in between/after commands
- Example: Worst-case stopping distance
  - System: Never commands a speed above the worst-case stopping distance
  - Environment: In case of signal loss train stops within worst-case distance
Finding Solutions: **Necessary Precondition**

### Worst Case Stopping Distances for Different States of a Train

<table>
<thead>
<tr>
<th>cs</th>
<th>ct</th>
<th>cf</th>
<th>ms</th>
<th>. . .</th>
<th>wcsd(cs, cf, ct, cv, ca, ms, mt, v, a, mc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>&gt; 0</td>
<td>=  ct</td>
<td>P:C</td>
<td>. .</td>
<td>v × ct + wcsd(N, cf – ct, 0, cv, ca, P:C, 0, v, a, mc)</td>
</tr>
<tr>
<td></td>
<td>. .</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>N</td>
<td>- 0</td>
<td>-</td>
<td>.</td>
<td>.</td>
<td>wcsd(F, 0, 0, 0, 0, ms, 0, v, a, MBR)</td>
</tr>
<tr>
<td></td>
<td>. .</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>F</td>
<td>-</td>
<td>P:C</td>
<td>. .</td>
<td>.</td>
<td>wcsd(F, 0, 0, 0, 0, EB, MC, v, 0, MBR)</td>
</tr>
<tr>
<td></td>
<td>. .</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>F</td>
<td>-</td>
<td>EB</td>
<td>. .</td>
<td>.</td>
<td>v × mt + wcsd(F, 0, 0, 0, 0, B:D, 0, v, 0, MBR)</td>
</tr>
<tr>
<td></td>
<td>. .</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>F</td>
<td>-</td>
<td>B:K</td>
<td>. .</td>
<td>.</td>
<td>-1 / (2 × MBR) × v + wcsd(F, 0, 0, 0, 0, B:S, 0, 0, 0, 0)</td>
</tr>
<tr>
<td></td>
<td>. .</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>F</td>
<td>-</td>
<td>B:S</td>
<td>. .</td>
<td>.</td>
<td>0</td>
</tr>
</tbody>
</table>

Local Variables of TrainModel:TrainModelBehavior
real acc = 0.0 * Current Acceleration of Train
real Timer = 0.0 * Timer for reconfiguration

**Images:**
- Diagram showing the state transitions and variables of the TrainController and TrainModel.
- Graph illustrating the necessary conditions for stopping distances with state transitions and variables.
Finding Solutions: Inductive Construction

| Worst Case Stopping Distances for Different States of a Train |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| cs  | ct  | cf  | ms  | ... | wcsd(cs, cf, ct, cv, ca, ms, mt, v, a, mc) |
| N   | > 0 | >= ct | P:C | ... | v \times ct + wcsd(N, cf – ct, 0, cv, ca, P:C, 0, v, a, mc) |
| N   | 0   | -    | P:C | ... | wcsd(F, 0, 0, 0, 0, ms, 0, v, a, MBR) |
| F   | -   | EB   | ... | wcsd(F, 0, 0, 0, 0, EB, MC, v, 0, MBR) |
| F   | -   | B:K  | ... | wcsd(F, 0, 0, 0, 0, B:D, 0, v, 0, MBR) |
| F   | -   | B:S  | ... | wcsd(F, 0, 0, 0, 0, B:S, 0, 0, 0, 0) |

- wcsd(F, 0, 0, 0, 0, MBR)