Understand A Control System’s Requirements and Its Specifications

7/25/2019

Presented by: Ruoxi Zhang
CS 846 – Summer, 2019
Outline

- Terminologies
  - Control system
  - Requirement and specification
  - Domain knowledge
- Example
- Conclusion
A stainless steel turnstile is designed to provide unsupervised access control for office or building which realize intelligent pedestrian control and management.

EXAMPLE OF A CONTROL SYSTEM (2)

A conventional fixed-wing aircraft flight control system consists of necessary operating mechanisms to control an aircraft's direction in flight.

A control system is a machine that interacts with its environment to bring about or maintain relationships in that environment [4].

- **Machine**: the computer-based machine to be developed
- **Environment**: the relevant physical world
- **Interactions**: e.g., the machine receives input from the physical world via sensors and influences it via actuators

Such systems are often deployed in safety-critical environments.
Requirement vs. Specification (1)

- A **requirement** states desired relationships in the environment — relationships that will be brought about or maintained by the machine [4].
  - “What are we trying to accomplish?”

- A **specification** describes the behavior of the machine at its interface which sufficient to achieve the requirement [4].
  - “What would that software system do?”
The environment is described in two ways [5]:

- **Indicative** statements: describe the environment as it is in the absence of the machine
- **Optative** statements: describe the environment as we hope it will become because of the machine

- $R$ (a set of requirements) and $S$ (a set of specifications) are optative.

- Both requirement and specification are expressed entirely in terms of environment phenomena.
  - Avoid implementation bias since no statements are made about the proposed machine [4].
**Requirement vs. Specification (3)**

- Each **action** must be identified as belonging to exactly one of the three categories.
  - All three types of actions are relevant to RE, and they might need to be described or constrained.

<table>
<thead>
<tr>
<th>Action Type</th>
<th>Environment Controlled</th>
<th>Machine Controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shared</strong></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Unshared</strong></td>
<td>X</td>
<td>![X]</td>
</tr>
</tbody>
</table>

*indicates which agent controls, performs, or takes responsibility for the action*

- [interface] observable by both the machine and environment
- private to the environment and unobservable by the machine
- internal actions of the machine
A specification is derived from a requirement, and it is a restricted kind of requirement.

- Remove all references to machine inaccessible phenomena.

Three rules of specifications [5]

- The members of specifications do not constrain the environment;
- They are not stated in terms of any unshared actions or state components;
- They do not refer to the future.

<table>
<thead>
<tr>
<th>Action Type</th>
<th>Environment Controlled</th>
<th>Machine Controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Unshared</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Domain Knowledge (1)

- $K$: relevant domain knowledge (assumptions about the environment).
- $K$ is indicative.

- Each property or assertion must be identified as a requirement (R), statement of domain knowledge (D), or specification (S).

- $S$ and $K$ together must be sufficient to guarantee that the requirements are satisfied: $S, K \vdash R$ [1, 5].
Domain Knowledge (2)

- Requirements that constrain the environment are satisfied by coordinating specifications with domain knowledge.

- Requirements with unshared information are refined using domain knowledge relating unshared information to shared information.

<table>
<thead>
<tr>
<th>Action Type</th>
<th>Environment Controlled</th>
<th>Machine Controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Unshared</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Domain Knowledge (3)

- The machine is under no obligation if it is used in an environment in which the assumption is false.

- E.g. specify the control program of a room heating system [2, 3]
  - Specification: corrective action should follow when the temperature sensor indicates that some limit value has been exceeded
  - Domain knowledge: the assumptions about the accuracy of the sensors
Example (1)

- E.g., specify the control system of a turnstile at the entrance to a zoo [4]
  - Mechanical hardware: a rotating barrier, a coin slot, and a locking device
  - Development job: controlling software

- Relevant environment phenomena

<table>
<thead>
<tr>
<th>Unary predicates</th>
<th>Designations</th>
<th>Share</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push(e)</td>
<td>In event e a visitor pushes the barrier to its intermediate position</td>
<td>Y</td>
<td>Env.</td>
</tr>
<tr>
<td>Enter(e)</td>
<td>In e a visitor pushes the barrier fully and so gains entry</td>
<td>N</td>
<td>Env.</td>
</tr>
<tr>
<td>Coin(e)</td>
<td>In e a valid coin is inserted into the coin slot</td>
<td>Y</td>
<td>Env.</td>
</tr>
<tr>
<td>Lock(e)</td>
<td>In e the turnstile receives a locking signal</td>
<td>Y</td>
<td>Machine</td>
</tr>
<tr>
<td>Unlock(e)</td>
<td>In e the turnstile receives an unlocking signal</td>
<td>Y</td>
<td>Machine</td>
</tr>
</tbody>
</table>
Example (2)

- Assume events are atomic and instantaneous.

<table>
<thead>
<tr>
<th>Event(e)</th>
<th>e is an atomic instantaneous event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ends(e, v)</td>
<td>Event e ends interval v</td>
</tr>
</tbody>
</table>
Example (3)

- Domain Knowledge
  - [IND1] Push and Enter events alternate, starting with Push.
  - [IND2] \( \forall e, v \cdot ((LU0(v) \land Ends(e, v)) \rightarrow \neg Push(e)) \)
    - If Lock and Unlock events alternate, starting with Unlock, then a Push event can occur only after an Unlock and before the next Lock.
    - Push events are impossible in state \( LU0 \).
Example (4)

- Requirement: no-one should enter without paying.
  - Note that “payments alternate with entries” is not a requirement.
  - \([\text{OPT1}] \forall \, v, m, n \cdot ((\text{Enter}(v, m) \land \text{Coin}(v, n)) \to (m \leq n))\) \text{ Enter} \leq \text{Coin}
    - Define Push\((v, n)\), Enter\((v, n)\) and Coin\((v, n)\) meaning that the count of Push, Enter and Coin events respectively preceding interval \(v\) is \(n\).

- Problems
  - The enter events are not shared phenomena.
  - The specification constrains the state in every interval, including those that are in the future.
Example (5)

- Specification
  - [OPT1] \( \forall v, m, n \cdot ((\text{Enter#}(v, m) \land \text{Coin#}(v, n)) \rightarrow (m \leq n)) \) \( \text{Enter#} \leq \text{Coin#} \)
  - [IND3] \( \forall v, m, n \cdot ((\text{Enter#}(v, m) \land \text{Push#}(v, n)) \rightarrow (n - 1 \leq m \leq n)) \) \( \text{Push#-1} \leq \text{Enter#} \leq \text{Push#} \)
  - [OPT1a] \( \forall v, m, n \cdot ((\text{Push#}(v, m) \land \text{Coin#}(v, n)) \rightarrow (m \leq n)) \) \( \text{Push#} \leq \text{Coin#} \)

- When \( \text{Push#} \) already equals \( \text{Coin#} \), the machine must prevent a further \( \text{Push} \) at least until after a further \( \text{Coin} \) event.

- [OPT2-safety] \( \forall v, e, n \cdot (((\text{LU0}(v)) \land \text{Push#}(v, n) \land \text{Coin#}(v, n) \land \text{Ends}(e, v)) \rightarrow \neg \text{Unlock}(e)) \)

- [DEF2-liveness] \( \text{ReqLock}(v) \triangleq (\text{LU1}(v) \land \exists n \cdot (\text{Push#}(v, n) \land \text{Coin#}(v, n))) \)

  - In state \( \text{ReqLock} \) the machine must perform a Lock event.
Conclusion

- The specification is implantable if [5]:
  - There is a set $R$ of requirements (validated as acceptable to the customer).
  - There is a set $K$ of statements of domain knowledge/assumptions (validated as true of the environment).
  - There is a set $S$ of specifications.
    - Do not constrain the environment;
    - Do not consist of unshared actions;
    - Do not refer to the future.
  - A proof shows that $S, K \vdash R$.
  - A proof shows that $S$ and $K$ are consistent. Together imply that $S$, $K$, and $R$ are consistent with each other.
References


THANK YOU