

Sequence-Based Specification

Tom Swain

tomswain@comcast.net

Specification Objectives

- **Completeness**

- a response is defined for every stimulus history

- **Consistency**

- each stimulus history maps to only one response

- **Correctness**

- the specification is explicitly traceable to the requirements

How Can We Ensure Correct Operation of Critical Software?

- Testing?
 - A sampling process
 - As complexity increases, a sufficient sample becomes impractical
- Peer Review of Documentation/Code?
 - Often subjective
 - Labor intensive
 - No objective criteria for sufficiency

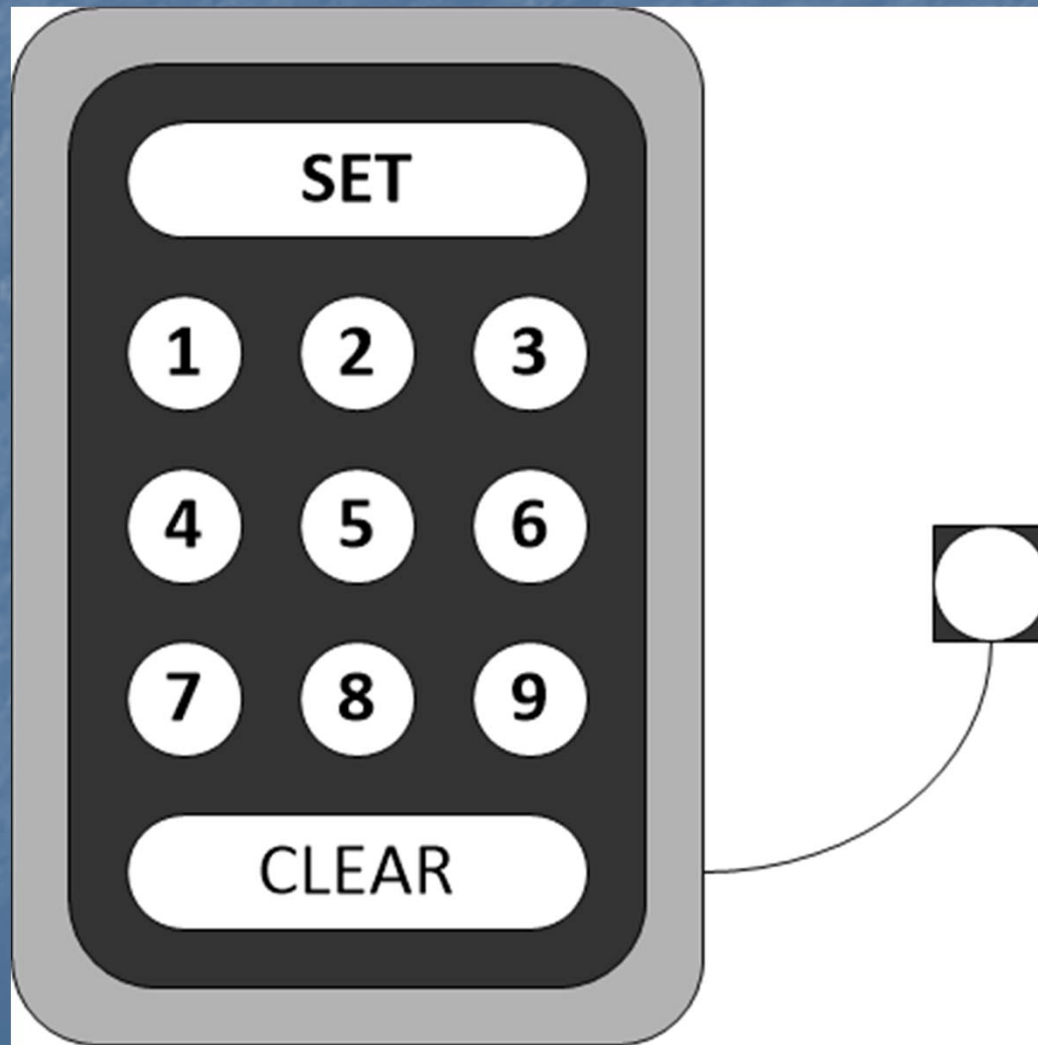
The Only Practical Alternative: Rigorous Specification and Design

- Basic Premise
 - A software program is a rule for computing a mathematical function that maps all possible stimulus histories to all possible responses
- Approach
 - Derive mathematically rigorous specification and design from requirements
 - Prove critical design properties using basic mathematics of software

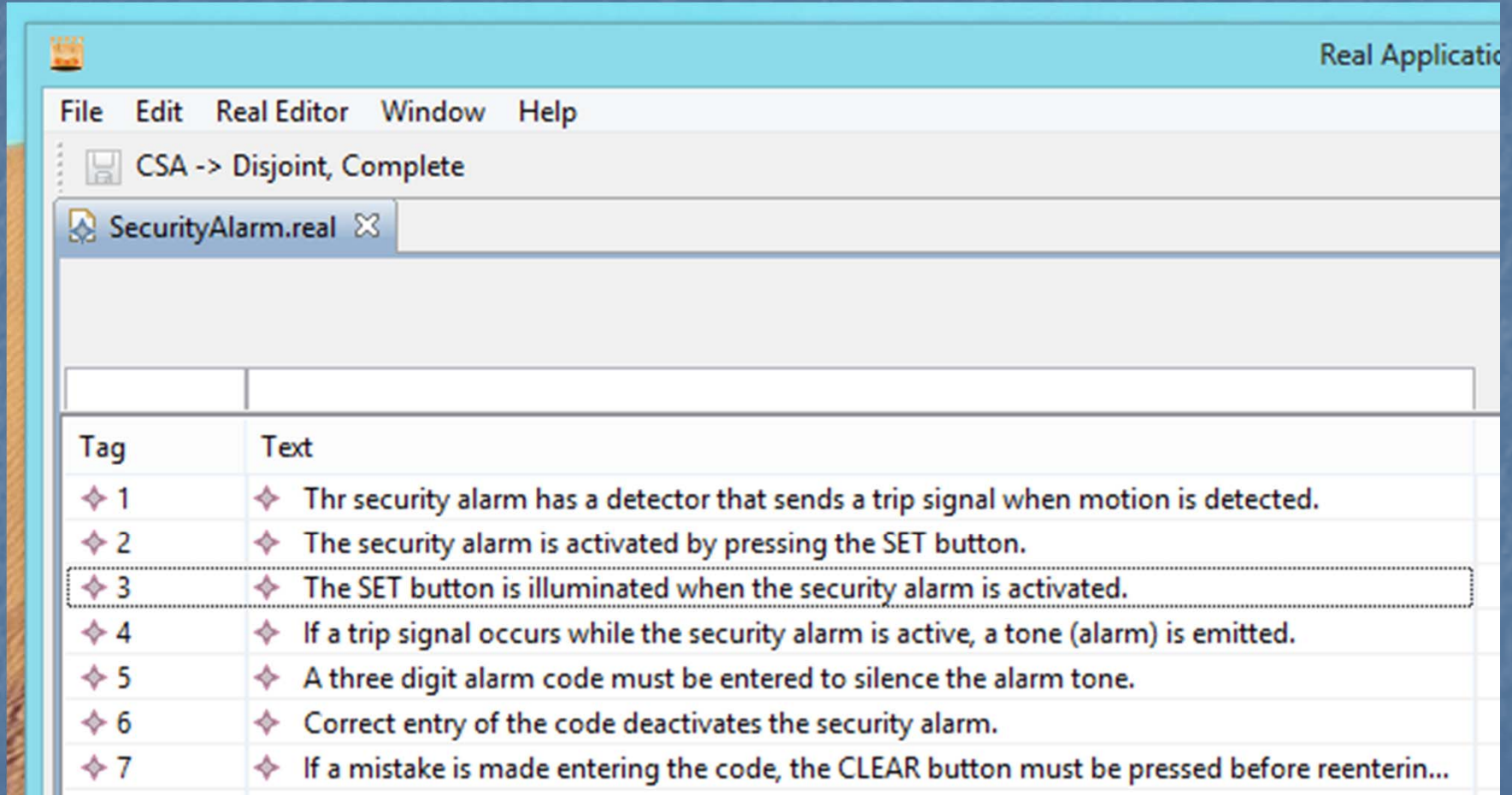
Problem: How Do We Produce the Formal Specification?



Security Alarm Example



Security Alarm Requirements



Real Application

File Edit Real Editor Window Help

CSA -> Disjoint, Complete

SecurityAlarm.real

Tag	Text
1	The security alarm has a detector that sends a trip signal when motion is detected.
2	The security alarm is activated by pressing the SET button.
3	The SET button is illuminated when the security alarm is activated.
4	If a trip signal occurs while the security alarm is active, a tone (alarm) is emitted.
5	A three digit alarm code must be entered to silence the alarm tone.
6	Correct entry of the code deactivates the security alarm.
7	If a mistake is made entering the code, the CLEAR button must be pressed before reenterin...

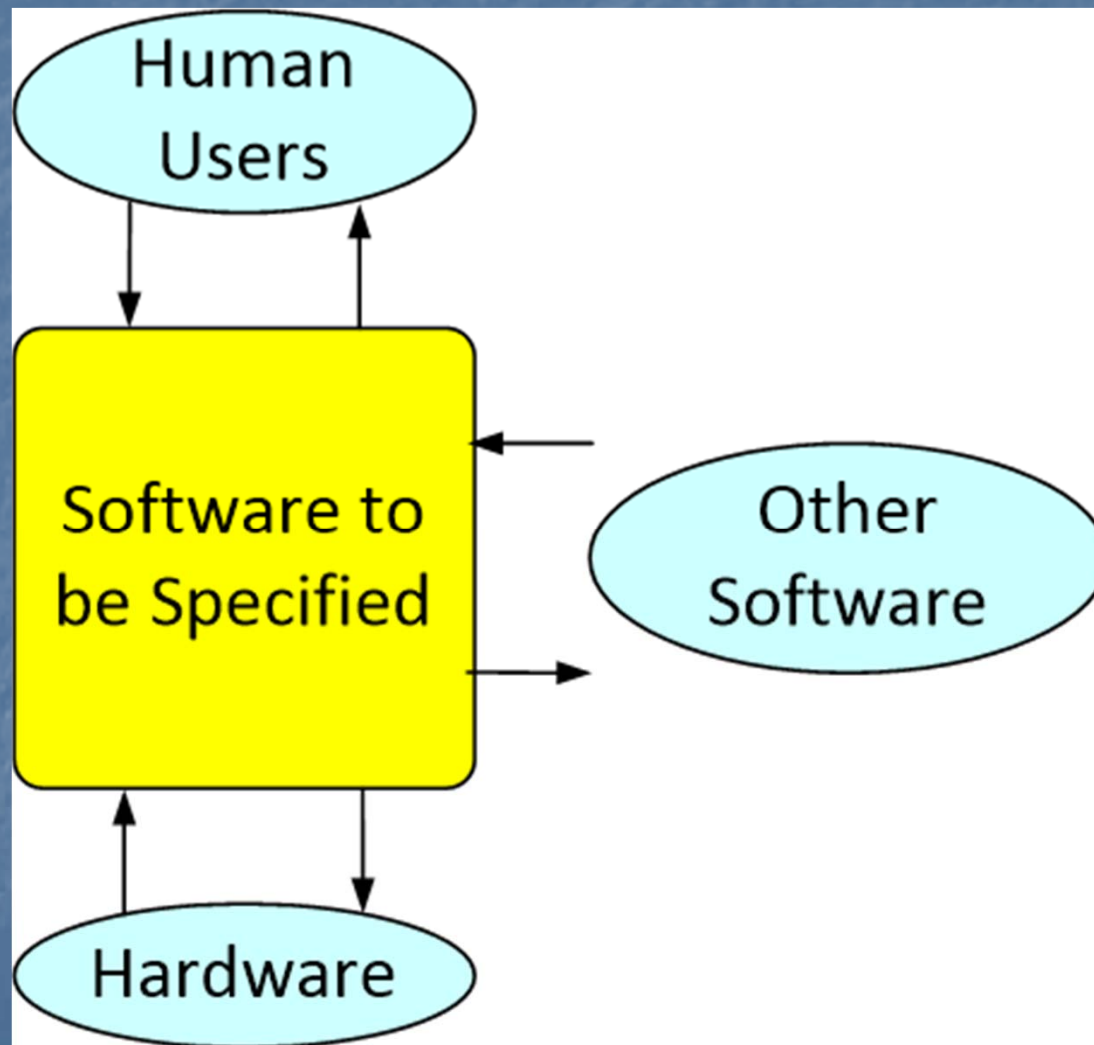
Requirements

- Individual requirements tagged for traceability.
- Initial requirements assumed to be incomplete, inconsistent, and possibly incorrect.
- Enumeration is a systematic process for discovering omissions and ambiguities.
- Domain experts review clarified requirements for correctness.

Rigorous Specification Process

- Establish system boundary.
- Define human/software/hardware interfaces.
- Itemize stimuli.
- Itemize responses.
- Perform enumeration.
- Perform canonical sequence analysis.
- Generate state machine specification.
- Transform to formal notation (e.g., ACL2).

System Boundary and Interfaces



Level of Detail to Specify

- **Necessary Detail Test**

- A behavioral detail is necessary to the spec if it affects interface development or long-run system behavior.

- **Detail to be Deferred**

- Specific algorithm used for internal calculation.

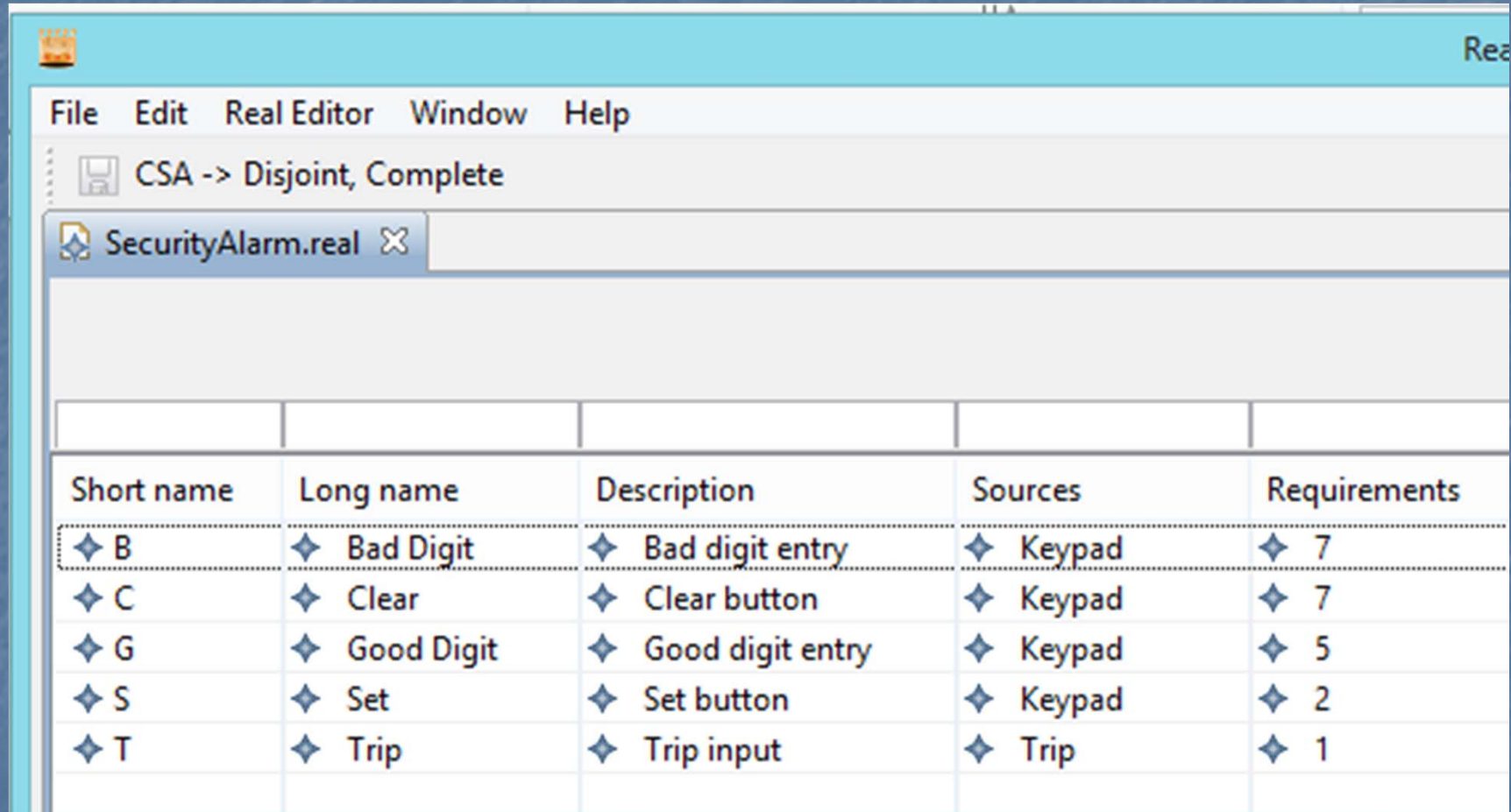
- **Necessary Detail**

- Changes to default settings.

Definitions

- Stimulus - an event resulting in information flow from the outside to the inside of the system boundary
- Response - information flow from inside to outside the system boundary

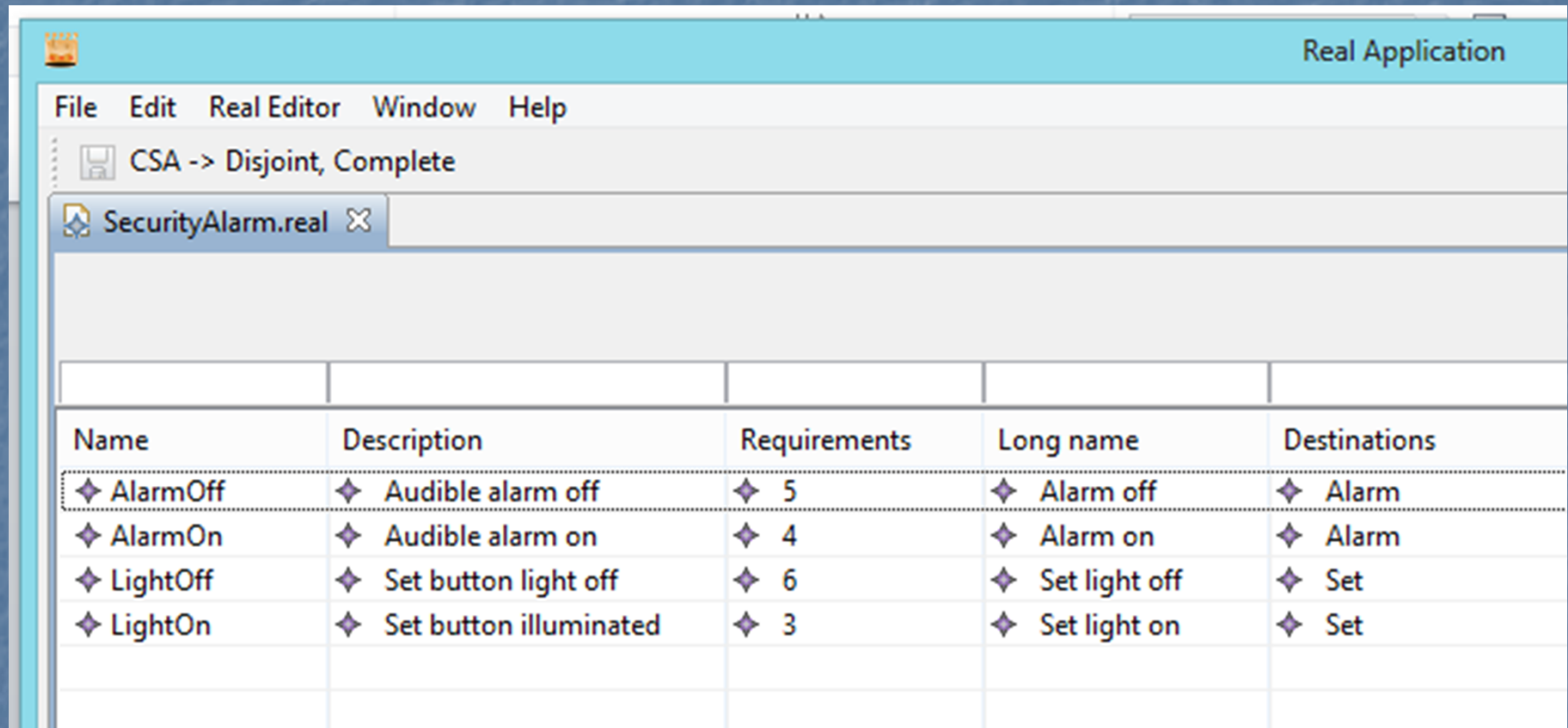
Security Alarm Stimuli



The screenshot shows the Real Editor software interface. The menu bar includes File, Edit, Real Editor, Window, and Help. The title bar indicates the current project is 'CSA -> Disjoint, Complete'. A tab labeled 'SecurityAlarm.real' is open. Below the tab is a table with five columns: Short name, Long name, Description, Sources, and Requirements. The table contains five rows of stimuli, each marked with a diamond icon. The first row is highlighted with a dashed border.

Short name	Long name	Description	Sources	Requirements
◆ B	◆ Bad Digit	◆ Bad digit entry	◆ Keypad	◆ 7
◆ C	◆ Clear	◆ Clear button	◆ Keypad	◆ 7
◆ G	◆ Good Digit	◆ Good digit entry	◆ Keypad	◆ 5
◆ S	◆ Set	◆ Set button	◆ Keypad	◆ 2
◆ T	◆ Trip	◆ Trip input	◆ Trip	◆ 1

Security Alarm Responses



The screenshot shows the 'Real Application' window of a software tool. The menu bar includes 'File', 'Edit', 'Real Editor', 'Window', and 'Help'. The title bar indicates the current project is 'CSA -> Disjoint, Complete'. A tab labeled 'SecurityAlarm.real' is active. Below the tab is a table with five columns: 'Name', 'Description', 'Requirements', 'Long name', and 'Destinations'. The table contains four rows of data, each representing a different alarm response.

Name	Description	Requirements	Long name	Destinations
◆ AlarmOff	◆ Audible alarm off	◆ 5	◆ Alarm off	◆ Alarm
◆ AlarmOn	◆ Audible alarm on	◆ 4	◆ Alarm on	◆ Alarm
◆ LightOff	◆ Set button light off	◆ 6	◆ Set light off	◆ Set
◆ LightOn	◆ Set button illuminated	◆ 3	◆ Set light on	◆ Set

Enumeration Overview

- Define response to all possible stimulus sequences by considering them in order of length.
- Continue until all sequences of a given length are illegal or equivalent to previous sequences.
- Identify canonical sequences.

Enumeration Mechanics

- For each sequence of length n , do the following:
 - If illegal, mark it illegal and do not extend further.
 - Document correct response based on requirements.
 - If no requirement found, create derived requirement.
 - Record requirements trace.
 - Check for equivalence with previous sequences.
- Extend only those sequences that are not illegal or equivalent.

Important Definitions

- A sequence is
 - illegal by definition if it is impossible for the system to observe it or for the environment to generate it
 - illegal by design if it violates the system definition

More Important Definitions

- A sequence is
 - equivalent to another sequence if their responses to any future stimulus sequences are identical
 - reduced if it has been declared equivalent to a previous sequence
 - canonical if it is legal and unreduced when enumeration is complete

Abstract Stimulus

- Single stimulus representing multiple events.
- Examples:
 - Menu selection, toolbar button, and keyboard accelerator that initiate the same function.
 - Numeric input mapped into ranges (e.g., normal, critical, invalid).

Sequence Abstraction

- Arrival of a stimulus with certain historical conditions in effect.
- Uses-Examples:
 - Temporal Semantics - Replace individual clock pulses with every n^{th} pulse.
 - Interface Details - Combine multiple keystrokes/mouse clicks into a single selection stimulus.
 - Function Isolation - Separate specification of largely independent interfaces or subsystems.

Enumeration Example

Sequences of Lengths 1-2

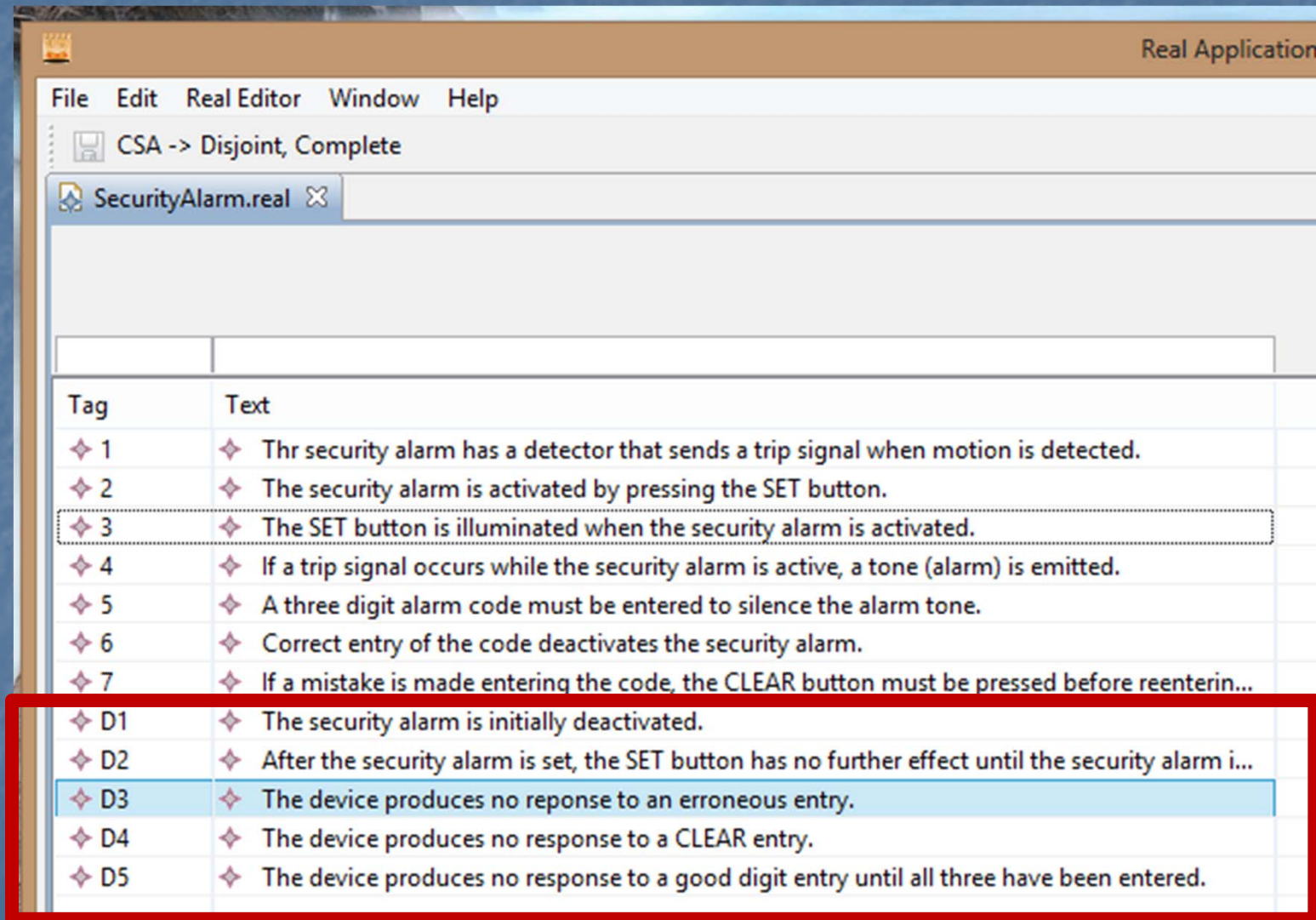
Name	Length	Prefix	Action	Responses	Response trace	Legal	Equivalence
◆ B	◆ 1	◆ <lambda>	◆ B	◆	◆ D1	◆	◆
◆ C	◆ 1	◆ <lambda>	◆ C	◆	◆ D1	◆	◆
◆ G	◆ 1	◆ <lambda>	◆ G	◆	◆ D1	◆	◆
◆ S	◆ 1	◆ <lambda>	◆ S	◆ LightOn	◆ 2, 3	◆ yes	◆
◆ T	◆ 1	◆ <lambda>	◆ T	◆	◆ D1	◆	◆
◆ S.B	◆ 2	◆ S	◆ B	◆	◆ D3	◆ yes	◆
◆ S.C	◆ 2	◆ S	◆ C	◆	◆ D4	◆ yes	◆ S
◆ S.G	◆ 2	◆ S	◆ G	◆ LightOn	◆ D5	◆ yes	◆
◆ S.S	◆ 2	◆ S	◆ S	◆ AlarmOff	◆ D2	◆ yes	◆ S
◆ S.T	◆ 2	◆ S	◆ T	◆ AlarmOn	◆ 4	◆ yes	◆
◆ S.B.B	◆ 3	◆ S.B	◆ B	◆	◆ D3	◆ yes	◆ S.B
◆ S.B.C	◆ 3	◆ S.B	◆ C	◆	◆ 7, D4	◆ yes	◆ S
◆ S.B.G	◆ 3	◆ S.B	◆ G	◆	◆ 7	◆	◆
◆ S.B.S	◆ 3	◆ S.B	◆ S	◆	◆ D2	◆ yes	◆ S.B
◆ S.B.T	◆ 3	◆ S.B	◆ T	◆ AlarmOn	◆ 4	◆ yes	◆
◆ S.G.B	◆ 3	◆ S.G	◆ B	◆	◆ D3	◆ yes	◆ S.B
◆ S.G.C	◆ 3	◆ S.G	◆ C	◆	◆ D4	◆ yes	◆ S

Requirements	Interfaces	Stimuli	Outputs	Named Responses	Predicates	Mappings	State Variable Definition	Canonical Sequences
--------------	------------	---------	---------	-----------------	------------	----------	---------------------------	---------------------

Properties

Property	Value
Name	S.T.T
Length	3

Derived Requirements



Real Application

File Edit Real Editor Window Help

CSA -> Disjoint, Complete

SecurityAlarm.real

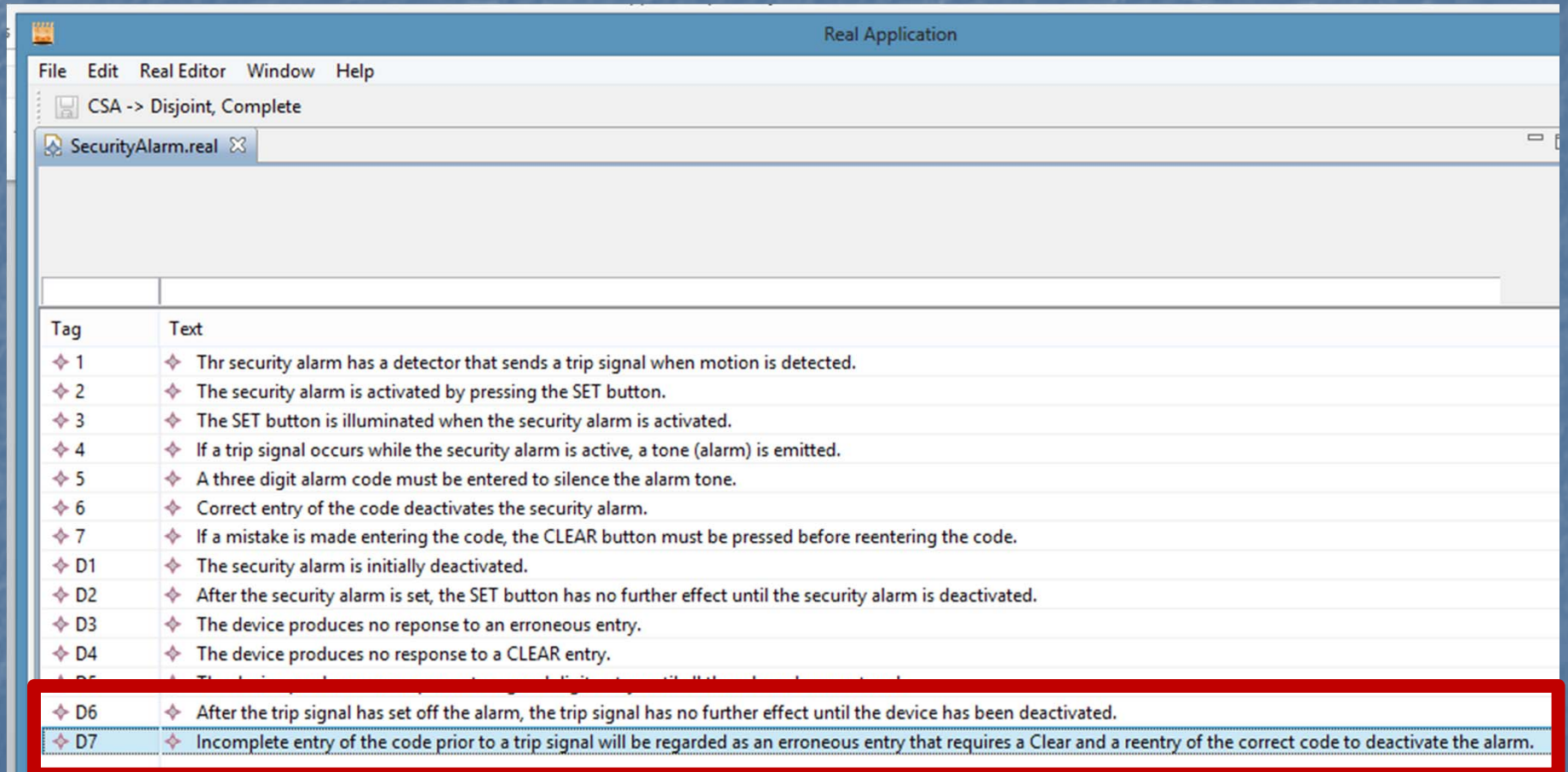
Tag	Text
1	Thr security alarm has a detector that sends a trip signal when motion is detected.
2	The security alarm is activated by pressing the SET button.
3	The SET button is illuminated when the security alarm is activated.
4	If a trip signal occurs while the security alarm is active, a tone (alarm) is emitted.
5	A three digit alarm code must be entered to silence the alarm tone.
6	Correct entry of the code deactivates the security alarm.
7	If a mistake is made entering the code, the CLEAR button must be pressed before reenterin...
D1	The security alarm is initially deactivated.
D2	After the security alarm is set, the SET button has no further effect until the security alarm i...
D3	The device produces no reponse to an erroneous entry.
D4	The device produces no response to a CLEAR entry.
D5	The device produces no response to a good digit entry until all three have been entered.

Enumeration Continued

Sequences of Length 3

Name	Length	Prefix	Action	Responses	Response trace	Legal	Equivalence
◆ S.G	◆ 2	◆ S	◆ G	◆ LightOn	◆ D5	◆ yes	◆
◆ S.S	◆ 2	◆ S	◆ S	◆ AlarmOff	◆ D2	◆ yes	◆ S
◆ S.T	◆ 2	◆ S	◆ T	◆ AlarmOn	◆ 4	◆ yes	◆
◆ S.B.B	◆ 3	◆ S.B	◆ B	◆	◆ D3	◆ yes	◆ S.B
◆ S.B.C	◆ 3	◆ S.B	◆ C	◆	◆ 7, D4	◆ yes	◆ S
◆ S.B.G	◆ 3	◆ S.B	◆ G	◆	◆ 7	◆	◆
◆ S.B.S	◆ 3	◆ S.B	◆ S	◆	◆ D2	◆ yes	◆ S.B
◆ S.B.T	◆ 3	◆ S.B	◆ T	◆ AlarmOn	◆ 4	◆ yes	◆
◆ S.G.B	◆ 3	◆ S.G	◆ B	◆	◆ D3	◆ yes	◆ S.B
◆ S.G.C	◆ 3	◆ S.G	◆ C	◆	◆ D4	◆ yes	◆ S
◆ S.G.G	◆ 3	◆ S.G	◆ G	◆	◆ D5	◆ yes	◆
◆ S.G.S	◆ 3	◆ S.G	◆ S	◆	◆ D2	◆ yes	◆ S.G
◆ S.G.T	◆ 3	◆ S.G	◆ T	◆ AlarmOn	◆ 4, D7	◆ yes	◆ S.T
◆ S.T.B	◆ 3	◆ S.T	◆ B	◆	◆ D3	◆ yes	◆ S.B.T
◆ S.T.C	◆ 3	◆ S.T	◆ C	◆	◆ D4	◆ yes	◆ S.T
◆ S.T.G	◆ 3	◆ S.T	◆ G	◆	◆ D5	◆ yes	◆
◆ S.T.S	◆ 3	◆ S.T	◆ S	◆	◆ D2	◆ yes	◆ S.T
◆ S.T.T	◆ 3	◆ S.T	◆ T	◆	◆ D6	◆ yes	◆ S.T
◆ S.B.T.B	◆ 4	◆ S.B.T	◆ B	◆	◆ D3	◆ yes	◆ S.B.T
◆ S.B.T.C	◆ 4	◆ S.B.T	◆ C	◆	◆ 7, D4	◆ yes	◆ S.T
◆ S.B.T.G	◆ 4	◆ S.B.T	◆ G	◆	◆ 7	◆	◆

More Derived Requirements



Real Application

File Edit Real Editor Window Help

CSA -> Disjoint, Complete

SecurityAlarm.real

Tag	Text
1	The security alarm has a detector that sends a trip signal when motion is detected.
2	The security alarm is activated by pressing the SET button.
3	The SET button is illuminated when the security alarm is activated.
4	If a trip signal occurs while the security alarm is active, a tone (alarm) is emitted.
5	A three digit alarm code must be entered to silence the alarm tone.
6	Correct entry of the code deactivates the security alarm.
7	If a mistake is made entering the code, the CLEAR button must be pressed before reentering the code.
D1	The security alarm is initially deactivated.
D2	After the security alarm is set, the SET button has no further effect until the security alarm is deactivated.
D3	The device produces no response to an erroneous entry.
D4	The device produces no response to a CLEAR entry.
D5	The device produces no response to a CLEAR entry.
D6	After the trip signal has set off the alarm, the trip signal has no further effect until the device has been deactivated.
D7	Incomplete entry of the code prior to a trip signal will be regarded as an erroneous entry that requires a Clear and a reentry of the correct code to deactivate the alarm.

Enumeration Continued

Sequences of Length 4

Name	Length	Prefix	Action	Responses	Response trace	Legal	Equivalence
◆ S.T.S	◆ 3	◆ S.T	◆ S	◆	◆ D2	◆ yes	◆ S.T
◆ S.T.T	◆ 3	◆ S.T	◆ T	◆	◆ D6	◆ yes	◆ S.T
◆ S.B.T.B	◆ 4	◆ S.B.T	◆ B	◆	◆ D3	◆ yes	◆ S.B.T
◆ S.B.T.C	◆ 4	◆ S.B.T	◆ C	◆	◆ 7, D4	◆ yes	◆ S.T
◆ S.B.T.G	◆ 4	◆ S.B.T	◆ G	◆	◆ 7	◆	◆
◆ S.B.T.S	◆ 4	◆ S.B.T	◆ S	◆	◆ D2	◆ yes	◆ S.B.T
◆ S.B.T.T	◆ 4	◆ S.B.T	◆ T	◆	◆ D6	◆ yes	◆ S.B.T
◆ S.G.G.B	◆ 4	◆ S.G.G	◆ B	◆	◆ D3	◆ yes	◆ S.B
◆ S.G.G.C	◆ 4	◆ S.G.G	◆ C	◆	◆ D4	◆ yes	◆ S
◆ S.G.G.G	◆ 4	◆ S.G.G	◆ G	◆	◆ 6	◆ yes	◆ <lamb...
◆ S.G.G.S	◆ 4	◆ S.G.G	◆ S	◆	◆ D2	◆ yes	◆ S.G.G
◆ S.G.G.T	◆ 4	◆ S.G.G	◆ T	◆	◆ 4, D7	◆ yes	◆ S.T
◆ S.T.G.B	◆ 4	◆ S.T.G	◆ B	◆	◆ D3	◆ yes	◆ S.B.T
◆ S.T.G.C	◆ 4	◆ S.T.G	◆ C	◆	◆ D4	◆ yes	◆ S.T
◆ S.T.G.G	◆ 4	◆ S.T.G	◆ G	◆	◆ D5	◆ yes	◆
◆ S.T.G.S	◆ 4	◆ S.T.G	◆ S	◆	◆ D2	◆ yes	◆ S.T.G
◆ S.T.G.T	◆ 4	◆ S.T.G	◆ T	◆	◆ D6	◆ yes	◆ S.T.G
◆ S.T.G.G.B	◆ 5	◆ S.T.G.G	◆ B	◆	◆ D3	◆ yes	◆ S.B.T
◆ S.T.G.G.C	◆ 5	◆ S.T.G.G	◆ C	◆	◆ D4	◆ yes	◆ S.T
◆ S.T.G.G.G	◆ 5	◆ S.T.G.G	◆ G	◆	◆ 3, 5, 6	◆ yes	◆ <lamb...
◆ S.T.G.G.S	◆ 5	◆ S.T.G.G	◆ S	◆	◆ D2	◆ yes	◆ S.T.G

Enumeration Completed

Sequences of Length 5

Name	Length	Prefix	Action	Responses	Response trace	Legal	Equivalence
◆ S.T.S	◆ 3	◆ S.T	◆ S	◆	◆ D2	◆ yes	◆ S.T
◆ S.T.T	◆ 3	◆ S.T	◆ T	◆	◆ D6	◆ yes	◆ S.T
◆ S.B.T.B	◆ 4	◆ S.B.T	◆ B	◆	◆ D3	◆ yes	◆ S.B.T
◆ S.B.T.C	◆ 4	◆ S.B.T	◆ C	◆	◆ 7, D4	◆ yes	◆ S.T
◆ S.B.T.G	◆ 4	◆ S.B.T	◆ G	◆	◆ 7	◆	◆
◆ S.B.T.S	◆ 4	◆ S.B.T	◆ S	◆	◆ D2	◆ yes	◆ S.B.T
◆ S.B.T.T	◆ 4	◆ S.B.T	◆ T	◆	◆ D6	◆ yes	◆ S.B.T
◆ S.G.G.B	◆ 4	◆ S.G.G	◆ B	◆	◆ D3	◆ yes	◆ S.B
◆ S.G.G.C	◆ 4	◆ S.G.G	◆ C	◆	◆ D4	◆ yes	◆ S
◆ S.G.G.G	◆ 4	◆ S.G.G	◆ G	◆	◆ 6	◆ yes	◆ <lamb...
◆ S.G.G.S	◆ 4	◆ S.G.G	◆ S	◆	◆ D2	◆ yes	◆ S.G.G
◆ S.G.G.T	◆ 4	◆ S.G.G	◆ T	◆	◆ 4, D7	◆ yes	◆ S.T
◆ S.T.G.B	◆ 4	◆ S.T.G	◆ B	◆	◆ D3	◆ yes	◆ S.B.T
◆ S.T.G.C	◆ 4	◆ S.T.G	◆ C	◆	◆ D4	◆ yes	◆ S.T
◆ S.T.G.G	◆ 4	◆ S.T.G	◆ G	◆	◆ D5	◆ yes	◆
◆ S.T.G.S	◆ 4	◆ S.T.G	◆ S	◆	◆ D2	◆ yes	◆ S.T.G
◆ S.T.G.T	◆ 4	◆ S.T.G	◆ T	◆	◆ D6	◆ yes	◆ S.T.G
◆ S.T.G.G.B	◆ 5	◆ S.T.G.G	◆ B	◆	◆ D3	◆ yes	◆ S.B.T
◆ S.T.G.G.C	◆ 5	◆ S.T.G.G	◆ C	◆	◆ D4	◆ yes	◆ S.T
◆ S.T.G.G.G	◆ 5	◆ S.T.G.G	◆ G	◆	◆ 3, 5, 6	◆ yes	◆ <lamb...
◆ S.T.G.G.S	◆ 5	◆ S.T.G.G	◆ S	◆	◆ D2	◆ yes	◆ S.T.G.G
◆ S.T.G.G.T	◆ 5	◆ S.T.G.G	◆ T	◆	◆ D6	◆ yes	◆ S.T.G.G

Canonical Sequence Analysis

Name	State var values
◆ <lambda>	◆ Alarm = [ANY], Code = [ANY], Invoked = No
◆ S	◆ Alarm = Off, Code = None, Invoked = Yes
◆ S.B	◆ Alarm = Off, Code = Error, Invoked = Yes
◆ S.G	◆ Alarm = Off, Code = 1_OK, Invoked = Yes
◆ S.T	◆ Alarm = On, Code = None, Invoked = Yes
◆ S.B.T	◆ Alarm = On, Code = Error, Invoked = Yes
◆ S.G.G	◆ Alarm = Off, Code = 2_OK, Invoked = Yes
◆ S.T.G	◆ Alarm = On, Code = 1_OK, Invoked = Yes
◆ S.T.G.G	◆ Alarm = On, Code = 2_OK, Invoked = Yes

Requirements Interfaces Stimuli Outputs Named Responses Predicates Mapping

Properties ✕

Canonical Sequence Analysis

- Construct a table and list the canonical sequences in the order enumerated.
- Define one or more state variables such that each canonical sequence corresponds to a unique combination of values.

State Machine Spec

- Start with state variables from canonical sequence analysis.
- Invent state variables required to compute abstractions*.
- Invent state variables required to compute abstract responses.
- Simplify state data, if possible.
- Recast stimulus/condition/response table (Black Box) to stimulus/start state/end state/response table (State Box).

**Removal of some abstractions may be deferred to the clear box specification.*

State Box Specification Excerpt:

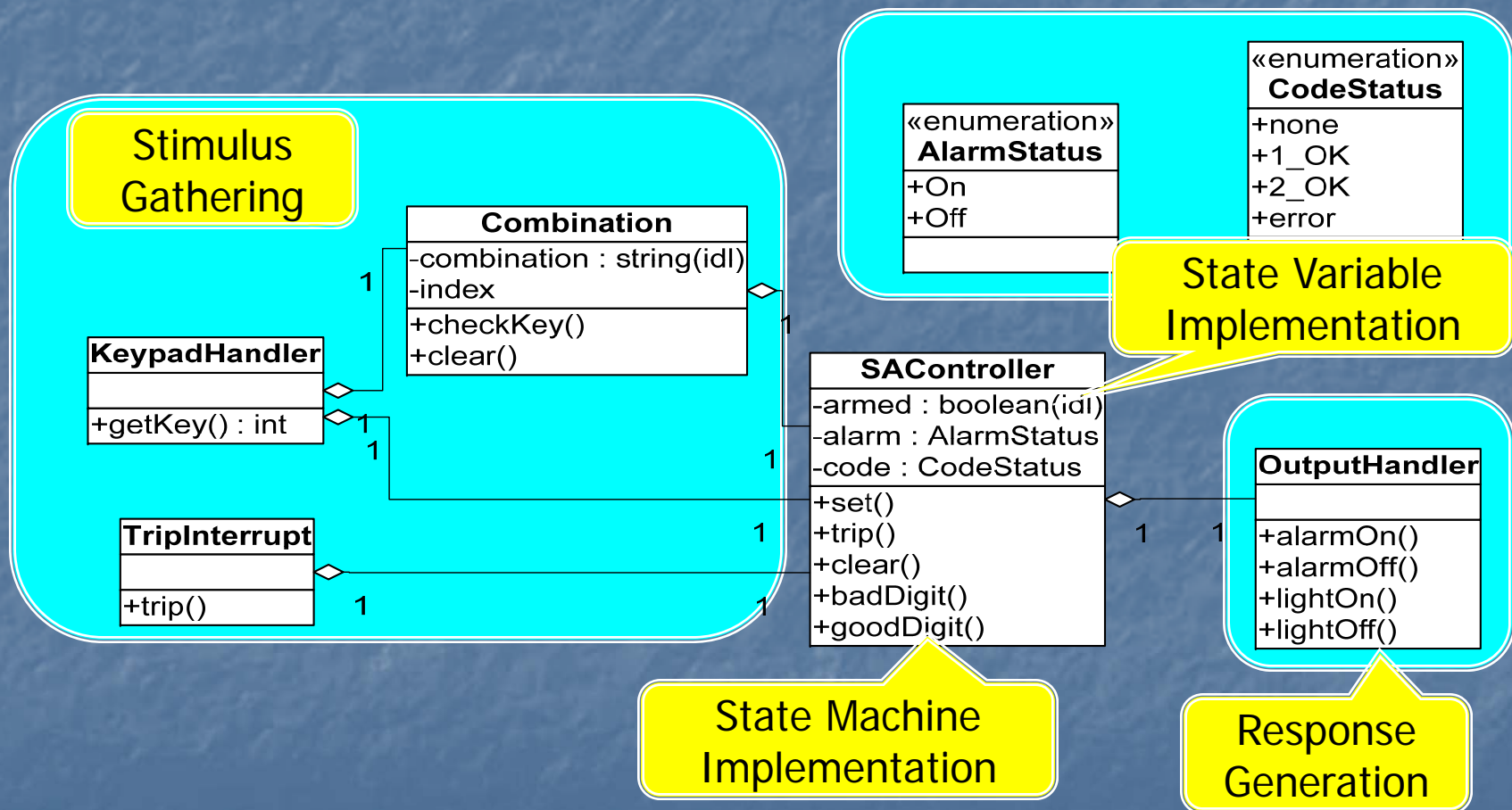
Stimulus: G, Good Digit

Initial State			Response	Final State		
<u>Alarm</u>	<u>Code</u>	<u>Invoked</u>		<u>Alarm</u>	<u>Code</u>	<u>Invoked</u>
[ANY]	[ANY]	No	illegal			
Off	None	Yes	<u>LightOn</u>	Off	None	Yes
Off	Error	Yes	illegal			
Off	1_OK	Yes	null	Off	1_OK	Yes
On	None	Yes	null	On	None	Yes
On	Error	Yes	illegal			
Off	2_OK	Yes	null	Off	2_OK	Yes
On	1_OK	Yes	null	On	1_OK	Yes
On	2_OK	Yes	null	On	2_OK	Yes

Implementation

- Define high level architecture.
- Define stimulus gathering implementation.
- Define response generation implementation.
- Map state data to specific architectural components.
- Define implementation of each transition in State Box table.
- Map implementation of State Box table entries to architecture.

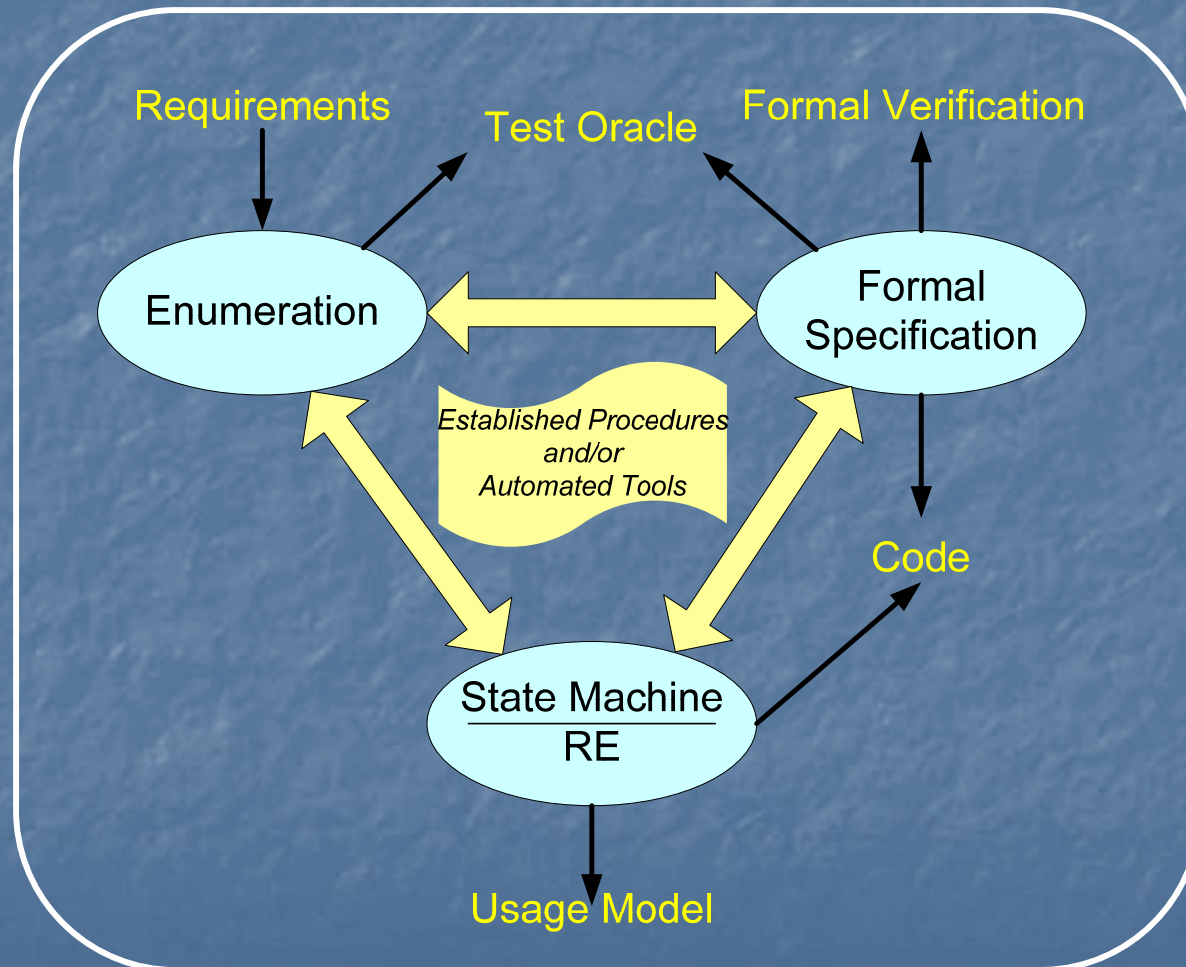
State Machine \Rightarrow Code



Rigorous Specification and Software Development Process

- Automatically transform enumeration to state machine or formal spec
- Generate code and test usage model from state machine spec
- Perform formal verification based on spec
- Generate test oracle from spec

Overview



Summary and Conclusion

- Sequence-Based Specification is a method for producing **consistent, complete, and traceably correct** software specifications.
- The method is based on a simple **sequence enumeration** procedure and basic requirements analysis skills.
- The results can be freely converted to state machines and other formal representations for
 - **Automatic Code Generation**
 - **Formal Verification via Theorem Provers**
 - **Test Case Generation and Results Analysis**