Use Cases

- Specify “actors” and how they interact with various component parts of a system
  - This is an external “black box” view of a system
- System is a collection of “use cases” (i.e. Capabilities provided to users/actors)
- Represented using diagrams and schemas
  - Diagrams show flow of “uses” between actors and use cases
  - Schemas are more formal non-pictorial definitions
Example Use Case diagram

Use Case schema

Use case: Heat Cooking Tank

Description: Heat a cooking tank to the temperature prescribed by the recipe of the juice to be mixed, and keep it at that temperature for the time prescribed by the recipe.

Reads: Cooking tank, Batch, Recipe.

Changes:
In: Operator: Batch ID
Thermometer: Current temperature.

Out: Thermometer: Temperature request.
Heater: switch on, switch off.
Operator: heating finished.

Assumes: Juice is present in tank.

Results: Juice has been kept at desired temperature for the desired time.

Transactions:
- The system switched on the cooking tank heater.
- The system checks the cooking tank thermometer every 10 seconds. When the desired temperature is reached, the heater is switched off, when the temperature is too low, the heater is switched on again.
- When the end temperature is reached, the heater is switched off and a message is sent to the operator.
5.2.10 Administrator login

Description
An Administrator logs into the ID Station by inserting their Token in the Admin Token Reader.

Stimulus
A Token is inserted in the Admin Token Reader.

Assumptions

SucLogOn.AscQueueAcc
- The ID Station is quiescent (no other access attempts, configuration changes or start-up activities are in progress).

SucLogOn.AscSecure
- The door is closed and locked.

SucLogOn.AscValidate
- The card inserted by the Administrator has a valid Authorization Certificate.

Success Conditions

SucLogOn.BusLogOn
- The ID Station is available for use by the Administrator, in that it will respond to the commands allowed to that Administrator as defined by the privileges in the Authorization Certificate read from the Token, and the Configuration data held on the ID Station.

SucLogOn.BusSecure
- The door is closed and locked.

SucLogOn.BusAudit
- The following events have been recorded in the Audit Log (in any order), and the existing audit records are preserved:
  - log-on by Administrator
  - insertion of card
  - reading data from card (possibly multiple failures, but at least one success)

Failure Conditions

SucLogOn.FailReadCard
- The card inserted by the Administrator does not allow all its necessary data to be successfully read, possibly due to being incorrectly inserted in the first place, being a faulty card, having the incorrect information on it, or being removed before all the information has been read. The set of data to be read is at least:
  - Authorization Certificate

SucLogOn.FailAudit
- Audit files cannot be successfully written. Result: the Door is locked and the system is shutdown.

SucLogOn.FailAudPreserve
- Space for audit files has been exhausted. Result: the oldest audit records are overwritten with the new audit records, and an alarm is relayed to the Guard.

Constraints

SucLogOn.ConNo_interact
- No ID Station shutdown or User use will be allowed during this scenario.

Rationale

SucLogOn.ConValidAdmin
- Only the Authorization Certificate is checked, because we assume that the purpose of the Authorization Certificate is to control access to the workstations within the enclave, and for these purposes the ID Station acts as a workstation. The ID, I&A and Privilege Certificates will have been used to gain access to the enclave.

5.2.11 Administrator logoff

Description
An Administrator logs off the ID Station.

Stimulus
The Token is removed from the Admin Token Reader.

Assumptions
5.2.1 Administrator logs off

**Description**
An Administrator logs off the ID Station.

**Stimulus**
The Token is removed from the Admin Token Reader.

**Assumptions**

**Success End-conditions**
- ScLogOff.SucLoggedOff
  The ID Station is unavailable for use by anyone at the console; it will respond to no commands typed in at the console.
- ScLogOff.Suc.Secure
  The door is closed and locked.
- ScLogOff.Suc.Audit
  The following events have been recorded in the Audit Log (in any order), and the existing audit records are preserved:
  - Log-off by Administrator

**Failure Conditions**
- ScLogOff.Fail.Audit
  Audit files cannot be successfully written. Result: the Door is locked and the system is shutdown.
- ScLogOff.Fail.AuditPreserve
  Space for audit files has been exhausted. Result: the oldest audit records are overwritten with the new audit records, and an alarm is raised to the Guard.

**Constraints**

15. Change of audit alarm state (to alarming or to silent)

5.2 Scenarios

5.2.1 User gains allowed initial access to Enclave

**Description**
A User who should be allowed access to the enclave is given access, making use of biometric authentication.

**Stimulus**
User inserts a smartcard into the smartcard reader.

**Assumptions**
- ScGainInitial.Aes.ValidStart
  The ID Station has valid start-up data.
- ScGainInitial.Aes.ValidConfig
  The ID Station has a valid data configuration.
The user is outside the enclave; the door is closed and locked.

**SoGearInInit.AskSecure**

The card inserted by the user has a valid ID Certificate, I&A Certificate, and privilege Certificate, and the card inserted by the user has a valid fingerprint template that matches the fingerprint of the user's finger.

**SoGearInInit.Ask,PopAC**

The card inserted by the user does not have a valid, current Authorization Certificate.

**Success End-conditions**

**SoGearInInit.Suc,UserCard**

The user has possession of the card he originally inserted.

**SoGearInInit.Suc,Card,AC**

The card inserted by the user contains a current, valid Authorization Certificate with:

- Validity time from now until expiration:
- Security level equal to the minimum of the security level defined in the ID Station configuration data and the security level in the Permission Certificate on the card inserted by the user.

**SoGearInInit.Suc,PermCerts**

The card inserted by the user contains the same, unchanged ID Certificate, I&A Certificate, and Privilege Certificate it had at the beginning of the session.

**SoGearInInit.Suc,UserWin**

The user is in the Enclave.

**SoGearInInit.Suc,Locked**

The entrance door is closed and locked.

**SoGearInInit.Suc,Alert**

The following events have been recorded in the Audit Log (in any order), and the existing audit records are preserved:

- Insertion of card
- Removal of card

**Failure Conditions**

**SoGearInInit.Fai,ReadCard**

The card inserted by the user does not allow all its data to be successfully read, possibly due to being incorrectly inserted in the first place, being a faulty card, having the incorrect information on it, or being removed before all the information has been read. The set of data to be read is at least:

- ID Certificate
- I&A Certificate
- Privilege Certificate
- Fingerprint Template (contained in the I&A Certificate)

**SoGearInInit.Fai,Process**

A matching fingerprint has not been read, possibly due to no finger being presented to the fingerprint reader within 5 seconds of the display requesting a fingerprint, or the fingerprint not being successfully read within 3 seconds of the display requesting a fingerprint, or the fingerprint that was successfully read not being successfully matched to the template read from the card. The value X shall be taken from configuration data of the ID Station.

**SoGearInInit.Fai,WriteCard**

The card originally inserted by the User does not allow a new Authorization Certificate to be successfully written, possibly due to being incorrectly inserted in the first place, being a faulty card, or being removed.
Stakeholders and Questions

- Users of the system
  - How do I want it to behave
- Developers of the system
  - What capabilities do I need to implement
- Customers
  - What are these capabilities and behaviors worth
Message Sequence Diagrams

- Sometimes called “ladder charts”
- Represent a particular sequence of messages exchanged between entities
- Popular in object-oriented methods to represent communications between objects
- Shows one particular communication sequence in one run of the system
- Shows behavior as well as communication
- Can be extended with conventions to represent looping, casing, timeouts, synchronization, global conditions across different entities, delayed message reception, etc.

Example Message Sequence Diagram

![Message Sequence Diagram](Image)
Stakeholders and Questions

• Users
  – What behaviors do I want/need
• Developers
  – What interactions need to be supported
  – What are the entities
  – What operations are to be supported
• Inspectors?
  – Are there safety issues
• ??

Class Diagram

• In widespread use. Consists of
  – Name
  – Attributes
  – Operations/Methods
  – Associations
    » Cardinalities
    » Annotations
    » Qualifiers
    » Interfaces
    » More….. (much more)
**Class diagram for juice plant**

- Additional object class
  - would be modeled by a control process in dataflow models.

**Collaboration Diagrams**

- Popular in object-oriented methods to represent message exchanges between objects
- Object specification augmented by annotations that represent dataflows between the communicating objects
- Differ from other notations
  - Nodes represent objects, not activities (as in DFDs, activity diagrams, activity charts, and block diagrams)
  - Nodes represent object instances, not object classes
- As in sequence diagrams, represent the sequence of messages in one particular scenario, not all possible communications scenarios.
Collaboration Diagrams

Representing Other Types of Things

- Data, Objects, Artifacts
  - These are clearly secondary in all of the above diagrams
  - Often are more important than functional view
  - Harder to depict diagrammatically

- Process artifacts and views
  - Primary interest of management and customer stakeholders for much of the time
  - Typical questions:
    » What is the (development) plan? schedule?
    » Are we almost done?
    » What are we going to do next?
    » What if Joe quits?
  - Different representations are needed to reply effectively
Primitive Process Representations

- PERT/CPM Charts
- Gantt Charts

Pert and CPM Charts

- Depict the process as a network of tasks
- Each step is a circle
- Incoming arrows are steps that must complete before this one
- Outgoing arrows are steps that might follow this one
- Each step has a time estimate
- No loops allowed
- So that maximum “flow time” can be computed
  - Along the “critical path”
- Early management tool
- Very naive and oversimplified view
  - no loops!!
  - Simplicity is its strength and weakness
Gantt Charts

• Familiar milestone charts, progress charts, ....
• Time represented along a horizontal axis
• Each task (person, ...) represented by a solid bar plotted against the time line
• Bar starts at “start time” and ends at “end time”
• Key Milestones represented by triangles placed along the bar
• Shows how tasks juxtapose
• Shows who should be doing what at all times
• Shows how product is supposed to evolve over time
• Effective for spotting schedule slippages

Example Gantt Chart

<table>
<thead>
<tr>
<th>Field Support</th>
<th>Test</th>
<th>Builder #1</th>
<th>Builder #2</th>
<th>Customer Liaison</th>
<th>Design</th>
<th>Get money</th>
<th>Document</th>
<th>Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2005 2006 2007 2008 2009
Multirepresentation Systems

- Have seen that different representations are of different uses
- One diagram may be useful in different ways to different stakeholders
- But most stakeholders require a variety of diagrams
- Several different diagrams can be expected to be needed to satisfy the different stakeholders
- Problems with different views/diagrams
  - Are they all representing the same software product?
  - How to assure that they are all consistent with each other?
  - If the product changes, then ALL views must change correspondingly
Plato’s Cave

STATEMATE

• Focus on Statecharts—an enhancement of FSM's
  – Augmented by other views (e.g. activity Diagrams)
• Key feature is maintenance of consistency among views
• Rigorously defined semantics
  – Including specification of needed consistency
• References
• Commercially available software system
Multiple Views in Statemate

- Rationale for multiple views: Too much information in a single diagram creates clutter, confusion, defeats clarity.

- Advantage of multiple views: Each represents a different viewpoint, different model, with a different diagram.

- Disadvantage: Reader needs to synthesize views, assure that they are really consistent with each other.

- Multiple views in Statemate:
  --Module Charts (a hierarchy representing capabilities)
  --Activity Charts (hierarchical dataflow charts)
  --Statecharts (hierarchical finite state machines)
  --Sequence Charts

- All facilitated by a slick user interface.

- Statemate views depict some different views, but also overlap with each other: facilitates cross-checking for consistency and easier comprehension.
Redundancy

• Specifying or doing the same thing more than once
• Usually considered undesirable in computing
• Typically regarded as desirable in engineering
• Particularly useful in safety engineering
  – NASA 5-way redundancy
• Can help assure that multiple views are not inconsistent
  – Different views should not be inconsistent about things in their intersection
Module Charts

- Hierarchy shown by
  - Indentation
  - Nesting module-charts inside each other

- How many levels of nesting without losing clarity?
Activity Chart

- A Data Flow Graph--Hierarchical
  --focus (depicted by solid boxes) on functions
  --Arrows depict data flows

- All of this helps user/reader to associate features of one with features of the other

- DFG incorporates Control Box (like in Kepler):
  --Control box (rounded): at most one per activity
  --Suggests need to depict how and when data will flow among functions--not just what
    **Example: How to represent an activity consisting of a set of cases with DFD's?**
  --Dashed arrows represent flow of control information (eg. signals, commands, status reporting/changing)
    >>This is redundant with arrows in Module Charts

- This, in turn, anticipates new view represented using the third type of chart
Use Case

Use Case Elaboration
Statecharts

- Extension of basic notion of FSM

- FSM's are effective in modeling systems that are
  --clearly and accurately modelled as being in only one of a
  finite number of states at a time
  --considered to move from state to state driven by events
  drawn from a finite set of possibilities

- Statecharts add some features to what basic FSM’s can represent
  --Hierarchy:
    >>Keeps charts from getting too big, hard to understand
  --ANDing and ORing of states:
    >> to model simultaneously being in >1 state
    >>example: elevator in moving/not or doors_open/not
  --Elaborate specification of transition conditions

- Correlation with Activity Charts helps comprehensibility
Add Activities and Actions

- **Activities**
  - Associated with a state
  - Start when the state is entered
  - Take time to complete
  - Interruptible
- **Actions**
  - Associated with a transition
  - Take an insignificant amount of time to complete
  - Non-interruptible

Activities and Guards in Statecharts

- **Activities**
  - An activity can also send an event
- **Transitions**
  - A transition may have a guard conditions as well as an event specified
  - Transitions can also specify an action that happens in response to the receipt of an event
**Statechart**

- **Initialize**
  - do: Initialize course object
  - cancelCourse

- **Unassigned**
  - do: Assign professor to course
  - Unassigned

- **Open**
  - entry: Register a student
  - addStudent
  - numStudents = 0
  - cancelCourse

- **Canceled**
  - do: Send cancellation notices
  - cancelCourse

- **Closed**
  - do: Report course is full
  - numStudents = 10

- **RegistrationComplete**
  - do: Generate class roster

**Statechart with Nested States**

- **Initialize**
  - RegistrationComplete
  - do: Generate class roster

- **Unassigned**
  - do: Assign professor to course
  - Unassigned

- **Open**
  - registration closed numStudents > 3
  - entry: Register a student
  - addStudent
  - numStudents = 0
  - cancelCourse

- **Closed**
  - do: Report course is closed
  - numStudents = 10
Message Sequence Chart View

- Very much like what we have seen before
- Vertical red lines augment diagram with timing information
  - Simultaneous activities
  - Specification of time lag between messages
1. Charging subsystem indicates electrical system load.
2. Customer requests the rear defog system to turn on.
3. The rear defog led is illuminated.
4. The rear defog relay is enabled.
5. Vehicle speed increases past the high speed threshold - the relay can be enabled indefinitely.
6. Load condition one exists, the system starts modulating of the rear defog relay.
7. Modulation starts at the calibrated duty cycle, the relay is disabled
8. Modulation continues, the relay is enabled.
9. Modulation continues, the relay is disabled.

_Time: 0.000000_ | [DEFOG DRIVE OFF-0x1]
_RSEFOG_SR_STATUS_ | [DEFOG DRIVE OFF-0x0]
_RSEFOG_SR_STATUS_ | [DEFOG DRIVE OFF-0x1]

_Time: 10.000000_ | [DEFOG DRIVE OFF-0x0]
[LOAD_RHS_IP--LR_LEVEL] | [DEFOG DRIVE OFF-0x1]

_Time: 18.000000_ | [DEFOG DRIVE OFF-0x0]
Statemate Support Environment

- Tools to support drawing/changing diagrams
- Tools to support input of textual information through forms/templates
- Diagrams enhanced by use of color (?)
- Tools to generate simulations automatically --support "stepping through" the system
- System assures consistency among the diagrams --changes automatically depicted consistently in all diagrams
- Tools to automatically generate Ada code that emulate Statechart behavior

Template Input
Cross-Checking/Redundancy Checking
Statemate Weaknesses

• Does not seem to scale all that well
  --Hierarchy depicted by nesting all on one 2-dimensional surface

• Data still treated as secondary

• Focus still on functionality
  --Other characteristics and views are worth thinking about too:
    >>Speed
    >>Implementation approaches and issues
    >> ...

UML (Unified Modeling Language): The Latest (?)

• Merger of Booch, Rumbaugh, Jacobsen work
  --“The three amigos”
  --All worked for Rational (now IBM)

• Comprehensive suite of diagrams

• Some semantics in place
  --But not all
  --International task forces (!) working on this

• Process for using them was developed too
  --Rational Unified Process (RUP)

• UML blew away the opposition
  --Not clear this was good
(Some) UML representations

- Class Diagrams
- Use Cases
- Sequence Diagrams
- Package Diagrams
- State Diagrams
- Activity Diagrams
- Collaboration Diagrams
- Deployment Diagrams

Different combinations used by different users for different projects

Major UML Problems/Objections

- What are semantics of all of these features of all of these diagrams?
  - Task forces working on them
  - Maybe there is just too much there (?)
- Diagram semantics overlap
  - Which diagram to use when
  - How to tell when they are inconsistent
- Extensibility
  - Use of “stereotype” feature
  - How to reconcile semantics of new features with existing ones
UML Tries to cover everything

• A diagram type for everything
• But they are not well connected to each other
• Few rules on what to use when
• Long reach with uncertain grasp

Evaluation of Diagrammatic Approach

• Pictures considerably aid clarity
• Significantly reduce possible ambiguity
• Increasingly strong semantics of increasingly intricate pictures yield increasing completeness and increasing assurance of consistency
• Increasingly intricate pictures are decreasingly clear, decreasingly modifiable
  --Modern approach is to provide tools to help
• In place of one intricate and complex diagram, many systems substitute a set of coordinated diagrams, each of which is relatively simple (e.g., Statemate)
  --leads to problems in assuring consistency of diagrams, but tools can help here too

BUT ALSO:
• Most diagrams help depict functionality, but not other characteristics, (e.g., data, process, etc.)