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 specification is a collection of “use cases” (ie. Capabilities provided to users/actors)
- Represented using diagrams and schemas
  - Diagrams show flow of “uses” between actors and use cases
  - Schemas are more formal, structured, non-pictorial definitions
Example Use Case diagram

Example Structure of a Use Case Schema

• Schema name
• Description
• Assumptions
• Success End Conditions
• Contingencies and Exceptions
• Stimuli
• Inputs
• Outputs
• Constraints

This is Structured Natural Language
Could correspond to an equivalent diagram
Example 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, Temperature.
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.

A More Serious Set of Use Cases: Access to a Secure Facility

• Want to assure that only authorized people are able to enter the facility
• Use token reader, biometrics to qualify people
• Control door
• Control alarms
• Support security console
• Log all transactions
5.2.10 Administrator log in

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

SUILogOn,Admin,Secure
The door is closed and locked.

SUILogOn,Admin,ValidToken
The card inserted by the Administrator has a valid Authorization Certificate.

Success End-conditions
SUILogOn,Success
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.

SUILogOn,Fail,Secure
The door is closed and locked.

SUILogOn,Fail,Audit
The following events have been recorded in the Audit Log (in any order), and the existing audit records are preserved:
- Log-in by Administrator
- Insertion of card
- Reading data from card (possibly multiple failures, but at least one success)

Failure Conditions
SULogOn,Fail,ReadCard
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

SULogOn,Fail,Audit
Audit file cannot be successfully written. Result: the Door is locked and the system is shutdown.

SULogOn,Fail,Audit,Preserve
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
SULogOn,Con,Noninterfere
No ID Station shutdown or User use will be allowed during this scenario.

Rationale
SULogOn,Res,ValidAdmin
Only the Authorization Certificate is checked, because we assume that the purpose of the Authorization Certificates is to control access to the workstations within the environment, and for these purposes the ID Station acts as a workstation. The ID, ISA and Privilege Certificates will have been used to gain entry to the environment.

5.2.11 Administrator log off

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**

- **Suc.LogOff**
  - The ID Station is unavailable for use by anyone at the console; it will respond to no commands typed in at the console.

- **Suc.LogOff, Suc.Secret**
  - The door is closed and locked.

- **Suc.LogOff, 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**

- **Fail.LogOff, Fail.Audit**
  - Audit files cannot be successfully written. Result: the Door is locked and the system is shutdown.

- **Fail.LogOff, Fail.Audit, Fail.Preserve**
  - 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**

- **Suc.Gain, Suc.ValidStart**
  - The ID Station has valid start-up data.

- **Suc.Gain, Suc.ValidConfig**
  - The ID Station has a valid data configuration.
The user is outside the elevator; the door is closed and locked.

Success End-conditions

The card inserted by the User does not have a valid Authorization Certificate.

Failure Conditions

The card inserted by the User contains a valid Authorization Certificate and is inserted correctly; however, the User does not have a valid ID Certificate, a valid I&A Certificate, or a valid Privilege Certificate.

The card inserted by the User contains an invalid Authorization Certificate.

The User has inserted the wrong card.

The card inserted by the User contains a valid Authorization Certificate, but the User does not have a valid ID Certificate, I&A Certificate, or Privilege Certificate.

The User has inserted the card incorrectly.

The User has inserted the card correctly, but the User does not have a valid ID Certificate, I&A Certificate, or Privilege Certificate.

The User has inserted the card correctly, but the User does not have a valid I&A Certificate.

The User has inserted the card correctly, but the User does not have a valid Privilege Certificate.

The User has inserted the card correctly, but the User does not have a valid Fingerprint Template.

The User has inserted the card correctly, but the User does not have a valid FingerPrint Reader.

The User has inserted the card correctly, but the User does not have a valid FingerPrint Reader or Fingerprint Reader.

The User has inserted the card correctly, but the User does not have a valid FingerPrint Reader, Fingerprint Reader, or Fingerprint Reader.
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.

Juice Tank Message Sequence Diagram
More Complex Message Sequence Chart

1. Customer requests the rear defog system on.
2. The rear defog led is turned on.
3. The rear defog relay is enabled.
4. After being enabled for Low Speed Max On Time seconds, the rear defog relay is disabled.
5. The rear defog relay led is turned off.

Simultaneous messages
More Complex Message Sequence Chart

**Simultaneous messages**

1. Customer requests the rear defog system on.
2. The rear defog led is turned on.
3. The rear defog relay is enabled.
4. After being enabled for Low Speed Max On Time seconds, the rear defog relay is disabled.
5. The rear defog relay led is illuminated.

**Triggering Event**

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 modulation 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.
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.
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
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
Multiple Views

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
Multiple Views in Statemate

Module view

Statechart view

Activity view

Sequence Chart

Textual view

The Importance of Redundancy
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 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

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

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

- **Open**
  - entry: Register a student
    - addStudent
    - cancelCourse
    - registration closed (numStudents < 3)
    - registration closed (numStudents > 3)

- **Closed**
  - do: Report course is full
  - registration closed (numStudents > 3)

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

- **addStudent**
  - [numStudents = 0]

- **cancelCourse**
  - [numStudents = 10]
Statechart with Nested States

---

superstate

substate

- Initialize
- RegistrationComplete
  - do: Generate class roster
- Unassigned
  - do: Assign professor to course
  - Add student / numStudents = 0
- Open
  - entry: Register a student
    - addStudent
    - if numStudents = 0
    - if numStudents = 10
      - Open
      - Closed
        - do: Report course is closed
- Closed
  - [ numStudents = 3 ]
  - [ numStudents < 3 ]
- Canceled
  - cancelCourse
  - registration closed[ numStudents > 3 ]

---

```plaintext
calc_radar_defog_state
```

- calc_radar_defog_state
  - [
    - R_SPEED
    - L_SPEED
    - R_TIMER
    - L_TIMER
    - R_TIMER
    - L_TIMER
  ]

- [
  - [R_SPEED] > R_SPEED
  - [L_SPEED] > L_SPEED
  ```
- ```
  ```
```
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. Customer requests the rear defog system on.

2. The rear defog led is turned on.

3. The rear defog relay is enabled.

4. After being enabled for Low Speed Max On Time seconds, the rear defog relay is disabled.

5. The rear defog relay led is turned off.
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

### REAR_DEFQG_OUTPUT_FCT (Basic Activity)

<table>
<thead>
<tr>
<th>Name</th>
<th>Defined In</th>
<th>Type</th>
<th>Def</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAR_DEFQG_OUTPUT_FCT</td>
<td>REAR_DEFQG</td>
<td>Activity</td>
<td>Yes</td>
<td>Update</td>
</tr>
</tbody>
</table>

### General

**Termination Type:** Reactive-Controlled

**Selected Implementations:** None

**Mini-Spec:**

```plaintext
started/
DEFQG_DRIVE_OUT = 1; /* initialize to off */
chDEFQG_DRIVE_SGS/REAR_DEFQG_OUTPUT_FT1;  
```

**Is Activity:** Implemented by Module:

**Subroutine Binding...** Comb. Assignment... Truth Table... Edit... Delete...
Cross-Checking/Redundancy Checking
Discrete Event Simulation Too
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.)