Eliciting and Modelling Dependability Requirements

The Unified Model of Dependability
The Unified Model of Dependability

The Unified Model of Dependability (UMD) is a modeling framework for discussing and reasoning about dependability. By providing a structured language for eliciting and modeling dependability requirements, UMD helps stakeholders to clearly identify the measurable and implementable properties a system should possess in order to be dependable.

There are few things we need to focus on (dependability, requirements, stakeholders)

Let’s proceed step by step, to understand Why are we doing this, the Background, and finally, the Model
Presentation Outline

Why

• The High Dependability Computing Program (HDCP)

• Software Engineering and Requirements Engineering

• Requirements: functional and non-functional

• Dependability

Background

• The Unified Model of Dependability (UMD)

• Empirical evaluation of UMD: an experiment

The Model
HDCP: High Dependability Computing Project

• **Overall Goal**: Increase the ability of NASA to engineer highly dependable software systems via the development of new techniques, processes, and technologies.

• **Research Goal**: Develop high dependability technologies, assess their effectiveness under varying conditions, and transfer them into practice at NASA.

• **Participants**: CMU, UMD, MIT, USC, UW, Fraunhofer-Maryland

Carnegie Mellon
HDCP: two-step research project

To achieve the project goal, we need to understand

1. **What users want**
   When a system is dependable for the users (stakeholders)?
   How do we verify that a system is dependable?

2. **Which techniques apply to achieve high dependability**
   When is technique X more effective than technique Y?
   What is the cost of applying the technique X?
   How should you tailor technique X for your environment?

We focus on this problem
Background: Software Engineering

Term introduced in the late 1960s (software crisis era)

There are many definitions of software engineering. All of these share some common factors:

- Software engineering is concerned with software systems built by teams rather than individuals,
- uses engineering principle in the development of these systems, and
- includes both technical and non-technical aspects

Software engineering implies the concept of software process. Many software process models have been suggested…. 
Background: Software Processes

- Different Models have been suggested:
  - Waterfall model
  - Spiral
  - Rapid Application Development (RAD)
  - Agile
  - .......

There are many differences, but all share a common point:
All the development model start with the definition of the requirements of the system to be developed
(Also the agile methods that want to get rid of everything!!!)
Example: SEL Waterfall Model

- **Specification (SP) Activity**
  - Specification defects reports
  - High level design defects reports
  - High level design defects reports
  - Low level design defects reports
  - Code defects reports
  - Code defects reports

- **High Level Design (HLD) Activity**
  - Specification changes
  - Specification increments
  - Specification corrections reports
  - High level design changes
  - High level design increments
  - HLD corrections reports

- **Low Level Design (LLD) Activity**
  - High level design defects reports
  - Low level design defects reports
  - Low level design changes
  - Low level design increments
  - LLD corrections reports

- **Implementation (IMP) Activity**
  - Low level design defects reports
  - Code defects reports
  - Code defects reports
  - Code changes
  - Code increments
  - Code corrections reports

- **System Test (ST) Activity**
  - Implementation defects reports
  - System tested code
  - System tested code changes
  - System tested code increments
  - System tested code corrections reports

- **Acceptance Test (AT) Activity**
  - System test defects reports
  - Acceptance-tested code
  - Acceptance-tested code changes
  - Acceptance-tested code increments
Background: Requirements

• A requirement is something a product must do or a quality that a product must have to be useful for the stakeholders

Thus, we can distinguish

• Functional requirements are things the product must do

• Non-Functional requirements are qualities (or properties) that the product must have, i.e. how the functions should be performed

Let’s see more details on stakeholders, functional and non-functional requirements…
Who are the Stakeholders?

People who have interest in (or are affected by) the new system

Figure 1. An onion model of stakeholder relationships. Each circle represents a different stakeholder zone.
Functional Requirements: Example

- Let’s consider an electronic bookstore (amazon.com), what are some functional requirements?

  - login, book search,
  - shopping cart, wish list,
  - credit card payment,
  - personal preferences setting, on-line suggestions, etc.

  Ideas?
Non-Functional Requirements: Example

- Again, for an electronic bookstore (amazon.com), what are some non-functional requirements?

  Ideas?

- book search (performance: 10 sec response time; accuracy: ?)

- shopping cart, wish list (performance: 100 titles can be memorized)

- credit card payment (reliability: ?; security: no risk!!!),

- personal preferences setting (confidentiality: ?)
Difficulties with Non-Functional Requirements

• Different users have a different interpretation of the same quality attribute, for example:
  • Performance: response time 10 sec, or 5 sec, or 2 sec?
  • Accuracy: ??
  • Reliability: ??

• Depend on the context, for example:
  • Reliability varies widely between industrial sectors, ranging from safety-critical applications, where complete absence of failures is required (with higher costs, longer time to market and slower innovations), to everyday software (mobile phones, PDAs, etc.), where services must be cost effective and reasonably low failures rates can be tolerated.
Requirements Engineering

• Importance and difficulty of dealing with requirements has led to a new branch of Software Engineering… Requirements Engineering

• Named at the end of the ‘70, an area in continuous evolution… in terms of tools, but mainly at methodological and conceptual level

• Evolution is due to the continuous evolution of the systems to be developed ……

• But it is forced by the constant presence of classical and old problems: the requirements are often
  • wrong, unclear, old, missing…
  • mainly functional…rarely non-functional
Requirements Engineering: Basic Activities

- Elicitation / Discovery
- Negotiation
- Description (specification...)
- Management
- Validation

UML use cases, Scenarios, etc

Formal and semi-formal languages, tools..

Critical point!!!
Dependability: a complex non-functional requirement

- IFIP WG-10.4 defines dependability as the trustworthiness of a computing system that allows reliance to be justifiably placed on the services it delivers.

- “Reliance” is contextually subjective and reflects the particular users’ needs.
- So, in different circumstances:
  1. the focus will be on different attributes, e.g. availability, and performance;
  2. as well as different levels of adherence to such attributes.
Attributes of Dependability
(a selection from the literature)

- Failure-related: reliability, availability, survivability, robustness
- Protection-related: security, safety
- Quality of Service-related: accuracy, fidelity, performance, usability
- Life cycle-related: maintainability, scalability

Attributes mostly compatible and synergetic, but some conflicts and tradeoffs
  - Spreading information: survivability vs. security
  - Graceful degradation: survivability vs. quality of service
“Levels” of Dependability Attributes

- Reliability
- Availability
- Maintainability
- Survivability
- Robustness
- Security
- Safety
- Algebraic properties
- Accuracy
- Performance
- Usability
- Scalability
- 

Any property on which we depend on…
A key modelling mechanism for Dependability

How can we elicit and model dependability requirements?

Maybe we can use a concept common to all the dependability attributes, the concept of failure....

Example of failure:
For a specific service (e.g., the query service) of a specific system (e.g., an on-line application) a performance requirement could be “response time less or equal 10 seconds”

Thus, we say that there is a PERFORMANCE FAILURE when: Response time > 10 sec.
The same failure can be “viewed” in different ways

**Robustness**

**Survivability**

**Performance**

**Safety**

**Availability**

UNINTENTIONAL
(1) HW fault (1 CPU becomes unavailable)
(2) Too many requests

INTENTIONAL
Denial of Service attack

Is a HAZARD?
It depends…
e-Book Store → NO!
911 → MAYBE!

Is the service still useful?
It depends…
Internal bank system → YES!
e-Book Store → MAYBE!
(customer goes away)

FAILURE:
Response time greater than 10 sec.

Is due to an external EVENT?
The notion of failure as the key concept for modelling dependability

- Dependability
- Security
- Reliability
- Accuracy
- Availability
- Performance
- Survivability

Event
UNINTENTIONAL
(1) HW fault (1 CPU becomes unavailable)
(2) Too many requests

INTENTIONAL
Denial of Service attack

Issue
FAILURE:
Response time greater than 10 sec.
HAZARD?
Depends on the application

A bottom-up approach
The “Unified Model of Dependability”

Modeling framework (designed around the concept of failure) that helps stakeholders to express their dependability requirements by defining the issues (failures) with respect to the system or a service (scope), and the possible responsible external events.

- **event**
- **cause**
- **issue**
- **concern**
- **scope**

Denial of Service

Response time > 10 sec.

*On-line application example*

query service
Supporting the elicitation process

You can use any available models and classification to flesh out your specific dependability model

**event**

- **cause**
  - Type
  - Adverse Condition
  - Attack
  - Upgrades
  - etc.

**issue**

- **concern**
  - Type
  - Accuracy
  - Response Time
  - etc.
  - **Availability impact**
    - Stopping
    - Non-Stopping
  - **Severity**
    - Levels of severity
    - Utility function

**scope**

- **characterization:**
  - Type
  - Whole System
  - {Service}
  - Operational Profile Description
    - Distribution of transaction
    - Workload volumes
    - etc.

Denial of Service

Response time > 10 sec.

On-line application example

Query service
**event**

**characterization:**
- Type
  - Adverse Condition
  - Attack
  - Upgrades
- etc.

**cause**

**characterization:**
- Type
  - Adverse Condition
- Attack
- Upgrades
- etc.

**issue**

**characterization:**
- Type
  - Whole System
  - Service
  - Operational Profile
- etc.

**manifest**

**characterization:**
- Type
  - Whole System
  - Service
  - Operational Profile
- etc.

**reaction**

**characterization:**
- Impact mitigation
  - warnings
  - alternative services
  - mitigation services
- Recovery
  - recovery time
  - recovery actions
- Occurrence reduction
  - guard services

**scope**

**characterization:**
- Type
- Whole System
- Service
- Operational Profile
- Description
  - Distribution of transaction
  - Workload volumes
- etc.

**measure**

<table>
<thead>
<tr>
<th>Characterization</th>
<th>Probabilistic</th>
<th>Deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ordinal</td>
<td>rarely/sometime/never</td>
<td></td>
</tr>
<tr>
<td>Interval</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ratio</td>
<td>MTBF</td>
<td>% of cases MAX cases</td>
</tr>
</tbody>
</table>

**UMD Structure**

**Warning to user**

**Denial of Service**

**Response time > 10 sec.**

**On-line application example**

**5% of the queries**

**Query service**

**Failure**

**Characterization:**
- Type
- Accuracy
- Response Time
- etc.
- Availability impact
  - Stopping
  - Non-Stopping
- Severity
  - Levels of severity
  - Utility function

**Trigger**

**Occurrence reduction**

- Impact mitigation
  - warnings
  - alternative services
  - mitigation services
- Recovery
  - recovery time
  - recovery actions
- Occurrence reduction
  - guard services

**Occurrence reduction**

- Impact mitigation
  - warnings
  - alternative services
  - mitigation services
- Recovery
  - recovery time
  - recovery actions
- Occurrence reduction
  - guard services
Measuring Dependability

Stakeholders indicate what manifestations of an undesired issue (failure and/or hazard) are tolerable by identifying/selecting the most appropriate measurement model.

**measure**

<table>
<thead>
<tr>
<th>Characterization:</th>
<th>Probabilistic</th>
<th>Deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ordinal</td>
<td>- rarely/sometime/never</td>
<td></td>
</tr>
<tr>
<td>Interval</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ratio</td>
<td>- MTBF</td>
<td>- % of cases</td>
</tr>
<tr>
<td></td>
<td>- Prob. of occurrence</td>
<td>- MAX cases</td>
</tr>
</tbody>
</table>

- MTBF: Mean Time Between Failures
- Prob. of occurrence: Probability of occurrence
- % of cases: Percentage of cases
- MAX cases: Maximum cases
Engaging stakeholders to improve “Dependability”

**reaction**

**characterization:**
- **Impact mitigation**
  - warnings
  - alternative services
  - mitigation services
- **Recovery**
  - recovery time
  - recovery actions
- **Occurrence reduction**
  - guard services

Warn the user (administrator) about the delay

Suggest when the user should try again and provide priority next time

Service should recover in < 1 hour

Recovery should be automatic

Prevent saturation/thrashing by rejecting incoming request

**Scope:** Query service

**Failure:** Response time > 10 sec.

**Measure:** 5% of the queries

**Event:** Denial of Service attack

*On-line application example*
UMD Application Process

Build the **scope** table by listing the system services
For each stakeholder and each service in the scope table:

1. Identify the **issues** (failures) and **events** (when applicable) and specify/select the appropriate characterization schemes (failure: type, impact, severity)
2. Define or select the appropriate **measurement model**, and define the tolerable manifestations
3. Define or select **reaction** service (warnings, mitigations) alternative services, guards) and the recovery behavior

Reconcile multiple stakeholder needs
Case Study: Dependability Requirements for TSAFE

The Tactical Separation Assisted Flight Environment (TSAFE) is a software system designed to aid air traffic controllers in detecting and resolving short-term conflicts between aircraft.
Process/Tool:
Build the scope table by listing the services

Example data from TSAFE

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>system</td>
<td>TSAFE</td>
<td></td>
</tr>
<tr>
<td>Display aircraft position</td>
<td>Display position of the aircraft on the map</td>
<td></td>
</tr>
<tr>
<td>Display flight planned route</td>
<td>Display aircraft planned route if available</td>
<td></td>
</tr>
<tr>
<td>Display flight synthetized</td>
<td>Display aircraft projected route when non conformant</td>
<td></td>
</tr>
<tr>
<td>Highlight flight non conformance</td>
<td>Change color (to white) when aircraft non conformant</td>
<td></td>
</tr>
</tbody>
</table>
Process: A Web-based tool
Gathering data from a single stakeholder
### Process: data from a single stakeholder (TSAFE example)

<table>
<thead>
<tr>
<th>Scope</th>
<th>Event</th>
<th>Failure</th>
<th>Measure</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Memory Board Fault</td>
<td>“Peak throughput less than 100 flights/hour”</td>
<td>MTBF = 10E3 h MTTR = 30 min (Corresponds to Very High Availability)</td>
<td>(a) Warn about diminished performance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Type: Throughput Availability Impact: Non-stopping</td>
<td></td>
<td>(b) Suggest flights to leave out</td>
</tr>
<tr>
<td>Display flight synthesized route</td>
<td>N/A</td>
<td>“Accuracy: Position error &gt; Horizontal 0.25 NM Vertical 300ft”</td>
<td>MTBF = 2X10E6 h MTTR = 1h Max TTR = 2 h (Corresponds to Mission critical Availability)</td>
<td>Different dependability needs</td>
</tr>
<tr>
<td>Display flight synthesized route</td>
<td>N/A</td>
<td>“Computation takes more than 500 ms”</td>
<td>5% of cases</td>
<td>a) Warn about delay</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Type: Response time Availability Impact: Non-stopping</td>
<td></td>
<td>b) Stop displaying other non critical routes</td>
</tr>
</tbody>
</table>

Various measurement models can be used
Process/Tool - Graphical analysis

For example, it is possible to visualize the distribution of the identified issues across the different services.

In this way, we can quickly visualize where stakeholders’ concerns are focused, and also identify areas that have been neglected.

Example data from TSAFE
Process
Reconciling multiple stakeholders’ needs

Different stakeholders needs have to be reconciled. Two approaches:

1. **Merging.** E.g., when two or more stakeholders had filled tables concerning the same service, and identified different classes of failures (e.g., accuracy, response time, etc.) or different manifestations of the same class (e.g., different levels of response time) all the single results need to be considered.

2. **Negotiation.** E.g., when stakeholders ask the system to behave in incompatible ways (asking the system to react to the same failure by stopping or by providing an alternative service), or where their combined requests cannot be addressed with available resources (e.g., a crypto algorithm would reduce response time).
### Process: reconciling multiple stakeholders needs (TSAFE Example of merging)

**Stakeholder 1**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Event</th>
<th>Failure</th>
<th>Measure</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display flight synthesized route</td>
<td>N/A</td>
<td>“Computation takes more than 500 ms”</td>
<td>5% of cases</td>
<td>a) Warn about delay</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Type: <strong>Response time</strong></td>
<td></td>
<td>b) Stop displaying other non critical routes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Availability Impact: <strong>Non-stopping</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Stakeholder 2**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Event</th>
<th>Failure</th>
<th>Measure</th>
<th>Reaction</th>
</tr>
</thead>
</table>
| Display flight synthesized route | N/A   | “Computation takes more than 1 sec”               | MTBF = 2X10E6 h  
MTTR = 1h  
Max TTR = 2 h  
(Corresponds to Mission critical Availability) | a) Put system off-line and activate emergency procedures |
|                   |       | Type: **Response time**                          |             |                         |
|                   |       | Availability Impact: **Stopping**                |             |                         |

Two different levels of response time and tolerable manifestations have been recognized by the stakeholders, together with different reactions.
Process/Tool - Data Analysis: aggregating values of dependability

It is possible to combine measures expressing the tolerable manifestation for each of the identified issues. For example:

1. The MTBF of a specific type of failures, e.g., accuracy;
2. The MTBF of a specific type of failures that are also stopping, e.g., accuracy failures that are also stopping failures;
3. The resulting desired availability, for the whole system and/or each service (by having the MTBF and the MTTR of all stopping failures affecting the system and/or a service).
### Aggregating values of dependability: Example resulting availability for a service (1)

#### Stakeholder 1

<table>
<thead>
<tr>
<th>Scope</th>
<th>Event</th>
<th>Failure</th>
<th>Measure</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display flight synthesized</td>
<td>N/A</td>
<td>“Accuracy: Position error &gt; Horizontal 0.25 NM Vertical 300ft”</td>
<td>MTBF = 2X10E6 h MTTR = 1h</td>
<td>(Corresponds to Mission critical Availability)</td>
</tr>
<tr>
<td>Route</td>
<td></td>
<td>Type: Accuracy</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Availability Impact: Stopping</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Stakeholder 2

<table>
<thead>
<tr>
<th>Display</th>
<th>Event</th>
<th>Failure</th>
<th>Measure</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>flight synthesized</td>
<td>N/A</td>
<td>“Computation takes more than 1 sec”</td>
<td>MTBF = 2X10E6 h MTTR = 1h</td>
<td>a) Put system off-line and activate emergency procedures</td>
</tr>
<tr>
<td>route</td>
<td></td>
<td>Type: Response time</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Availability Impact: Stopping</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These values can be combined to compute the availability for the service (see next slide)
Aggregating values of dependability: Example resulting availability for a service (2)

### Stakeholder 1

<table>
<thead>
<tr>
<th>Scope</th>
<th>Event</th>
<th>Failure</th>
<th>Measure</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display flight synthesized route</td>
<td>N/A</td>
<td>Accuracy: Position error &gt; Horizontal 0.25 NM Vertical 300ft*</td>
<td>MTBF = 2x10^6 h MTTR = 1 h Max TTR = 2 h (Corresponds to Mission critical Availability)</td>
<td></td>
</tr>
</tbody>
</table>

### Stakeholder 2

<table>
<thead>
<tr>
<th>Scope</th>
<th>Event</th>
<th>Failure</th>
<th>Measure</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display flight synthesized route</td>
<td>N/A</td>
<td>Accuracy: Position error &gt; Horizontal 0.25 NM Vertical 300ft*</td>
<td>MTBF = 2x10^6 h MTTR = 1 h Max TTR = 2 h (Corresponds to Mission critical Availability)</td>
<td></td>
</tr>
</tbody>
</table>

### Stakeholder i

<table>
<thead>
<tr>
<th>Scope</th>
<th>Event</th>
<th>Failure</th>
<th>Measure</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display flight synthesized route</td>
<td>N/A</td>
<td>Accuracy: Position error &gt; Horizontal 0.25 NM Vertical 300ft*</td>
<td>MTBF = 2x10^6 h MTTR = 1 h Max TTR = 2 h (Corresponds to Mission critical Availability)</td>
<td></td>
</tr>
</tbody>
</table>

### Stakeholder N

<table>
<thead>
<tr>
<th>Scope</th>
<th>Event</th>
<th>Failure</th>
<th>Measure</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display flight synthesized route</td>
<td>N/A</td>
<td>Accuracy: Position error &gt; Horizontal 0.25 NM Vertical 300ft*</td>
<td>MTBF = 2x10^6 h MTTR = 1 h Max TTR = 2 h (Corresponds to Mission critical Availability)</td>
<td></td>
</tr>
</tbody>
</table>

Service Availability = \( \frac{MTBF(S)}{MTBF(S) + MTTR(S)} \) x 100

where,

\[ MTBF(S) = \frac{1}{\sum_{i=1}^{N} \left( \frac{1}{MTBF_i} \right)} \]

\[ MTTR(S) = \text{Max } MTTR_i \]
Current Applications of UMD

UMD applications cover different systems and different areas:

1. UMD has been currently applied to build dependability models (precisely defining dependability requirements) of SW system: FC-MD TSAFE and CMU testbeds

2. Collaboration work is progress with NASA-JPL in using UMD to define a generic dependability model for a RTSJ (precisely defining metrics to collect suitable to represent high level dependability needs) to support empirical comparison among different physical implementations (see next slide)

3. Work has started to build a dependability model of a SW-intensive system (SCRover) in collaboration with USC
Empirical assessment of UMD

Next Tuesday!!!