Engineering Fine-Grained Dependability Requirements

An Environment Modeling based Approach

Zhi Jin

Key Lab of High Confidence Software Technologies (MoE)
Peking University
zhijin@pku.edu.cn

QRS, 2017.07.28, Prague, Czech Republic
Outline

• **Motivation:**
  – System need to be more dependable

• **Challenges:**
  – Dependability is non-functional feature and needs to be interweaved with functional features

• **Approach:**
  – Derive dependability concerns from environment features
  – Adopt control-based framework to interweave dependability and functionality

• **Expectation:**
  – Benefits and further efforts
Motivation: Trend in Computing

- **Cyber-Physical Systems**
  - Cyber-physical systems are integrations of computation with physical processes. Embedded computers and networks monitor and control the physical processes, ...... (Edward A. Lee)
  - The integration of physical systems and processes with networked computing has led to the emergence of a new generation of engineered systems: cyber-physical systems. (CPS steering group)
  - A world where physical objects are seamlessly integrated into the information network, and where the physical objects can become active participants in business processes. (SAP)
Motivation: Trend in Computing

Cyber-Physical (-Social) Systems
Software systems are to be tightly integrated with the physical systems and the social systems with networked sensing, computation, actuation, etc.
Populations of *computing entities* will be a significant part of our environment, *performing tasks* that support us, and we shall *be largely unaware of them*.

The most profound technologies are those that *disappear*. They weave themselves into the fabric of everyday life until they are *indistinguishable* from it.

*Mark Weiser, a pioneer of ubiquitous computing*

Invisible = Software and hardware are embedded in the physical world and human society. That produce a new operable "application scenario"
Motivation: Trend in Computing

**Traditional Application Scenario**

Software: be in charge of information processing

As tool of the information processing, software needs only to meet the predefined specification.

**New Application Scenario**

Software: be carrier of application values

As a carrier of application values, software needs to deal with the open and dynamic environment, continuously meet the diverse and varied needs of users.

**Adaptivity, Continuous Evolution, Dependability and Scalability**

*Turing Award winner: Joseph Sifakis (2011): A Vision for Computer Science, CACM.*
Outline

• **Motivation:**
  – System need to be more dependable

• **Challenges:**
  – Dependability is non-functional feature and needs to be interweaved with functional features

• **Approach:**
  – Derive dependability concerns from environment features
  – Adopt control-based framework to interweave dependability and functionality

• **Expectation:**
  – Benefits and further efforts
State of Art: Process

- A defined software process is essential
  - Enforcing standards, avoiding that issues fall through cracks, learning from past mistakes
  - Including procedures for version control, bug tracking and regression testing
  - Including standard structures for documents and guidelines for meetings
  - Including collection of detailed statistics and explicit mechanisms for adjusting the process accordingly
State of Art: Testing

• To find bugs
  – Structural tests identify bugs in known categories
    • A mutation test, a regression test, ……
  – A successful test is one that fails, and thus identifies a bug

• To provide evidence of dependability
  – Test cases are drawn randomly from the expected profile of use, and statistical inferences are made about the likelihood of failure
  – A successful test is one that succeeds to provide direct evidence for demonstrating dependability
Challenges Remained

• What form of process and testing should take that can offer just enough dependability considering the cost, usability, performance, etc.?
  – make balance

• The adoption of rigorous processes and testing has an indirect impact on dependability, evidence of a direct link between dependability and design is missing.
  – build trace links

• Developers find interweaving the business needs and dependability needs is still real headache
  – help operationalization

D. Jackson (2009), A Direct Path to Dependable Software, Communication of ACM, 52(4): 78-88
Challenges to Developers

• From where, the needs for dependability can be identified?
• What is the relationship between dependability needs and the business functionality?
• How will these two be interweaved together?
Outline

- **Motivation:**
  - System need to be more dependable

- **Challenges:**
  - Dependability is non-functional feature and needs to be interwoven with functional features

- **Approach:**
  - Derive dependability concerns from environment features
  - Adopt control-based framework to interweave dependability and functionality

- **Expectation:**
  - Benefits and further efforts
Dependability by Construction

• **Dependability by construction**
  – build dependability into every step
  – demand
    • rigorous requirements definition
    • precise system-behavior specification
    • solid and verifiable design
    • code that can be precisely understood

• **Construction starts from requirements definition**
Where comes Requirements

• **Needed business capabilities**
  – To solve the business domain problem

• **Needed quality properties of entire system, a system component, service, or function**
  – Not about business logics
  – But ensure the quality of domain problem solving
What is Dependability

The notion of **dependability**, defined as the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers, enables these various concerns to be subsumed within a single conceptual framework.

IFIP WG10.4
Quality Properties

Domain Feature Usage Scenario
- Integrity
- Reliability
- Availability
- Robust
- Security
- Safety

Algorithm Implementation
- Efficiency
- Flexibility
- Usability
- Interoperability
- ......

System Architecture
- Maintainability
- Portability
- Reusability
- Testability
- ......

Developer’s Design skill

Business Constraints

Usage Experience

Business Relevance

Dependability

It is a kind of unspecified functional requirements
The task of RE

*Given* Environment Assumptions
*And* Requirements
*When* Conduct RE
*Then* System Capability is decided
Three Penetrations

Analyzing Environment Features

Directly focusing on Requirements

Considering System Potential Faults

Environment Assumptions + Specification → Requirements
NFR Framework: Focus on Req

- Generic to any NFR
- Directly Analyze Requirements
- Associate to Function Implementation

UMD: Concerning System Failures

- Start from the potential issues of the system
- Identify the event that may cause the issues and the scope impacted by the issues
- Decide the measurements for detecting the issues
- Specifying the desired system reactions

Dependability from Environment

Analyzing Environment Features and Application Scenarios

Directly focusing on Requirements

Considering System Potential Faults

Environment Assumptions + Specification → Requirements
Why this is Reasonable?

When the Environment is Open, Dynamic, Uncertain, Safe-Critical, Malicious?
### Inherent Properties for the Physical/Social World

- Need to be adaptive to match and pace with

### Software System

- Context-aware Reqs.
- Self-adaptation Reqs.
- Real Time Reqs.
- Availability Reqs.
- Security Reqs.
- Functional Reqs.
- Changeable
- Safety-Critical Factors
- Non-Deterministic Factors
- Malicious Factors
- System Fault
- Malicious Factors
- Errors
- Real Time Reqs.

### Social World

- Undesired external interference
  - Need to guard against
- Undesired, loss-caused effects
  - Need to be prevented

### Why this is reasonable?

- Inherent Properties for the physical/social world
  - Need to be adaptive to match and pace with

- Undesired internal behaviors
  - Need to avoid by online self-healing
Why not Dependable?

- **Domain Assumptions**: Errors and Malicious Attacks from known or unknown environment entities
- **Specification**: Changes in causal entity, e.g. un-recognized states, new causal entities
- **Side Effects**: Un-proper system behaviors causing disaster to critical environment entity
- **Requirements**: Fault in system producing undesired system performance

Flow:
- **Domain Assumptions** → **Un-anticipated Domain Behaviors**
- **Specification** → **New Domain Assumptions**
- **Side Effects** → **Fatal System Behaviors**
- **Requirements** → **Un-anticipated System Behaviors**
Dependability from Scenario

Define just-enough quality property
Trace to application context
Be operationalized as interactions or constraints

Based on environment assumptions and application context to introduce dependability strategies

Directly focusing on Requirements
Considering System Potential Faults

Environment Assumptions + Specification → Requirements
Control Based Meta-Model

- Control Environment Abnormal Behaviors
  - Control System Abnormal Behaviors
    - Dependable System
      - Feedback controller
      - Feedforward controller
      - Core system
        - System behaviors
    - Desired Environment behaviors
    - Undesired Environment behaviors
Requirements Representation

A Knowledge Base about Threats and Faults

System

Feedback controller

Behaviors deviations

Desired behaviors

Controls

Feedforward controller

Core system

Environment Entities

Threats (Attacks, Malicious Usage, Operation Errors, ……)

Use-Cases

FB Control-Cases

FF Control-Cases

Environment Entities

Entities

Entities
**Conceptual Model**

Environment Model

- Data
- Designed Entity
- Operator
- Physical World

Share phenomena

System Model

- Threat
- Function Profile
- System Behavior Deviation
- Feedforward Control Profile
- Feedback Control Profile

Threaten

Detect

Control

Counter

Produce

Control

Detect

Counter

Share phenomena
## Concerns Identification from Interactive Environment

### Upper Level Ontology of Environment Model

<table>
<thead>
<tr>
<th>Environment Entity / System Asset / Interaction / Phenomenon</th>
<th>Undesired Feature</th>
<th>Implied Concern</th>
</tr>
</thead>
<tbody>
<tr>
<td>External / Internal Autonomous Entity</td>
<td>Has malicious intent to access</td>
<td>Authorization Concern</td>
</tr>
<tr>
<td>System / System Component</td>
<td>Produce unexpected behavior / output; Failure</td>
<td>Fault tolerance Adaptation Concern</td>
</tr>
<tr>
<td>External Entity</td>
<td>Trigger known attack / virus</td>
<td>Security Concern</td>
</tr>
<tr>
<td>External Symbolic Entity</td>
<td>Has different levels of sensitiveness</td>
<td>Privacy Concern</td>
</tr>
<tr>
<td>External Physical Device</td>
<td>Produce unexpected input</td>
<td>Robustness Concern</td>
</tr>
<tr>
<td>External Entity</td>
<td>Valuable or Critical</td>
<td>Safety Concern</td>
</tr>
<tr>
<td>Connection</td>
<td>Be lost, Be tampered</td>
<td>Security Concern</td>
</tr>
<tr>
<td>Interactive Environment</td>
<td>Uncertain</td>
<td>Adaptation Concern</td>
</tr>
<tr>
<td>Featured Entity / Service / Interaction</td>
<td>Threat</td>
<td>Countermeasure</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Private/sensitive data</td>
<td>Information disclosure in transmission or service delivering</td>
<td>Strong authorization to data accessing; Strong encryption to the data; Communication link securing with protocols that provides message confidentiality</td>
</tr>
<tr>
<td>High available system service</td>
<td>Denial of service by malicious user</td>
<td>Resource and bandwidth throttling; Input validation and filtering</td>
</tr>
<tr>
<td>Malicious operator</td>
<td>Spoofing for illegal usage</td>
<td>Strong authentication; Strong encryption to operators’ login data; Authentication cookie protection with Secure Sockets Layer</td>
</tr>
<tr>
<td>Critical / Valuable data, Device, or Interactor that can result in big loss</td>
<td>Tampering with data in transmission or data storage and/or processing</td>
<td>Data hashing and signing; Digital signatures; Strong authorization; Tamper-resistant protocols across communication links; Communication link securing with protocols that provides message integrity</td>
</tr>
<tr>
<td></td>
<td>System fault or behavior deviation</td>
<td>Oracle-based system behavior checking</td>
</tr>
<tr>
<td>Open system/service with highly-desired availability</td>
<td>Virus, e.g. Trojan horse, Worms, ......</td>
<td>Block all unnecessary ports at the firewall and host; Disable unused functionality; Harden weak, default configuration settings</td>
</tr>
</tbody>
</table>
Process of Requirements Elicitation

- **Adopt use cases** to specify the business functional requirements
- For each use case
  - Identify **feed-forward controllers** to handle the potential undesired inputs, e.g. errors, attacks, etc. They are the external threats
  - Identify **feed-back controllers** to handle the potential system behavior deviations. They are the internal threats
  - Adopt threat-counter patterns (specific domain knowledge) to specifying the operationalization of the controllers
  - **Weave controllers and use case** to build dependable use case
Requirements Elicitation Process

Start

Functional requirements elicitation

Concern: what services does the system need to deliver?

Use cases model

Participants: users, requirements engineers

Concern: how does the system ensure the dependability of the delivered services?

Dependability requirements elicitation

Use/control cases model

Participants: users, requirements engineers, domain experts

End
Requirements Elicitation Process

- Identify undesired interactions (potential attacks, errors, ......, and system behavior deviation)
- Assess the risks
- Determine the control policy

Patterns for controller
- Measure: Detect what?
- Compare: Compare with what?
- Compute: Based on what to make decisions?
- Correct: Control what?

Knowledge Base about Threats and Controls
Requirements Elicitation Process

- Each interaction described in a use case
  - Identify the interaction's properties of interest
  - Identify the deviations of each property
  - Determine the possible threats and behavior deviations causing the property deviations

Example:
- Interaction of use case: log-in:
  - The customer inputs the account and password.
- Properties of interest:
  - 1. The **frequency** of this interaction,
  - 2. The **confidentiality** of the account and password
- Deviations of properties:
  - 1. This interaction occurs frequently (**guideword:** more)
  - 2. The account and password is disclosed (**guideword:** no)
- Threats or system behavior deviations:
  - 1. Brute force attack
  - 2. Network monitoring

The threats and system behavior deviations derived from interaction of the use case
Use Case / Controller Model

Online store system

- Log in
  - Threat (T)
  - <<threaten>>
  - <<control>>
  - <<counter>>

- Search goods
  - <<produce>>
  - <<control>>

- Response delay
  - <<trigger>>
  - <<produce>>

- Unauthorized access
  - Threat
  - <<counter>>

- Response time monitor and control

Customer

Behavior deviation

Feedforward control case

Feedback control case
**Multi-level Controls**

- Customer
  - Log in
    - Authentication
      - Unauthorized access
      - Multi-factor authentication
        - Brute force attack
      - Response time monitor and control
      - Response delay
  - Search goods
    - <<threaten>>
    - <<control>>
    - <<counter>>
  - <<produce>>
  - <<trigger>>

- <<threaten>>
- <<counter>>
- <<produce>>
- <<trigger>>
<table>
<thead>
<tr>
<th>FFControl case:</th>
<th>Authentic controlled use case:</th>
<th>Stakeholders:</th>
<th>Threat model: Threat: Unauthorized access to the system</th>
<th>Threat description: The unauthorized customer may buy a lot of goods with malicious intentions in the name of others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controlled use case:</td>
<td>Log in</td>
<td>Customer</td>
<td>Unauthorized access</td>
<td></td>
</tr>
<tr>
<td>Stakeholders:</td>
<td>Customer</td>
<td></td>
<td>The unauthorized user may buy a lot of goods with malicious intentions in the name of others</td>
<td></td>
</tr>
<tr>
<td>Controls:</td>
<td>while the customer log in, the system needs to require the customer to provide the password, and validate it. If it is valid, allow to enter, otherwise, deny the login</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FBControl case:</th>
<th>Response time monitor and control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controlled use case:</td>
<td>Search goods</td>
</tr>
<tr>
<td>Stakeholders:</td>
<td>Customer, system manager</td>
</tr>
<tr>
<td>Behavior deviation model:</td>
<td>Response time &gt;30 sec</td>
</tr>
<tr>
<td>Behavior deviation description:</td>
<td>While the customers search what they are interested in, they expect the system to respond within 30 sec. But with the increase of the customers, the response time may delay, and it will affect the reputation of the enterprise</td>
</tr>
<tr>
<td>Controls:</td>
<td>the system needs to monitor the response time of each request. And if the response time delays, activate more computing resource to accelerate the system responses</td>
</tr>
</tbody>
</table>
Requirements Interweaving


Integrated with ICONIX by Including Control Cases
## Extended Modeling Icons

<table>
<thead>
<tr>
<th>Functional Object</th>
<th>Icon</th>
<th>Dependability Object</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary Object</td>
<td><img src="image1" alt="Boundary Object Icon" /></td>
<td>Dependability Boundary Object</td>
<td><img src="image2" alt="Dependability Boundary Object Icon" /></td>
</tr>
<tr>
<td>Entity Object</td>
<td><img src="image3" alt="Entity Object Icon" /></td>
<td>Dependability Entity Object</td>
<td><img src="image4" alt="Dependability Entity Object Icon" /></td>
</tr>
<tr>
<td>Controller</td>
<td><img src="image5" alt="Controller Icon" /></td>
<td>Dependability Controller</td>
<td><img src="image6" alt="Dependability Controller Icon" /></td>
</tr>
</tbody>
</table>
Static / Dynamic Controller Robustness Analysis

Process: use case/control case driven robustness analysis

Input:
- \( US \): the set of use cases
- \( CS \): the set of control cases

Output:
- Extended robustness diagrams

Begin:
1. for each use case \( uc \) \( \in US \)
2. identify the boundary object, entity object, and control object from \( uc \)’s description;
3. construct the robustness diagram for \( uc \);
4. find the set of control cases \( CS_{uc} \) that directly or indirectly control the use case \( uc \) from \( CS \);
5. for each control case \( cc \) \( \in CS_{uc} \)
6. identify boundary object, entity object, and control object from \( cc \)’s description;
7. if the controls described by \( cc \) are dynamic
   8. construct an independent robustness diagram for \( cc \);
   9. else
      10. if \( cc \) directly controls \( uc \)
         11. add the objects identified from \( cc \) to the robustness diagram of \( uc \);
      12. else
         13. add the objects identified from \( cc \) to the diagram of the control case that \( cc \) controls;
14. endif
15. endif
16. endfor
17. End
Extended Interaction Analysis

Process: "use case+control case" driven interaction modeling

Input:
- US: the set of use cases
- CS: the set of control cases
- RS: the set of extended robustness diagram

Output:
- Extended sequence diagrams

Begin:
1. for each extended robustness diagram ers \in RS
2. copy the involved use case and control cases of ers to the left margin of the sequence diagram;
3. add the involved actors in ers to the sequence diagram;
4. add the involved entity objects in ers to the sequence diagram;
5. add the involved boundary objects in ers to the sequence diagram;
6. for each control object co in ers
7. allocate the behaviors of co among the collabor objects;
8. endfor
9. endfor

End:

The symbols for use cases:

- \textbf{: Object}
- \textbf{Message for use case}

The symbols for control cases:

- \textbf{: Object}
- \textbf{Message for control case}
The Whole Process

Use Case Description

Use Case Diagram

Controllers and Use Case Diagram

Interactive Entities and Attributes

NLP Techniques:
Domain Entity Recognition, Relation Classification

Risk Analysis
Countermeasure Selection
Threats-Countermeasure Knowledge Base

Domain Knowledge
Candidate Controlled Domain, Threats and Concerns

Environment Entity / System Asset | Undesired Feature | Implied Concern
--- | --- | ---
System / System Component | Produce unexpected behavior / output; Failure | Fault tolerance Adaptation Concern
External Entity | Trigger known attack / virus | Security Concern
External Symbolic Entity | Has different levels of sensitiveness | Privacy Concern
External Physical Device | Produce unexpected input | Robustness Concern
External Entity | Valuable or Critical | Safety Concern

Guided Generation

Candidate Controlled Domain, Threats and Concerns

Domain Knowledge
Domain Expert Experience
(Entity's Property, Relation's Property)
A Case Study: Online Stock Trading System

- Create an account
- Submit and execute an order
- Un-allocate an allocation from an account
- Allocate trade to an account
- Un-allocate an account
- Update an order
- Suspend an order
- Subscribe business information
- Submit and execute a market order
- Submit and execute an order with float limit
- Submit and execute an order with hard limit
- Submit and execute an order with stop price
- Update order price according to tickers
- Execute an order
- Route an order to exchanger
- Report order executed information
- Calculate the allocation
- Update the constraint to broker
- Update broker exposure
- Subscribe client information
- Unallocate an allocation from an account
- Allocate trade to an account
- Un-allocate an account
- Update an order
- Suspend an order
- Subscribe business information
- Submit and execute a market order
- Submit and execute an order with float limit
- Submit and execute an order with hard limit
- Submit and execute an order with stop price
- Update order price according to tickers
- Execute an order
- Route an order to exchanger
- Report order executed information
- Calculate the allocation
- Update the constraint to broker
- Update broker exposure
- Subscribe client information
- Unallocate an allocation from an account
- Allocate trade to an account
- Un-allocate an account
- Update an order
- Suspend an order
Online Stock Trading System: Use Cases

Use case: Log in
Actor: Trader
Preconditions:
  ● The stock trading system is available.
Main flow:
  1. The trader clicks the login button on the Home page.
  2. The system displays the Login page.
  3. The trader enters the account name and password, and click the submit button.
  4. The system validates the account information against the persistent account data and returns the customer to the Home page.
Postconditions:
  ● The trader has logged in the system.
Alternative flows:
  4a. The account information is not right:
  4a1. The system displays a message to inform the failure and prompts the [trader to either re-enter the account information or click the create account button]

Use Case: Submit an order
Actor: Trader
Preconditions:
  ● The exchanger which the order will route is connected and can accept instructions from system.
  ● The trader has logged in.
Main flow:
  1. The trader clicks the submit order button on the Home page
  2. The system displays the order submission page.
  3. The trader sets the basic information of the order: the stock symbol, the size, the type of the order in remote flag field, the price, and the type of the transaction (buy or sell).
  4. The trader clicks the submit button to send the order to system.
  5. The system checks the order if legal.
  6. The system routes the order to the exchange where the stock lists for trading
  7. The system sends a submission success message to the trader
Postconditions:
  ● The system has received an order from the trader.
  ● The system waits for the trading result of the order.
Alternative Flows:
  5a. The order is not legal.
  5a1. The system asks the trader to reset the information of the order.
  7a. The order’s submission fails.
  7a1. The system returns the failure information to the trader.
Online Stock Trading System: Use Case with Controllers

Threats Identification and Controller Design

- Decrease order processing time
  - <<control>>
  - <<counter>>
  - <<produce>>
- Submit an order
  - <<threaten>>
- Encrypt order
  - <<counter>>
  - <<counter>>
  - <<threaten>>
  - <<counter>>
- Choose alternative
  - <<counter>>
  - <<threaten>>
  - <<counter>>
- Decrease order price according to ticker
  - <<counter>>
  - <<threaten>>
  - <<counter>>
- Route an order to exchanger
  - <<counter>>
  - <<counter>>
  - <<counter>>
- Multifactor authentication
  - <<counter>>
  - <<counter>>
  - <<counter>>
  - <<threaten>>
- Log in
  - <<threaten>>
  - <<counter>>
  - <<counter>>
  - <<counter>>
- Access authentication
  - <<counter>>
  - <<counter>>
- Encrypt account information
  - <<counter>>
  - <<counter>>
  - <<counter>>
  - <<counter>>
- Encrypt account information
  - <<threaten>>
  - <<counter>>
- Load monitor and balance
  - <<counter>>
  - <<counter>>
  - <<counter>>
- Exchanger monitor
  - <<counter>>
  - <<counter>>
- Failure of link to ticker feed
  - <<counter>>
  - <<counter>>
- Failure of link to exchanger
  - <<counter>>
  - <<counter>>
- Current connection failure
  - <<counter>>
  - <<counter>>
- Delay of trader to submit order
  - <<counter>>
  - <<counter>>
- Interception of the information
  - <<threaten>>
  - <<counter>>
  - <<counter>>
- Increase of waiting time for routing an order
  - <<counter>>
  - <<counter>>
  - <<counter>>
  - <<counter>>
- Unauthorized access
  - <<counter>>
  - <<counter>>
  - <<counter>>
  - <<counter>>
- Delay of the trader to submit order
  - <<counter>>
  - <<counter>>
  - <<counter>>
  - <<counter>>
- Unauthorized access
  - <<counter>>
  - <<counter>>
  - <<counter>>
  - <<counter>>
- Interception of the information
  - <<counter>>
  - <<counter>>
  - <<counter>>
  - <<counter>>

Online Stock Trading:
Threats Identification and Controller Design
Online Stock Trading System: Controller Description

**FFcontrol case: Encrypt account name and password**
Stakeholder: Trader
Controlled use case: Log in
Threat model:
  - Threat name: Data interception
  - Threat description: After the customer enters the account information, the account information may be intercepted by some malicious persons through some sniffers. The malicious person may use the account information for some purpose undesired by the customer.
Controls:
  - Alternative 1: Encryption
    - Actions: After the customer enters the account information for login, the system encrypts the account name and password before other actions.

**FFcontrol case: limit the number of password attempts**
Stakeholder: Trader
Controlled use case: Log in
Threat model:
  - Threat name: Password cracking
  - Threat Description: Once some malicious persons know the account name of the trader, he will crack the account password by testing the password again and again with the help of some software tools.
  - Characteristics quantity: The number of the password attempts in one trading day
  - Acceptable interval: [0,5]
  - Event: The number of the password attempts in one trading day>5
Controls:
  - Alternative 1: Limit the number of password attempts in one trading day
    - Actions: 1. while the trader enters the account information, the system first check the number that the trader has attempted, and them the system validate the account.
    2. If the password is right, then return the trader to the Home page.
    3. If the password is not right, the system needs to increase the number of the password attempts.
    4. IF the number of attempts is bigger than three, the system displays the message about the closure of the account on the Login page.

**FFcontrol case: Encrypt order**
Stakeholder: Trader
Controlled use case: Submit an order
Threat model:
  - Threat name: Data interception
  - Threat description: Someone may use some agents to intercept the order information that the trader has submitted. In that way, the malicious person may fake the information to destroy the system or cause losses to the trader.
Controls:
  - Alternative 1: Encryption
    - Actions: The system encrypts the order after the trader has submitted it.

**FFcontrol case: Enable alternate connection**
Stakeholder: Trader
Controlled use case: Submit an order
Threat model:
  - Threat name: Exchange connection failure
  - Threat description: Because of the physical reasons, the connection between the system and each exchange may be not available. This will cause that the order can’t be routed to the exchange timely, and bring some losses.
  - Characteristics quantity: The state of the connection to the exchange
  - Acceptable interval: The connection is ok.
  - Event: The connection is not available.
Controls:
  - Alternative 1: Monitor the state of the connection and alarm the failure
    - Actions: 1. The system sends the "SYSTEM CHECK" message to the exchange in every 5mintues.
    2. If the connections are ok, the system will receive the same message from the exchange.
    3. If one connection is down, the system needs to alarm, and enable the alternate connection.

**FFcontrol case: Decrease order process time**
Stakeholder: Trader
Controlled use case: Submit an order
Online Stock Trading System: Log-in Static Controller and Robustness Analysis

1. Use cases + control cases diagram

- Encrypt account information
- Data interception
- Limit the number of password attempts

2. Analyze use case: log in

- Enter account and click submit
- Login page
- Validate account
- Display
- Account

3. Analyze control case: limit the number of password attempts

- Customer
- Click login
- Home page
- Display
- Validate account
- Increase the number of password attempts

4. Analyze control case: encrypt account

- Customer
- Click login
- Login page
- Encrypt account
- Account
Online Stock Trading System: Log-in Controlled Use Case

Use case: Login/control cases: encrypt account name and password, limit the number of password attempts

Main flow:
1. The trader clicks the login button on the Home page.
2. The system displays the Login page.
3. The trader enters the account information, and click the submit button.

_encrypt account name and password_: After the customer enters the account information for login, the system encrypts the account information before other actions.

4. The system validates the account information against the persistent account data and returns the trader to the Home page.

_limit the number of password attempts_: While the trader enters the account information, the system first checks the number that the trader has attempted, and then validate the account.

2. If the password is right, then return the trader to the Home page.
3. If the password is not right, the system needs to increase the number of the password attempts.
4. If the number of attempts is bigger than three, the system displays the denying message on the Login page.

Alternative flows:
4a. The account information is not right:
4a1. The system displays a message to inform the failure and prompts the trader to either re-enter the account information or click the create account button.
Online Stock Trading System: Submit Order
Robustness Analysis Diagram with Static Controller

1. Click submit order button
2. Home page
3. Display
4. Order submission page
5. Encrypt order
6. Check order
7. Route order
8. Exchange connector
9. Exchange
10. Submit order

Steps:
- Set order information and click submit
- Order
Online Stock Trading System: Submit Order
Dynamic Controllers

Reduce Processing Time

Activate Alter Connection

Activate alarm
Connection failure alarm driver
Connection failure alarm

Set alarm “on”

Enable alternate connection

“SYSTEM CHECK” message

Exchange connector

Exchange

Count time
Monitor exchange connection

Monitor average order process time
Activate alarm

Set alarm “on”

Order submission page

Increase servers
Server manager
Server pool

Exchange connector

Activate
Alter Connection
Online Stock Trading System: Submit Order
Dynamic Controllers: Reduce Processing Time

Control cases: decrease order process time

Controls:

Alternative 1: Increase computing resource
Precondition: There are idle servers which can be allocated

Actions: 1. After the trader submits the order, the system needs to start to monitor the process time of the order.
2. After the system has routed the order, the system ends the monitor.
3. The system computes the average time of processing the order.
4. The system allocates more servers to deal with the orders accepted from the traders if the average time > 0.8.

Alternative 2: Submission delay alarm
Precondition: There are no allocatable servers.

Actions: 1. The system activates the submission delay alarm to report that the submission delay occurs and the allocatable resources are exhausted.

Including Controller and Control actions are interweaved with business actions
Online Stock Trading System: Submit Order
Dynamic Controllers: Activate Alter Connection

Control cases: enable alternate connection

Controls:
The system sends the "SYSTEM CHECK" message to the exchange in every 5 minutes.

If the connections are ok, the system will receive the same message from the exchange.

If one connection is down, the system needs to alarm, and enable the alternate connection.

Including Controller and Control actions are interwoven with business actions.
Key Points

• Start from function scenarios
  • Modeling functional/business requirements
    (Dependability is accordance with business logic and domain value)
  • Focusing on interactions between the system and its interactive environment (input threats and output effect take place here)
  • Each dependability requirement is attached onto a functional point (just enough scope, and dependability trace links)
• Knowledge based
• The strategies dealing with the dependability issues are IT techniques based (reuse mature experience)
1. Model system as a control system. Within a certain context, for handling the critical factors in the interactive environment $D$, and the unexpected system behaviors, use controllers to guarantee the satisfiability of $R$.

2. Use feed-forward controllers to control the environment factors; use feed-back controllers to avoid disasters resulted by system behavior deviations.

3. Provide guidelines to help identifying controlling policies based on knowledge about strategies of enhancing system dependability.

4. Integrate with ICONIX framework to provide fine grained operationalization of dependability requirements that are integrated into functional requirements to reduce the burden of developers.
Outline

• **Motivation:**
  – System need to be more dependable

• **Challenges:**
  – Dependability is non-functional feature and needs to be interweaved with functional features

• **Approach:**
  – Derive dependability concerns from environment features
  – Adopt control-based framework to interweave dependability and functionality

• **Expectation:**
  – Benefits and further efforts
Benefits

• Providing guided process to support
  – the elicitation of dependability requirements
  – trace link building among dependability of different layers
  – interweaving of the business functionality and dependability functionality of fine-grained
Future Work

• More case studies, real industry applications
• Quantify risks, threats and countermeasures so to prioritize dependability needs and other NFRs
• To become a go-through approach from specification to execution depends on:
  – Dynamic re-configuration and deployment
  – Run-time system adaptation and evolution
  – .......
• All are challenges
Acknowledgements

• National Grand Fundamental Research Program of China under Grant No. 2009CB320701, Ministry of Science and Technology

• Key Project of National Natural Science Foundation of China under Grant No. 90818026

• Thanks to the students and colleagues who contribute to these projects

• Zhi Jin, Environment Modeling based Requirements Engineering for Software-Intensive Systems (to be published by Elsevier)
Thanks For Your Attention