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

高可信软件技术教育部重点实验室

QRS, 2017.07.28, Prague, Czech Republic

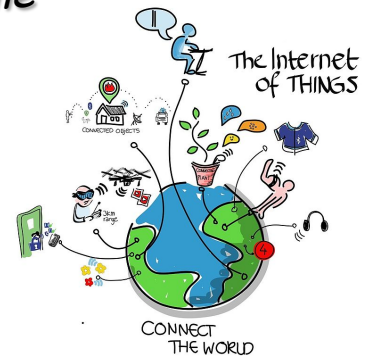
北京大学

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.

Realm of Integration: Everywhere + Invisible

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

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

Traditional Application Scenario

Software: be in charge of

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

New Appli

Software: be carrier of application values

ication values, e needs to deal with the open and dynamic environment, continuously meet the diverse and varied needs of users

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 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

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

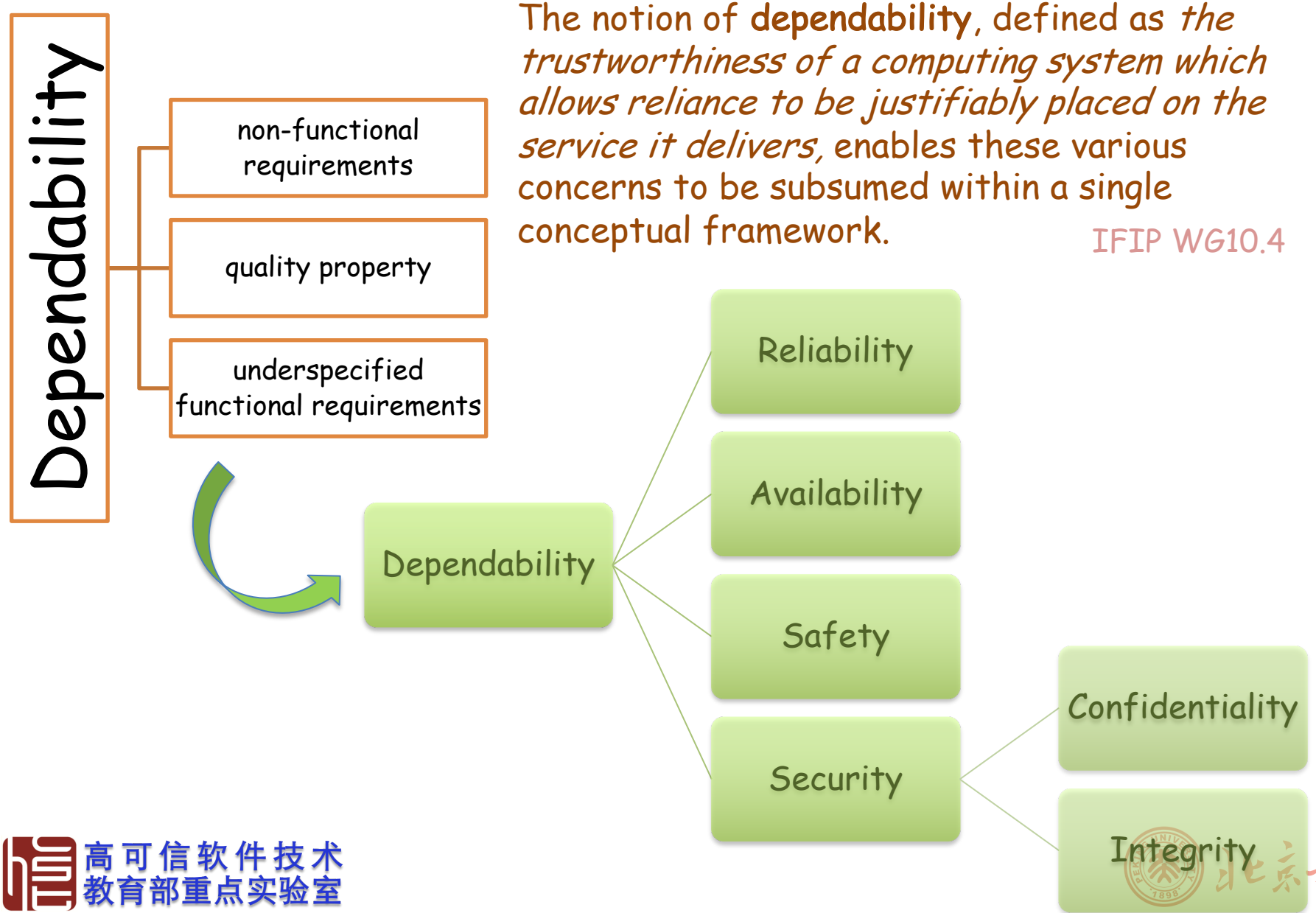
Where comes Requirements

- 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

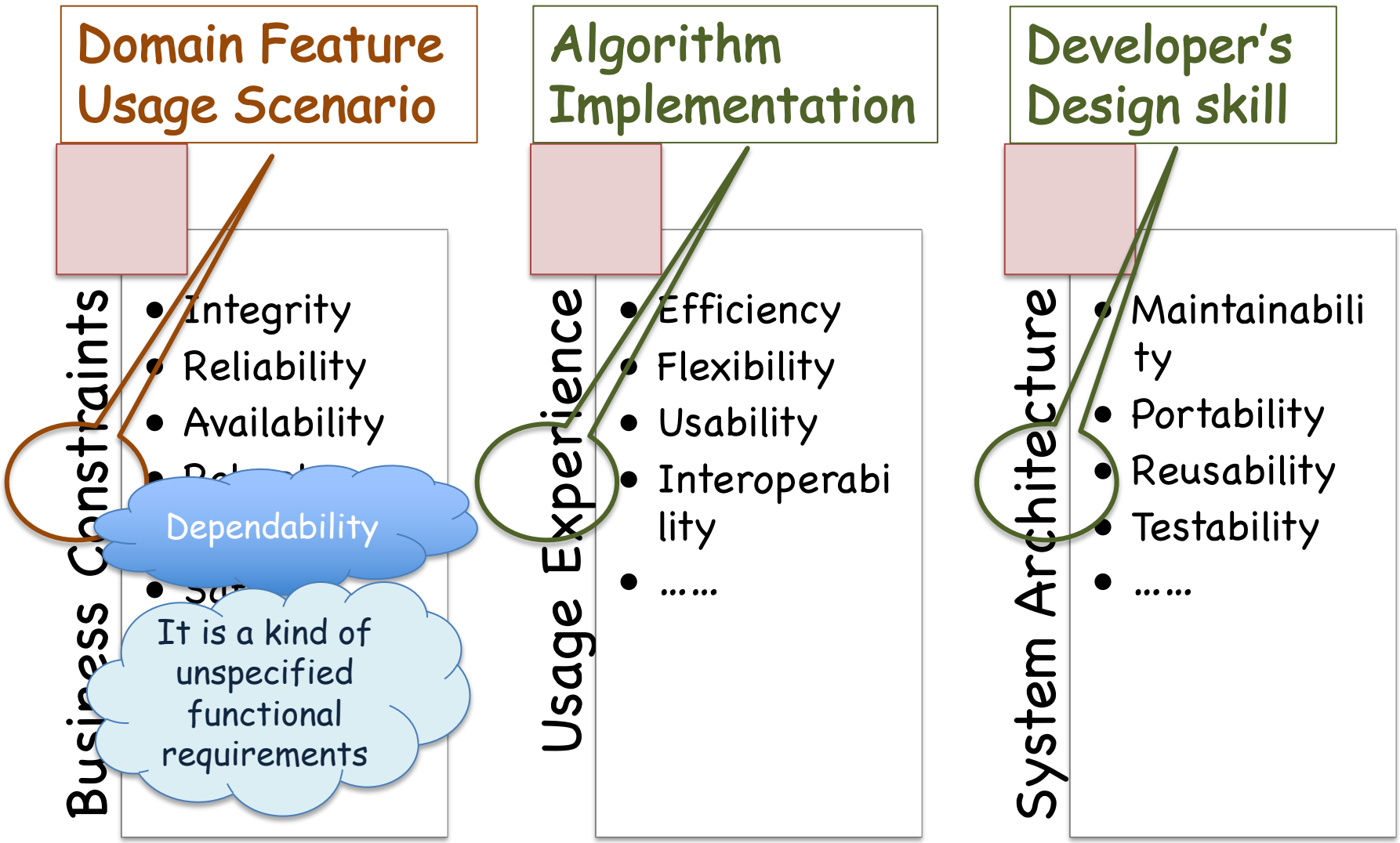
What is Dependability

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



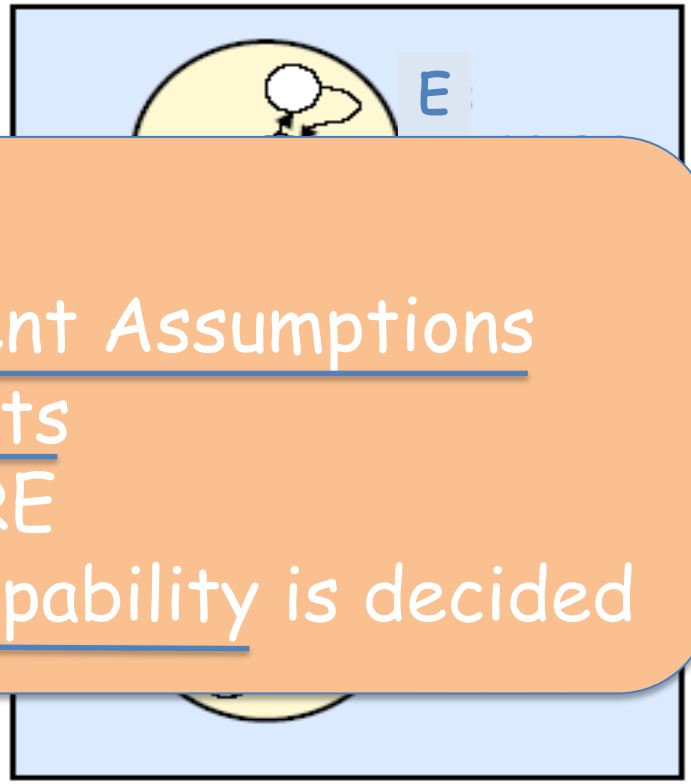
Quality Properties



Business Relevance

RE assumption and Dependability Argument

Environment Assumptions + Specification ⊢ Requirements



The task of RE

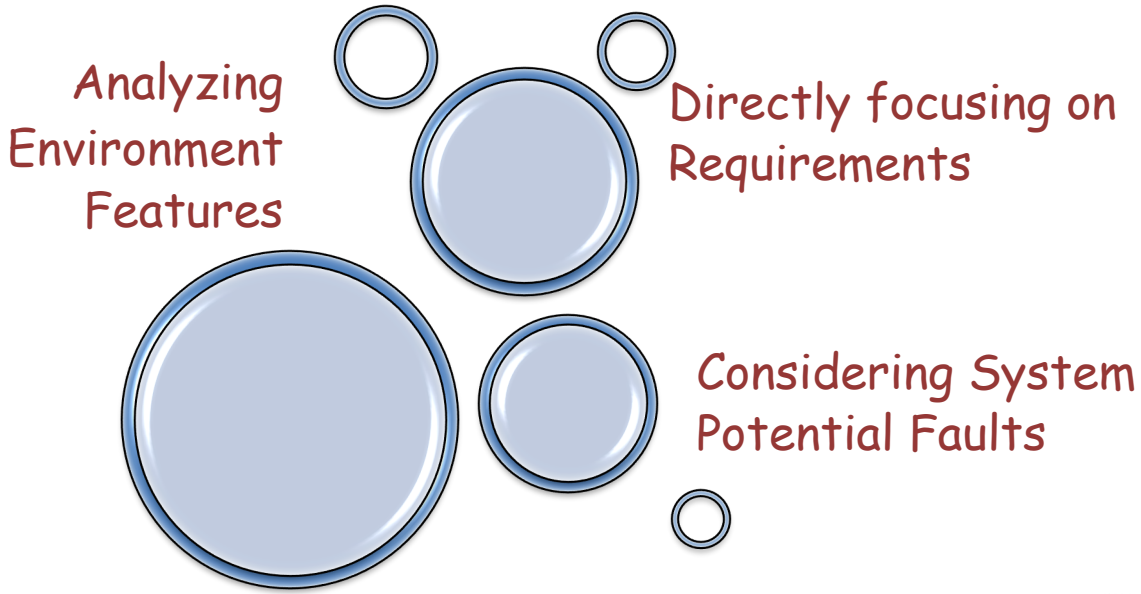
Given Environment Assumptions

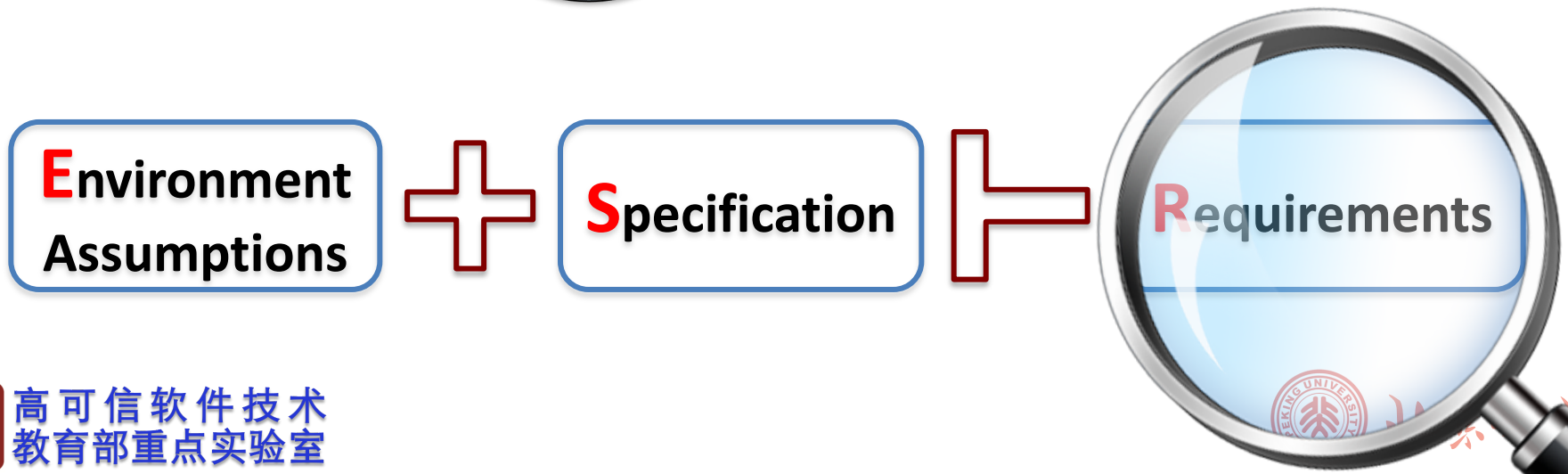
And Requirements

When Conduct RE

Then System Capability is decided

Three Penetrations





NFR Framework: Focus on Req

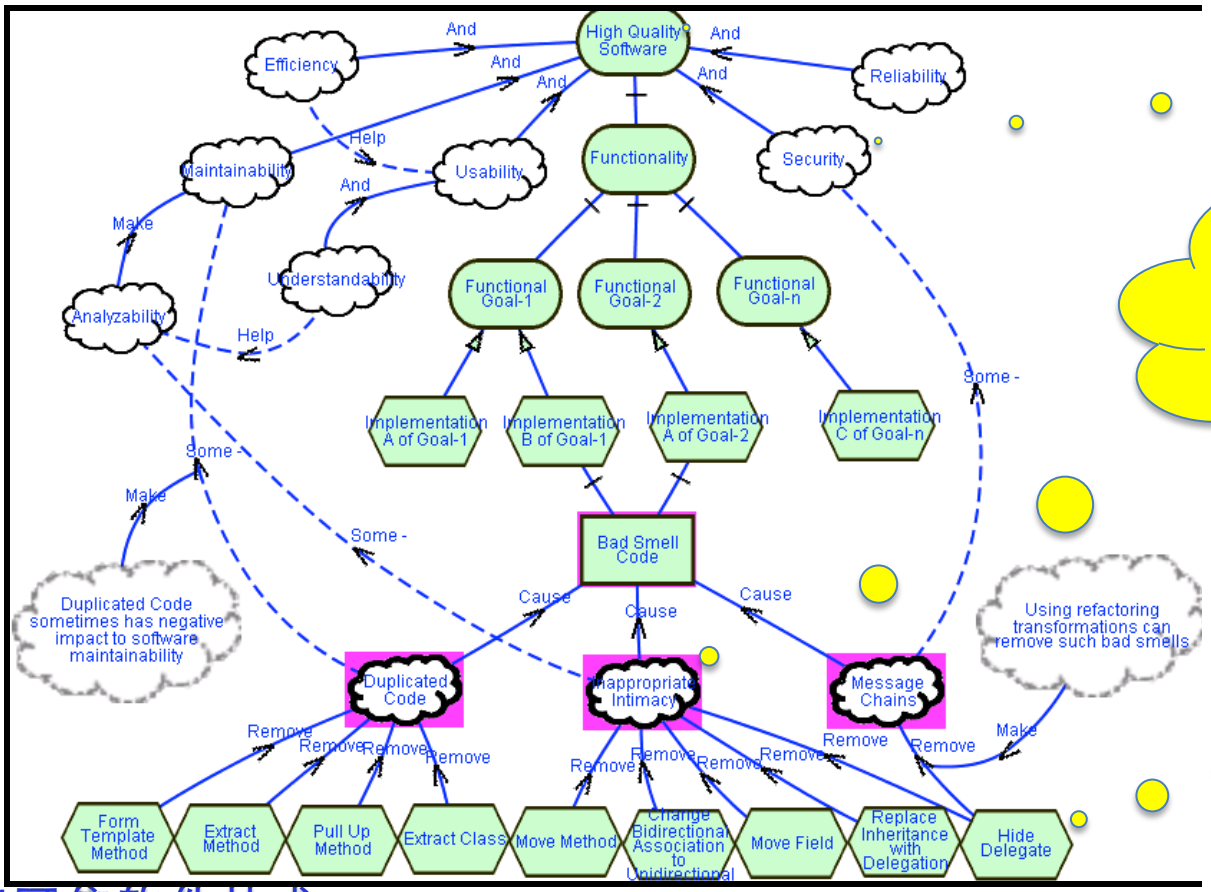
- Generic to any NFR
- Directly Analyze Requirements
- Associate to Function Implementation

I want high quality

Which types of quality ?

Which vulnerabilities may be introduced during implementation that may reduce quality ?

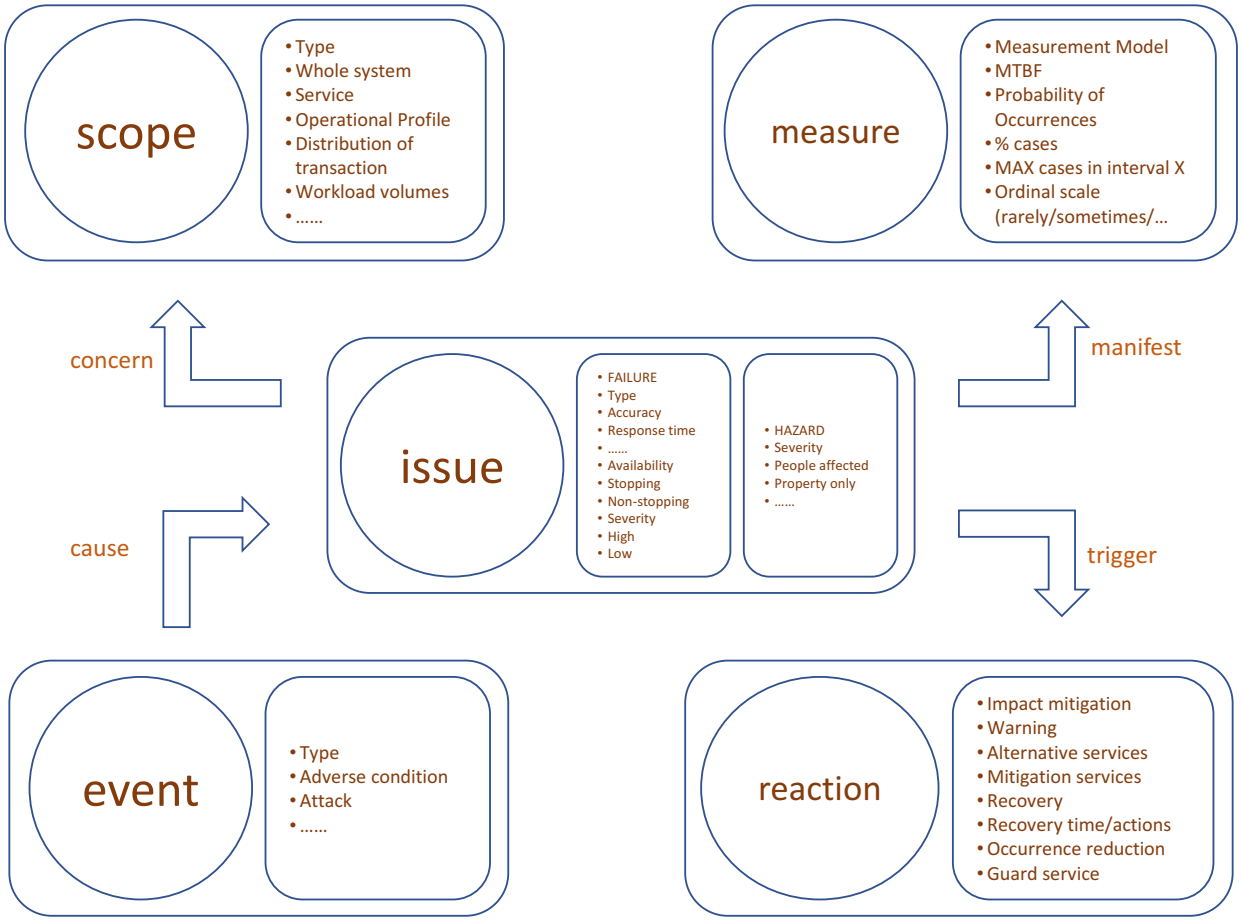
What are countermeasures dealing with the vulnerabilities ?



Chung, L., Nixon, B., Yu, E., Mylopoulos, J., 2000, Non Functional Requirements in Software Engineering, Kluwer Academic Publisher.

UMD: Concerning System Failures

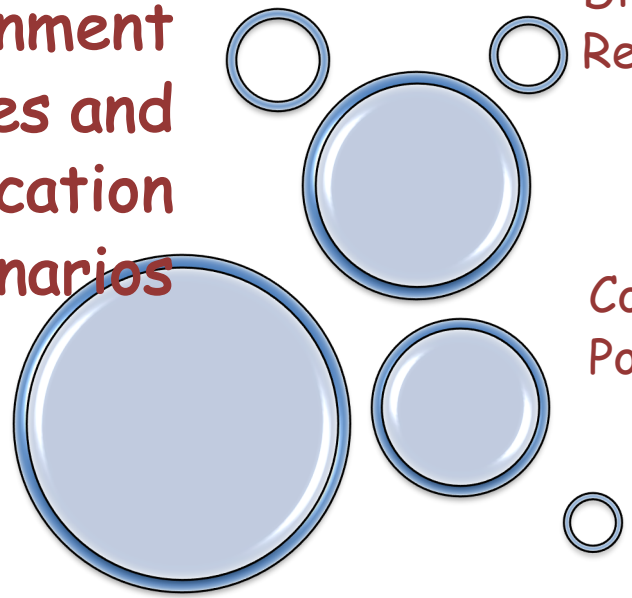
- Start from the potential issues of the system
- Identify the event that may cause the issues and the scope impacted by the issues
- Decide the measurements for detecting the issues
- Specifying the desired system reactions



Basili, L., Clements, P., Asgari, S., 2004. The Unified Model of Dependability: Putting Dependability in Context, IEEE Software 21(3): 19-25

Dependability from Environment

Analyzing Environment Features and Application Scenarios



Directly focusing on Requirements

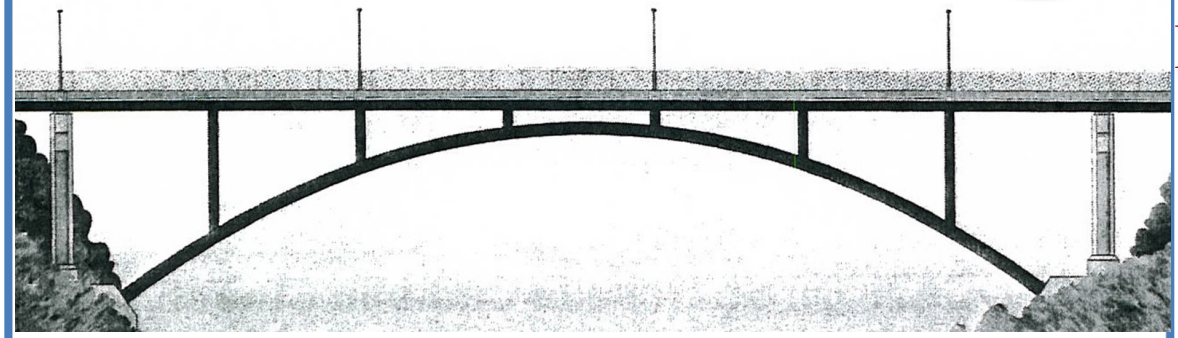
Considering System Potential Faults

Environment Assumptions + **S**pecification ⊢ **R**equirements

Why this is Reasonable ?

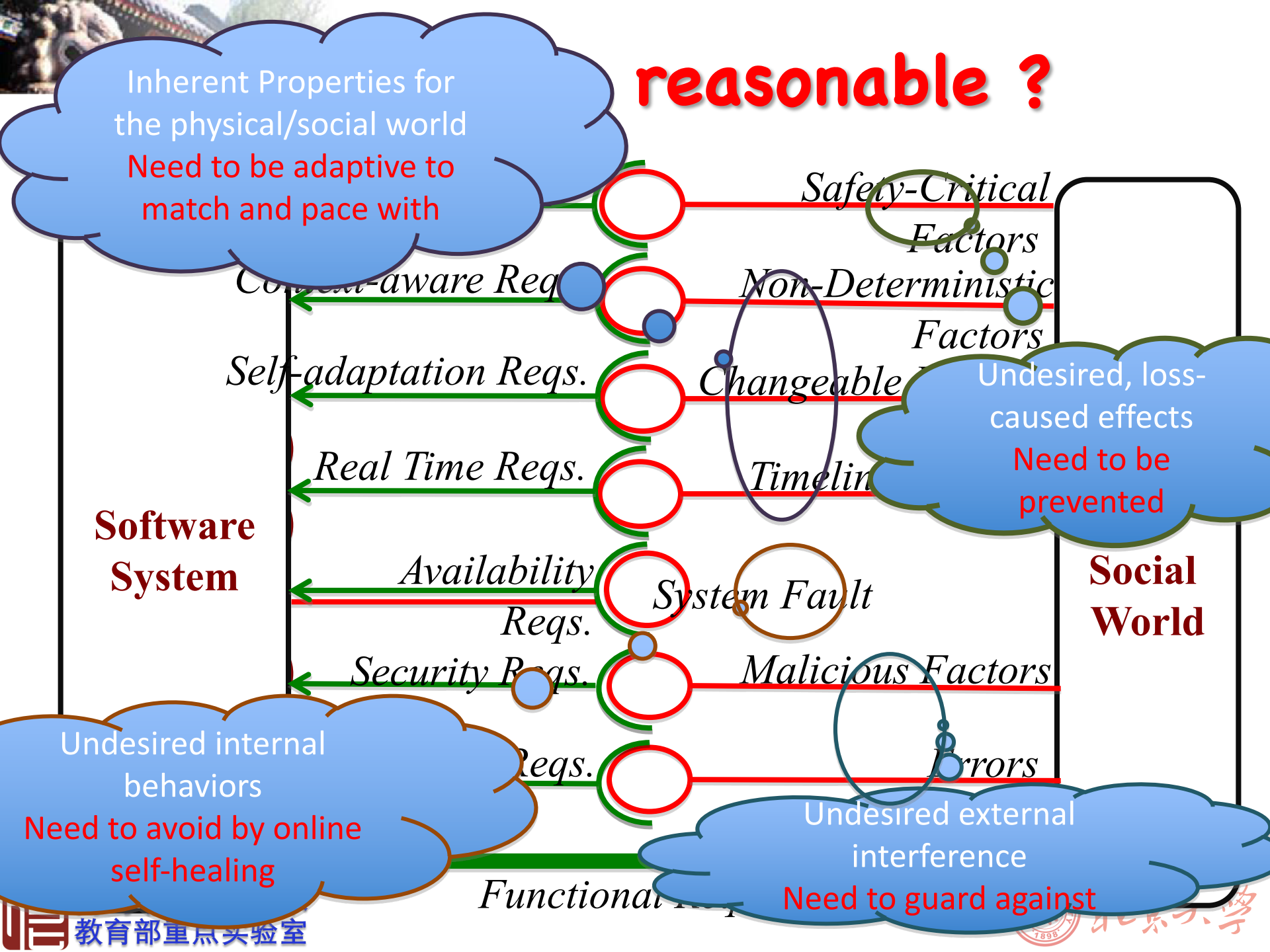
When the Environment is Open, Dynamic, Uncertain, Safe-Critical, Malicious ?

Software System



Physical and Social World

Functionality


reasonable ?
Inherent Properties for the physical/social world
Need to be adaptive to match and pace with
Safety-Critical Factors
Non-Deterministic Factors
Changeable
Timelin
Undesired, loss-caused effects
Need to be prevented
Software System
Social World
Availability Reqs.
System Fault
Malicious Factors
Errors
Undesired internal behaviors
Need to avoid by online self-healing
Undesired external interference
Need to guard against

Why not Dependable ?

Errors and Malicious Attacks from known or unknown environment entities	→	Un-anticipated Domain Behaviors
Changes in causal entity, e.g. un-recognized states, new causal entities	→	New Domain Assumptions
Un-proper system behaviors causing disaster to critical environment entity	→	Fatal System Behaviors
Fault in system producing undesired system performance	→	Un-anticipated System Behaviors

Dependability from Scenario

Define just-enough quality property

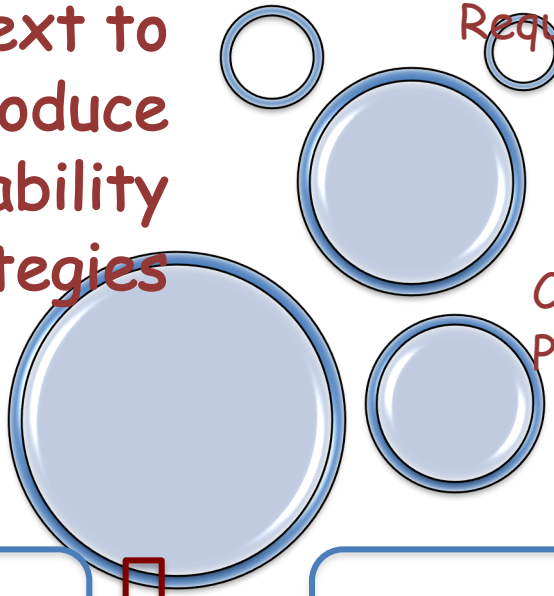
Trace to application context

Be operationalized as interactions or constraints

Based on environment assumptions and application context to introduce dependability strategies

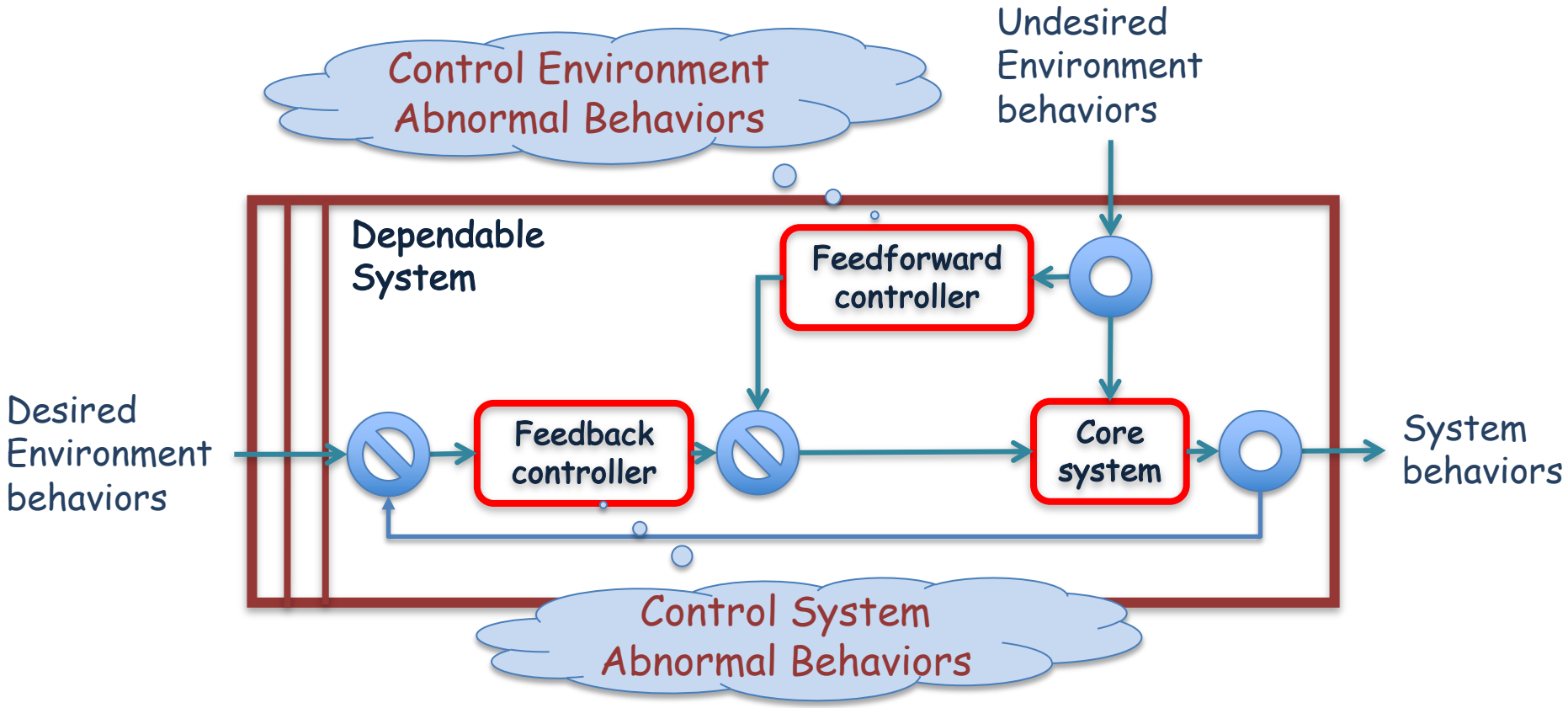
Directly focusing o
Requirements

Considering Sys
Potential Faults

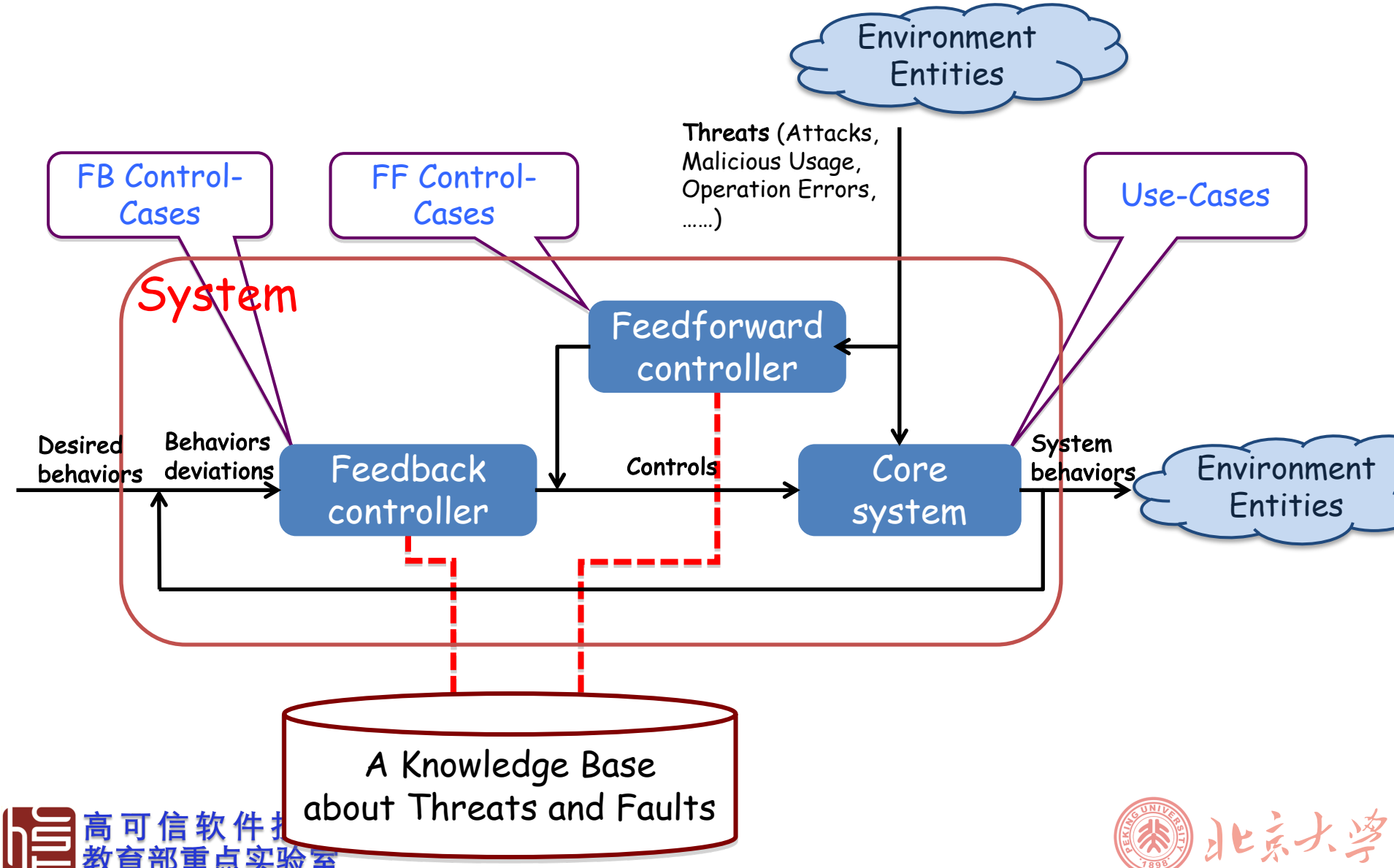


Environment Assumptions + **S**pecification ⊢ **R**equirements

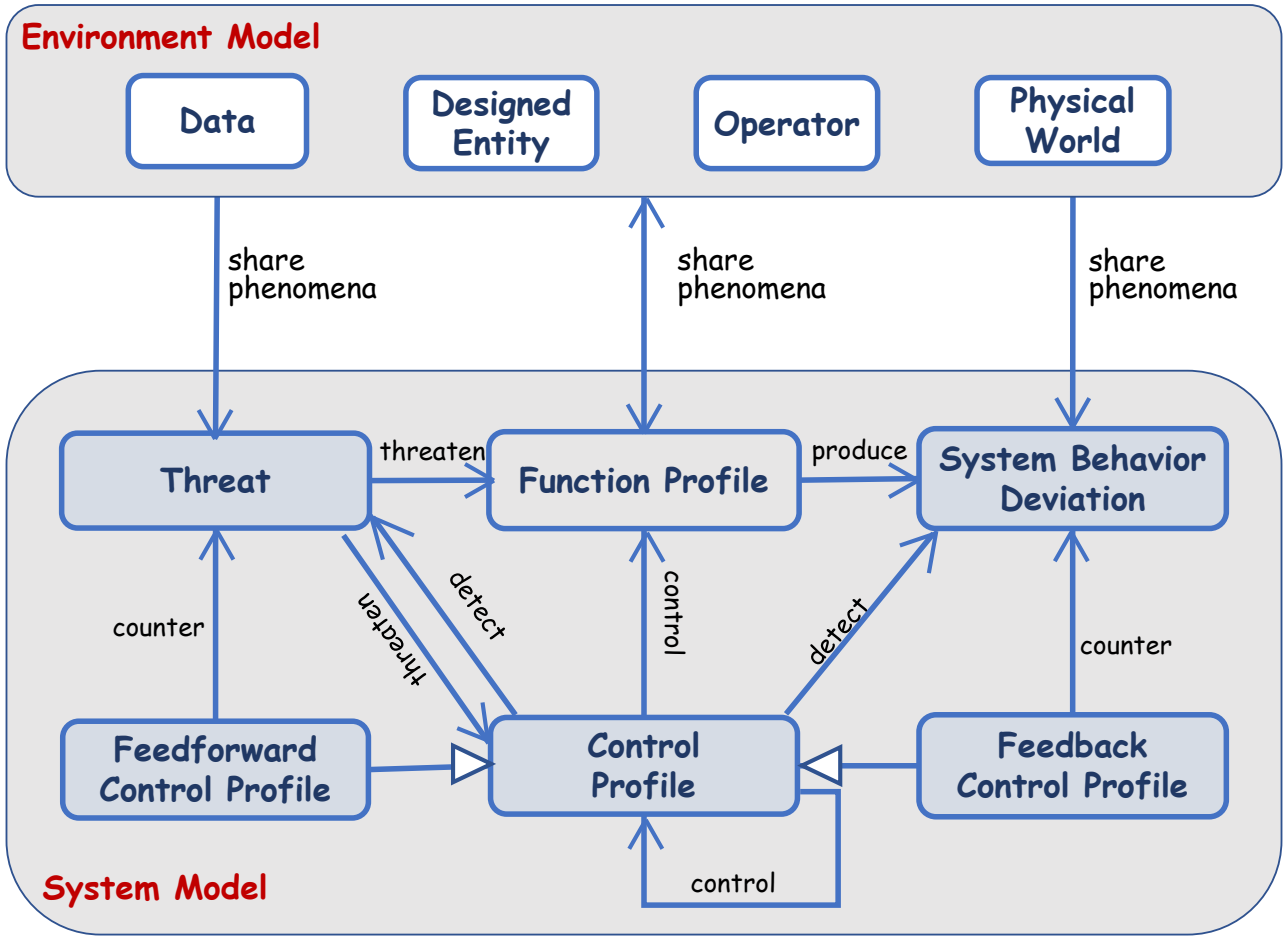
Control Based Meta-Model



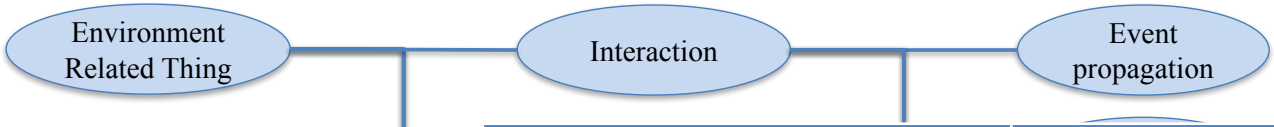
Requirements Representation



Conceptual Model



Concerns Identification from Interactive Environment



Upper Level Ontology of Environment Model

Environment Entity / System Asset / Interaction / Phenomenon	Undesired Feature	Implied Concern
External / Internal Autonomous Entity	Has malicious intent to access	Authorization 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
Connection	Be lost, Be tampered	Security Concern
Interactive Environment	Uncertain	Adaptation Concern

Entity / Threat / Countermeasure

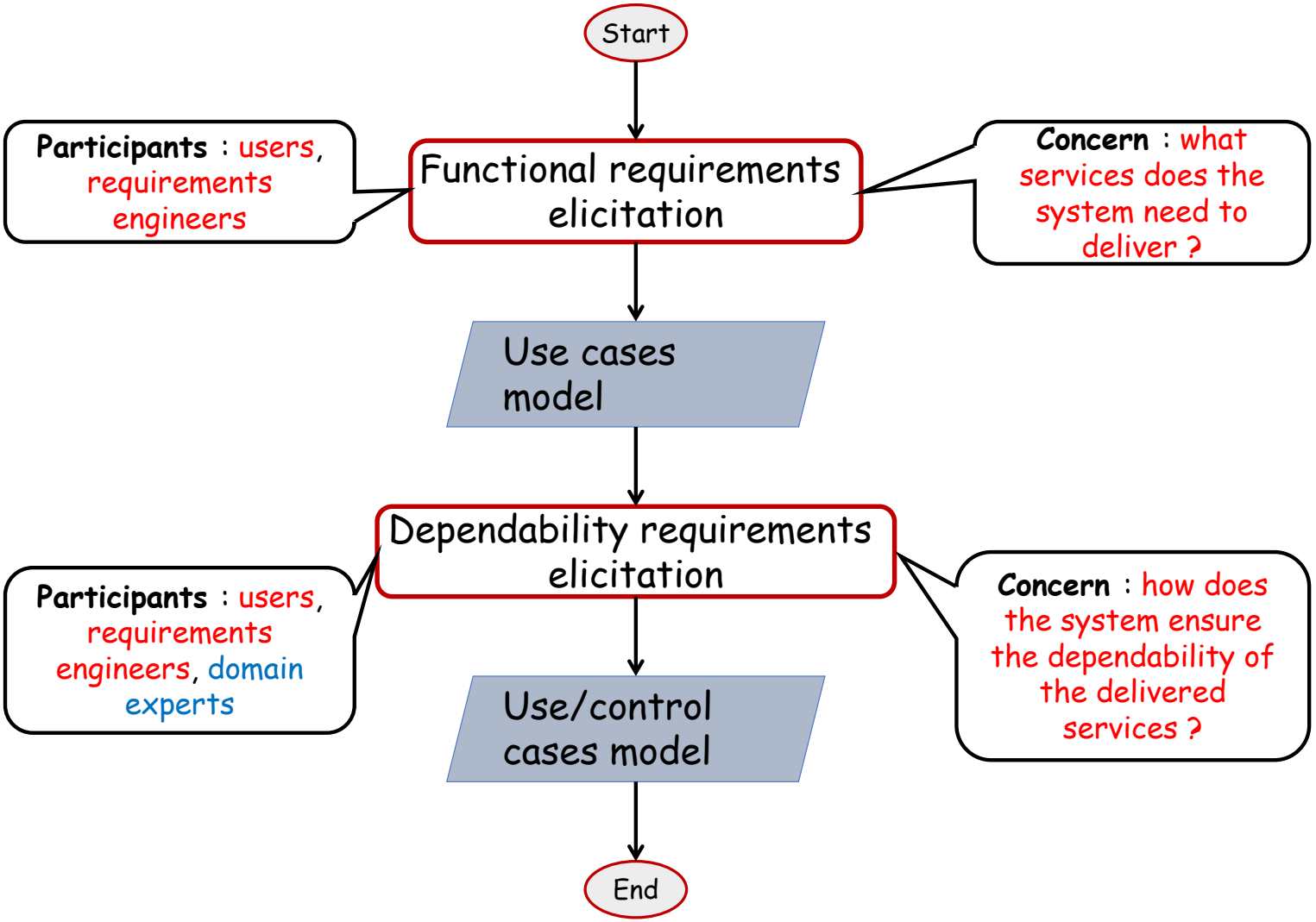
Featured Entity / Service / Interaction	Threat	Countermeasure
Private/sensitive data	Information disclosure in transmission or service delivering	Strong authorization to data accessing; Strong encryption to the data; Communication link securing with protocols that provides message confidentiality
High available system service	Denial of service by malicious user	Resource and bandwidth throttling; Input validation and filtering
Malicious operator	Spoofing for illegal usage	Strong authentication; Strong encryption to operators' login data; Authentication cookie protection with Secure Sockets Layer
Critical / Valuable data, Device, or Interactor that can result in big loss	Tampering with data in transmission or data storage and/or processing	Data hashing and signing; Digital signatures; Strong authorization; Tamper-resistant protocols across communication links; Communication link securing with protocols that provides message integrity
	System fault or behavior deviation	Oracle-based system behavior checking
Open system/service with highly-desired availability	Virus, e.g. Trojan horse, Worms, ……	Block all unnecessary ports at the firewall and host; Disable unused functionality; Harden weak, default configuration settings

MicroSoft, Improve Web Application Security;
Common Criteria for Information Technology Security Evaluation;

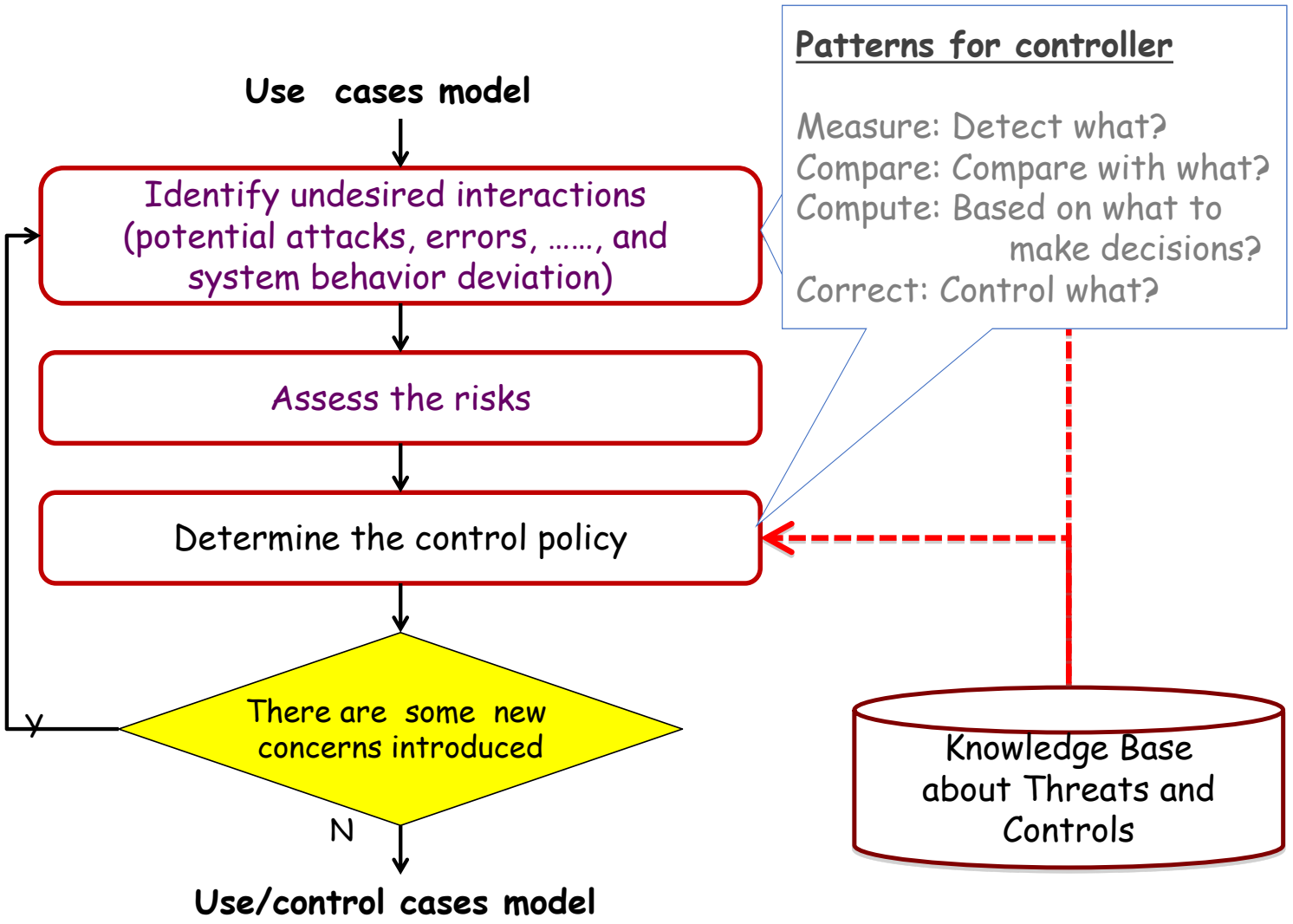
Process of Requirements Elicitation

- 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



Requirements Elicitation Process



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

The threats and system behavior deviations derived from interaction of the use case

Example:

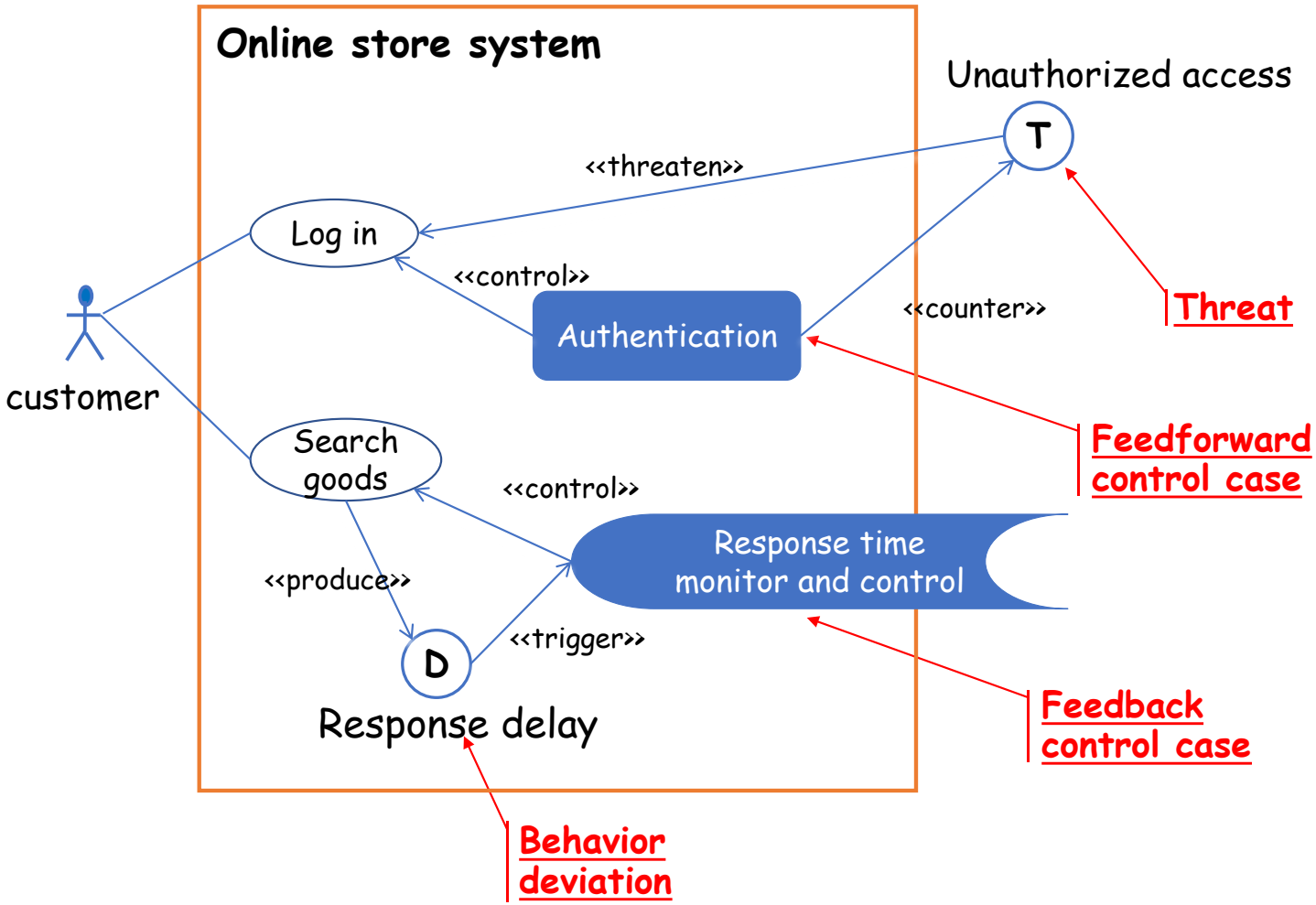
Interaction of use case: log-in:
The customer inputs the account and password.

Properties of interest:
1, The frequency of this interaction,
2, The confidentiality of the account and password

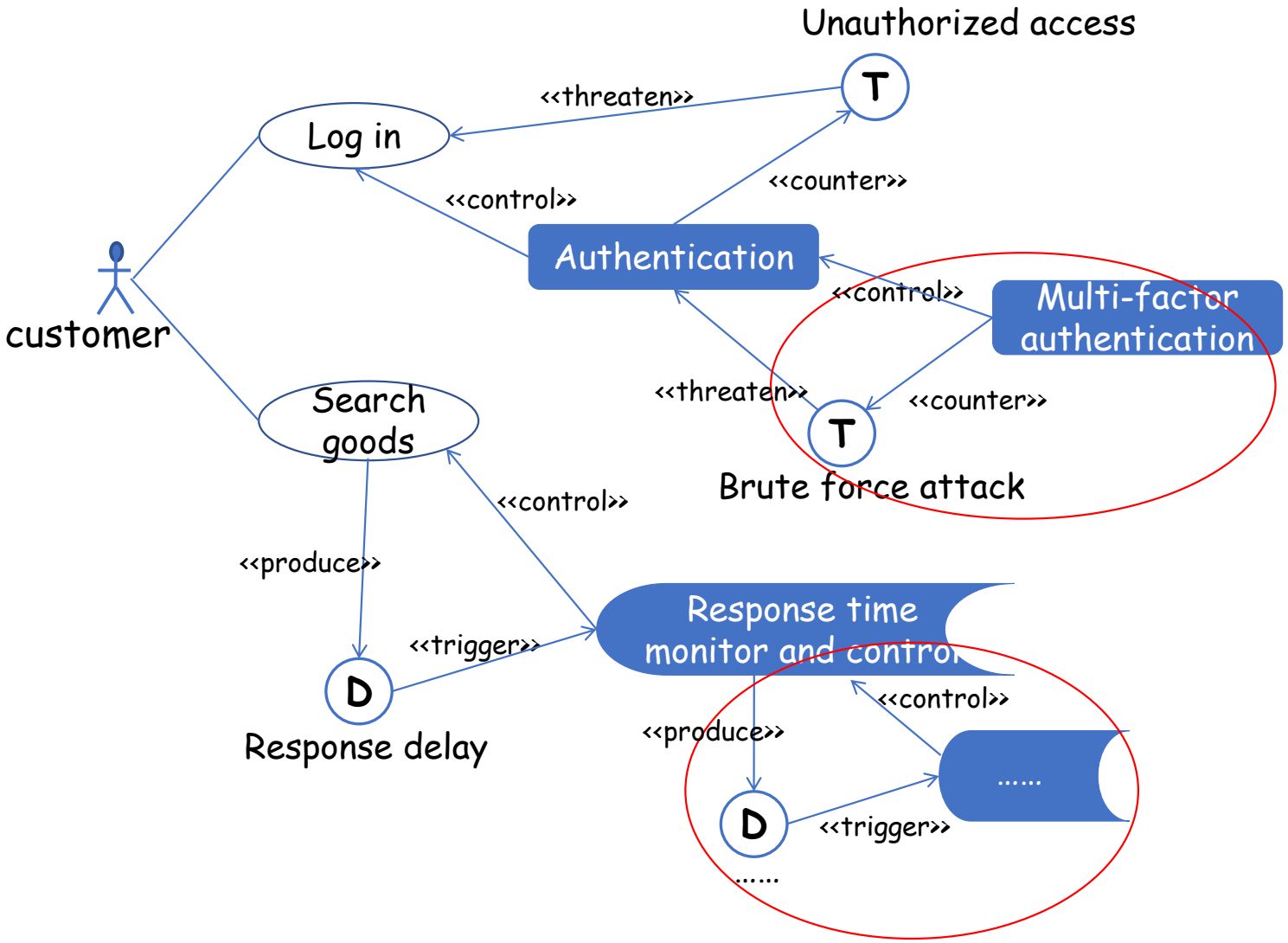
Deviations of properties:
1, This interaction occurs frequently (guideword: more)
2, The account and password is disclosed (guideword: no)

Threats or system behavior deviations:
1, Brute force attack
2, Network monitoring

Use Case / Controller Model



Multi-level Controls

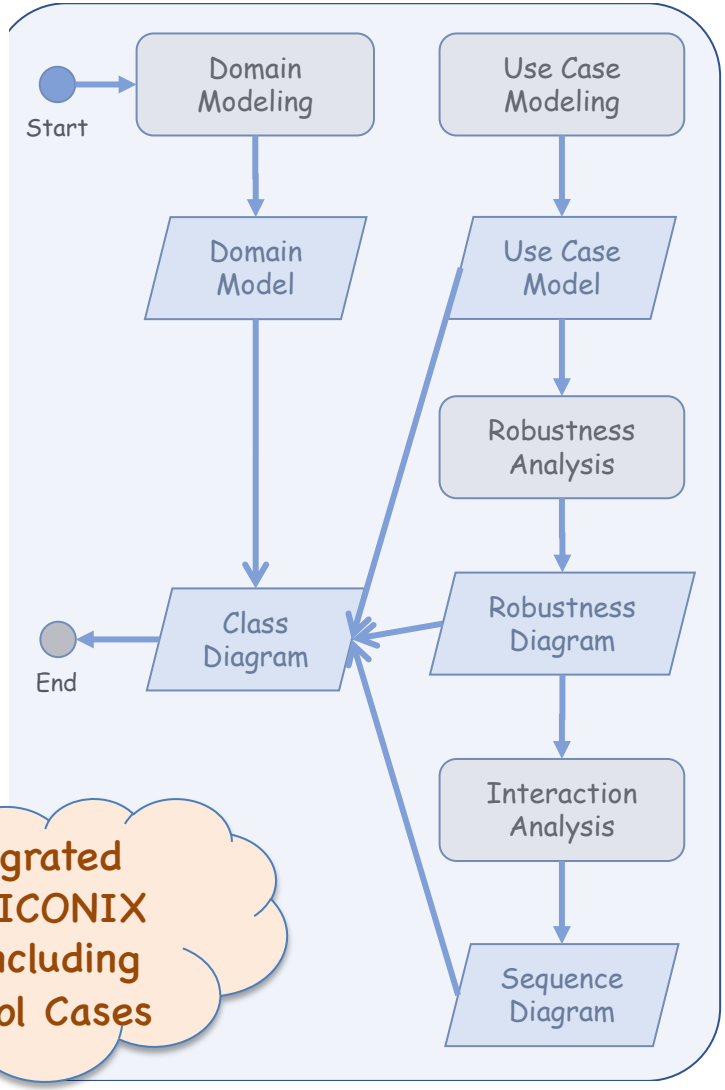


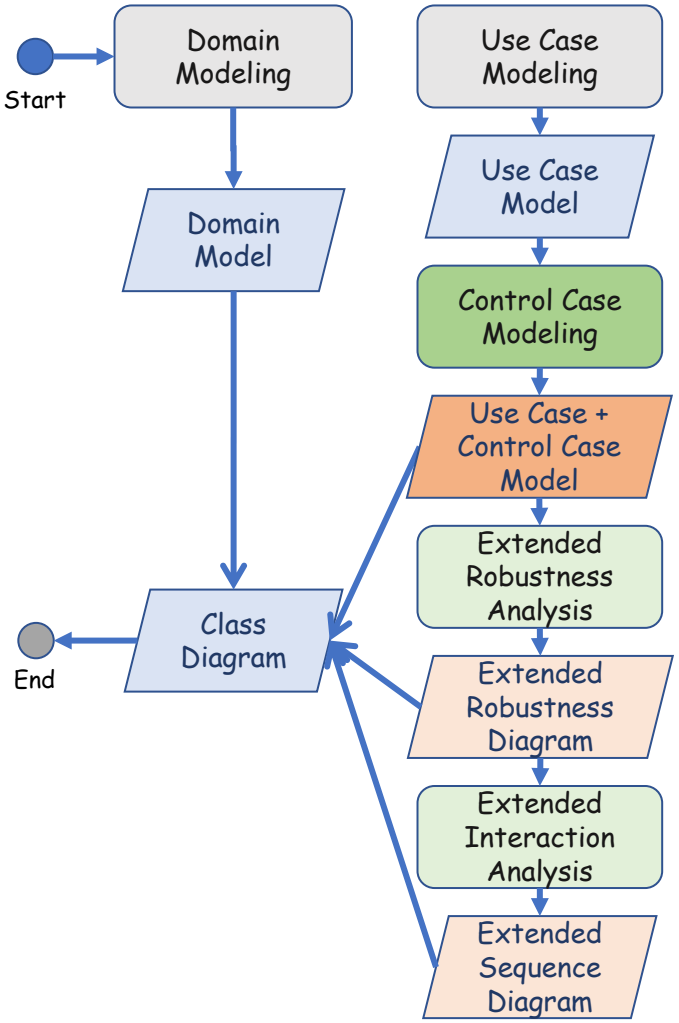
Controller Representation

FFControl case:	Authentic
Controlled use case:	Log in
Stakeholders:	Customer
Threat model: Threat: Threat description:	Unauthori The unaut buy a lot o malicious i of others
Controls:	while the c system nee customer t password, valid, allow deny the l

FBControl case:	Response time monitor and control
Controlled use case:	Search goods
Stakeholders:	Customer, system manager
Behavior deviation model: Behavior deviation: Deviation description:	Response time >30 sec 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
Controls:	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

Requirements Interweaving

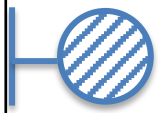




Integrated with ICONIX by Including Control Cases

D. Rosenberg, et al., 2001, Applying Use Case Driven Object Modeling with UML: an Annotated e-Commerce Example, Addison-Wesley

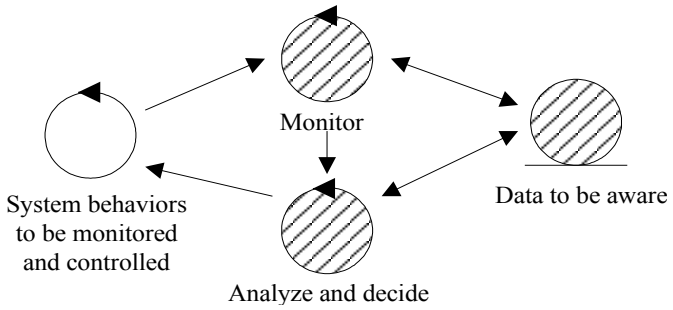
Extended Modeling Icons

Functional Object	Icon	Dependability Object	Icon
Boundary Object		Dependability Boundary Object	
Entity Object		Dependability Entity Object	
Controller		Dependability Controller	

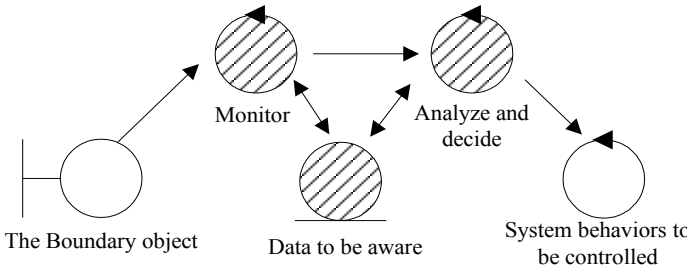
Static / Dynamic Controller Robustness Analysis

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∈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∈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 endfor
End.
```



Pattern for Dynamic Feedback controller



Pattern for Dynamic Feedforward controller

Extended Interaction Analysis

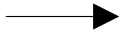
Process: "use case+control case" driven interaction modeling
Input:
US: the set of use cases
CS: the set of control cases
RS: the set of extended robustness diagram
Output:
Extended sequence diagrams
Begin:

```
1  for each extended robustness diagram ers∈ 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 sequ
          diagram;
6     for each control object co in ers
7        allocate the behaviors of co among the collabora
             objects;
8     endfor
9  endfor
```

End.

The symbols for use cases:

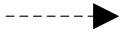
Object for use case



Message for use case

The symbols for control cases:

Object for control case



Message for control case

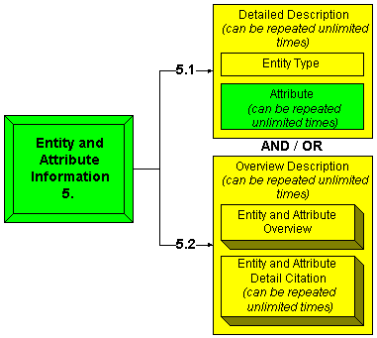
The Whole Process

Use Case Description

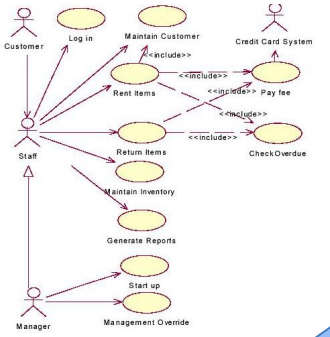
Name	Save item for purchase.
ID	UC_001
Description	While browsing items in the eStore, a user finds an item he is not ready to purchase yet, but he wants to save it to a list so that he can later find the item that he was previously interested in.
Actors	eStore customer.
Organizational Benefits	Increase sales by helping the customer remember products he was previously interested in.
Frequency of Use	20% of users save an item to be bought later each time they visit the site. 50% of saved items are purchased within one year of the saved date.
Triggers	The user selects an option to save an item.
Preconditions	User is viewing an item in the catalog.
Postconditions	The item selected to be saved is visible to the user when he views his saved items. The item selected to be saved is reflected as a saved item when the user views his eStore search and browse results.
Main Course	1. System prompts user to confirm saving selected item instead of purchasing it right away. 2. User confirms to save now (see EX1). 3. System determines user is not logged in and redirects user to log on (see AC1). 4. User logs on (see AC2, AC3). 5. System stores the saved item (see EX2). 6. System redirects the user to their saved items list to view the full list.
Alternate Courses	AC1 System determines user is already logged on. 1. Return to Main Course step 5. AC2 User logs off again. 1. Return user to Main Course step 3. AC3 User does not have an account already. 1. User creates an account. 2. System confirms account creation. 3. Return user to Main Course step 4.
Exceptions	EX1 User decides to purchase the item now. 1. See "Purchase Item" Use Case. EX2 System fails on saving item to list. 1. System notifies user that an error has occurred. 2. Return user to Main Course step 1.

NLP Techniques:
Domain Entity Recognition,
Relation Classification

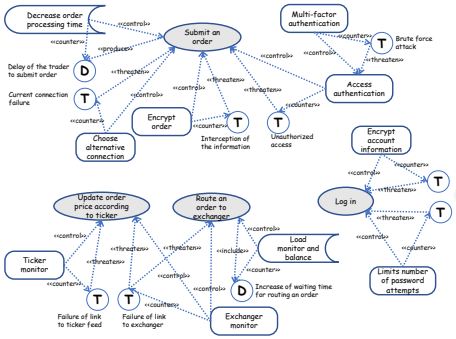
Interactive Entities and Attributes



Use Case Diagram



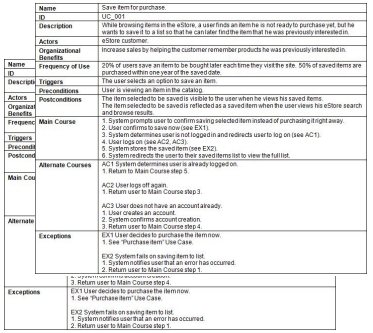
Controllers and Use Case Diagram



Guided Generation

Risk Analysis
Countermeasure Selection
Threats-Countermeasure Knowledge Base

Dependable Use Cases

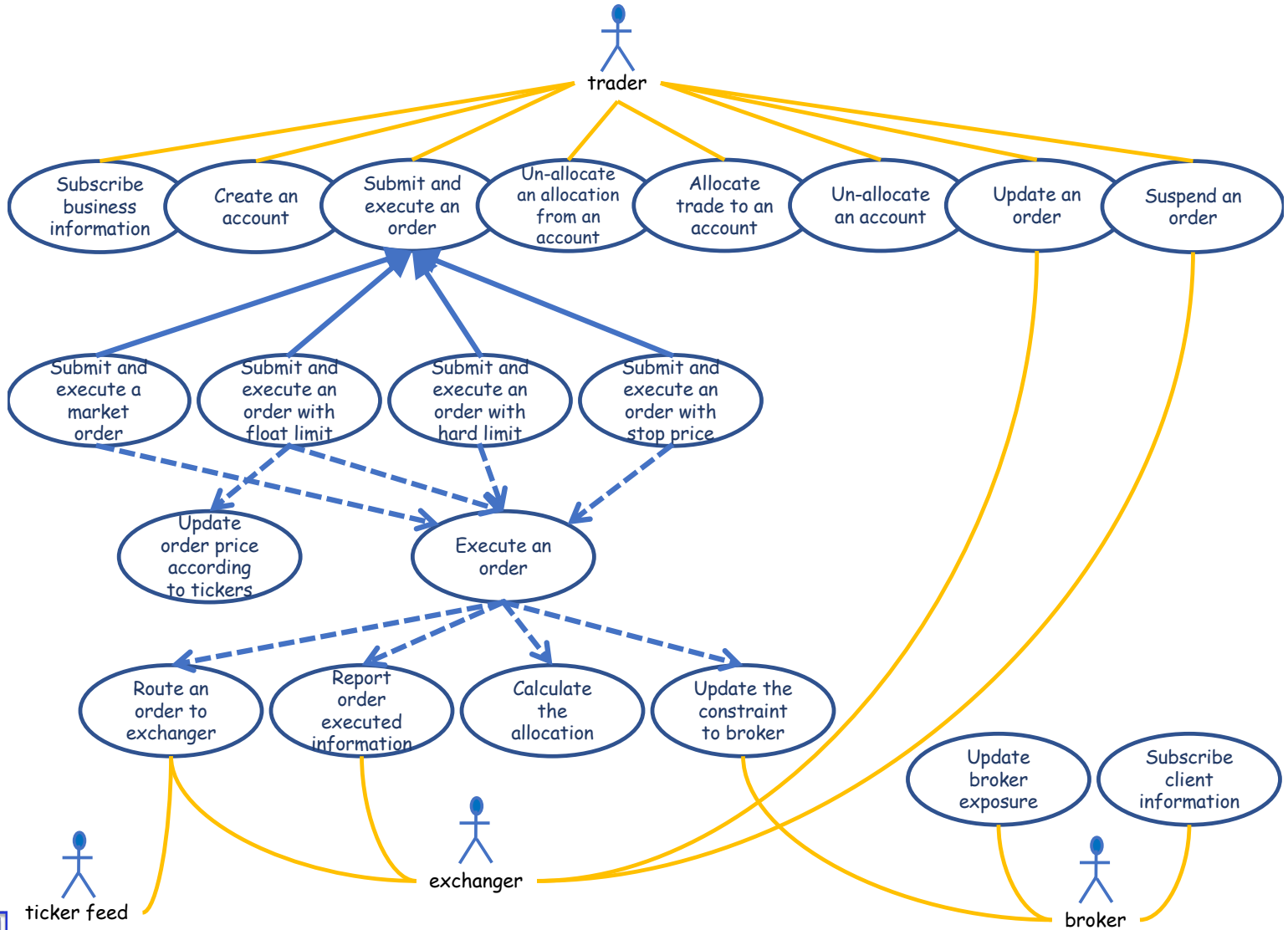


Environment Entity / System Asset	Undesired Feature	Implied Concern
System / System Component	Produce unexpected behavior / output; Failure	Fault tolerance Adaptation Concern
External Entity	Trigger known attack / virus	Security Concern
External Symbolic Entity	Has different levels of sensitiveness	Privacy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

Candidate Controlled Domain, Threats and Concerns

Domain Knowledge
Domain Expert Experience
(Entity's Property, Relation's Property)

A Case Study:
Online Stock Trading System



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

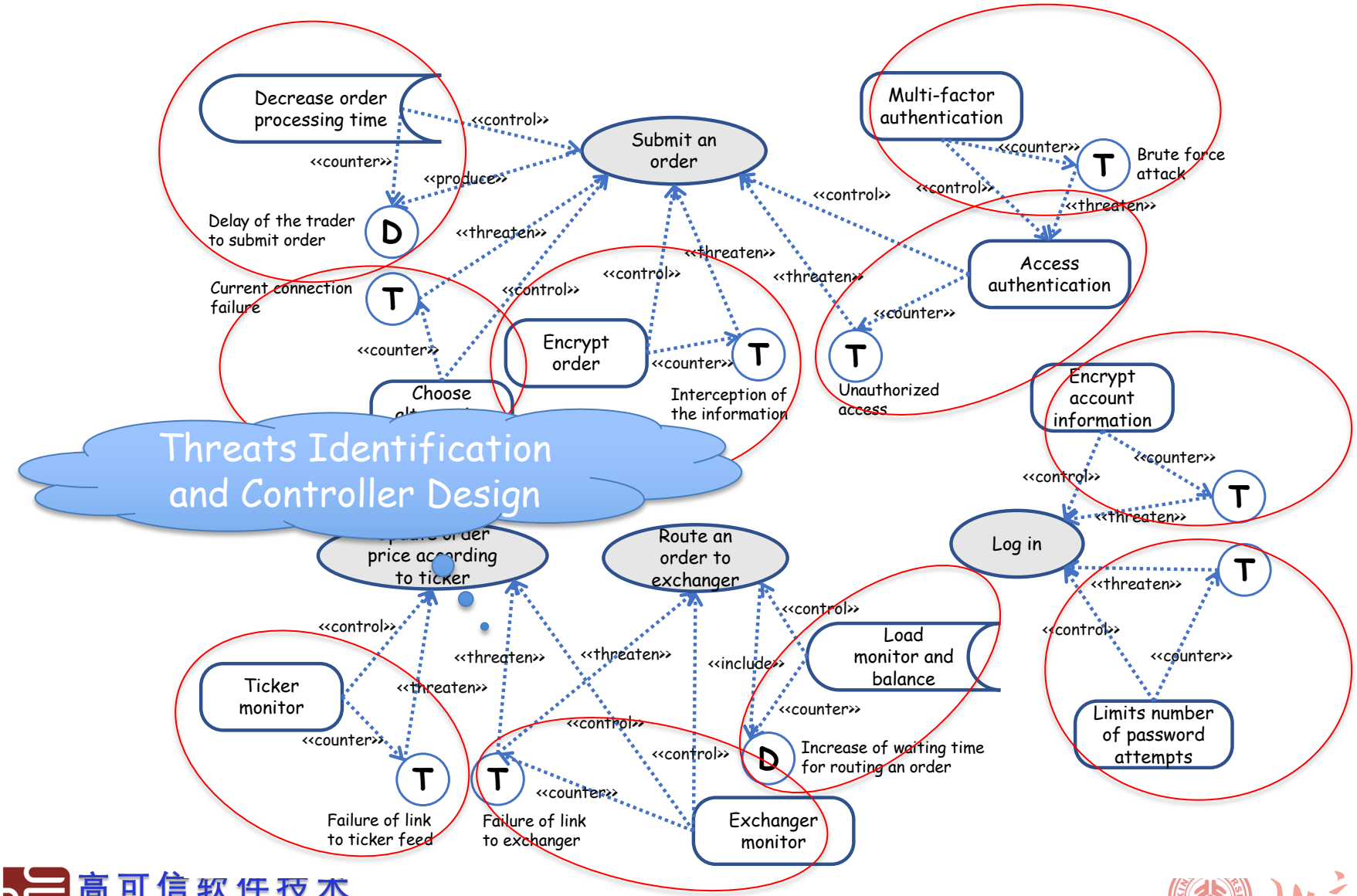
Use Case : Submit an order
Actor: Trader
Preconditions:
- The exchanger which the order will route is connected and can accept instructions from system.
- The trader has logged in.

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



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.

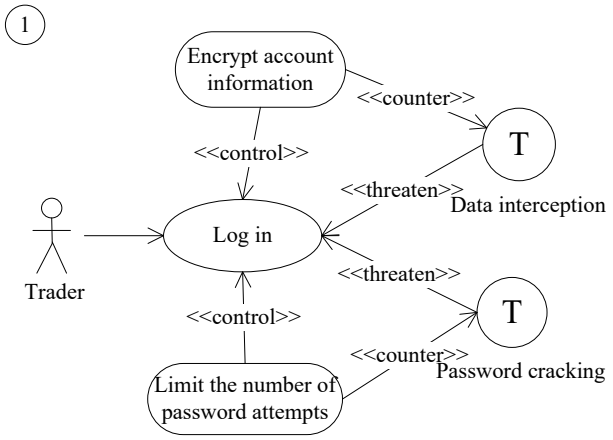
FFcontrol case: limit the number of password attempts
Stakeholder: Trader
Controlled use case: Log in
Threat model:
Threat name: Password cracking
Threat Description: Once some malicious persons know the account name of the trader, he will crack the account password by testing the password again and 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.

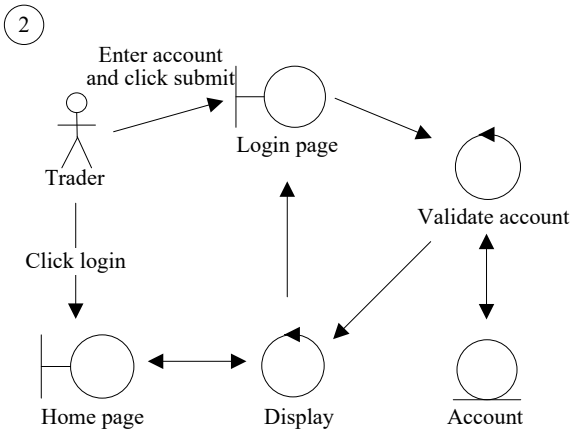
FFcontrol case: Enable alternate connection
Stakeholder: Trader
Controlled use case: Submit an order
Threat model:
Threat name: Exchange connection failure
Threat description: Because of the physical reasons, the connection between the system and eacH exchange may be not available. This will cause that 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 5mintures.
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.

FBcontrol case: Decrease order process time
Stakeholder: Trader
Controlled use case: Submit an order

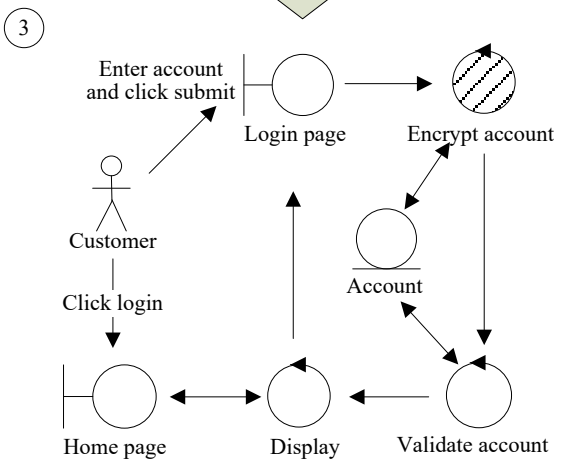
Online Stock Trading System: Log-in Static Controller and Robustness Analysis



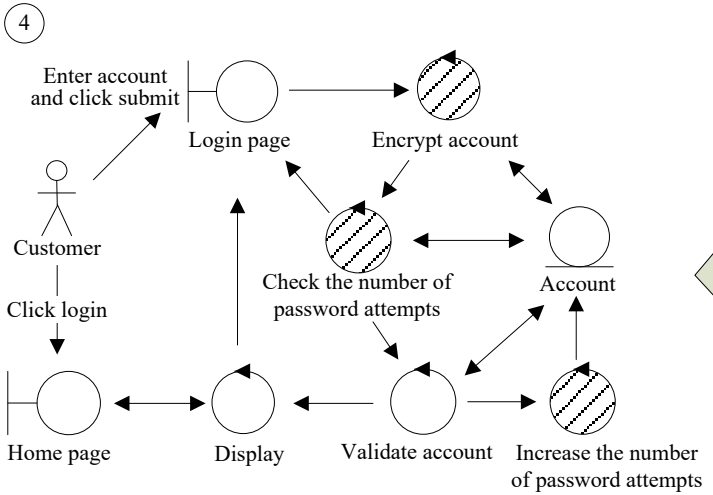
(a) use cases +control cases diagram



(b) analyze use case: log in



(c) analyze control case: encrypt account



(d) analyze control case: limit the number of password attempts

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.

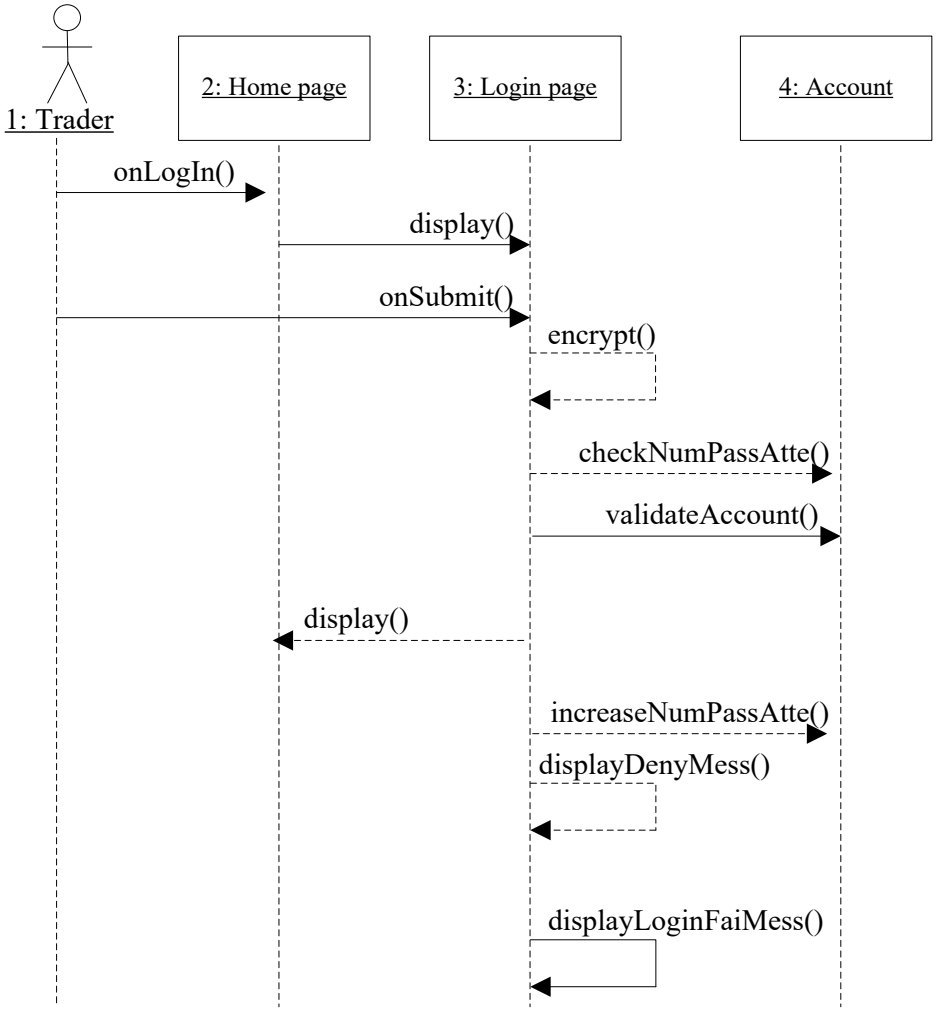
/**limit the number of password attempts:** 1,while the trader enters the account information, the system first checks the number that the trader has attempted, and then validate the account.

2, If the password is right, then return the trader to the Home page.

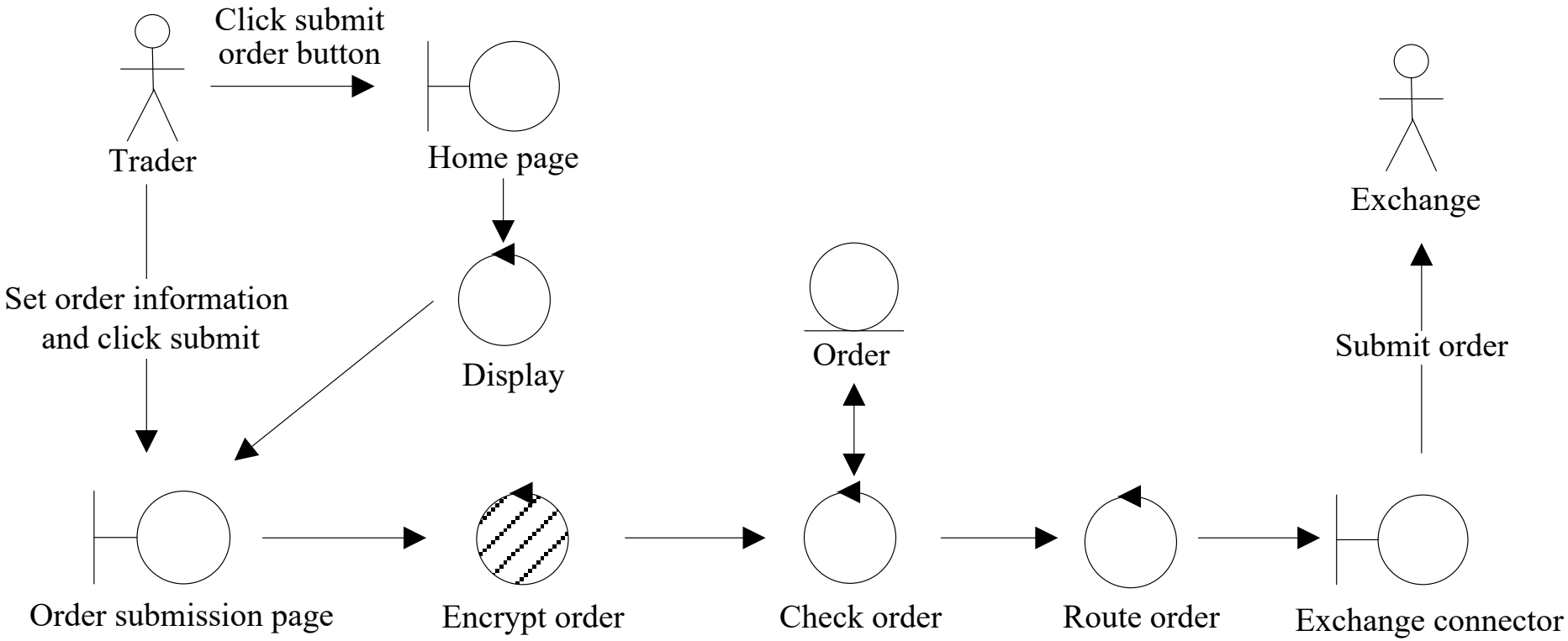
3, If the password is not right, the system needs to increase the number of the password attempts.

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

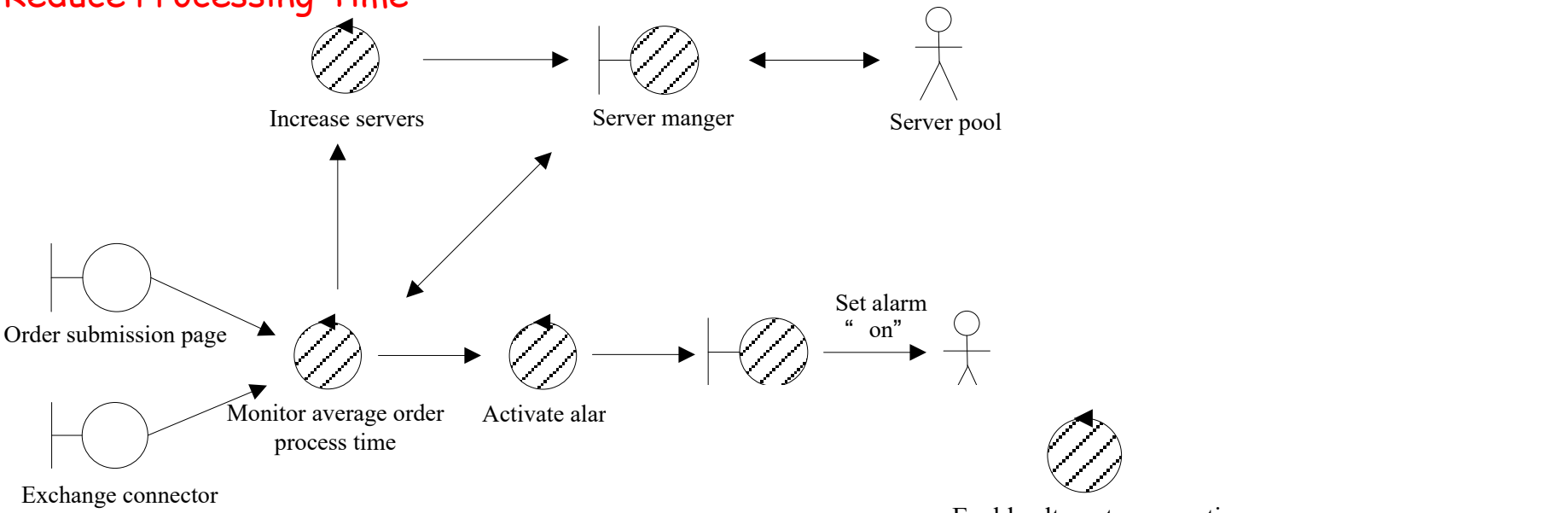


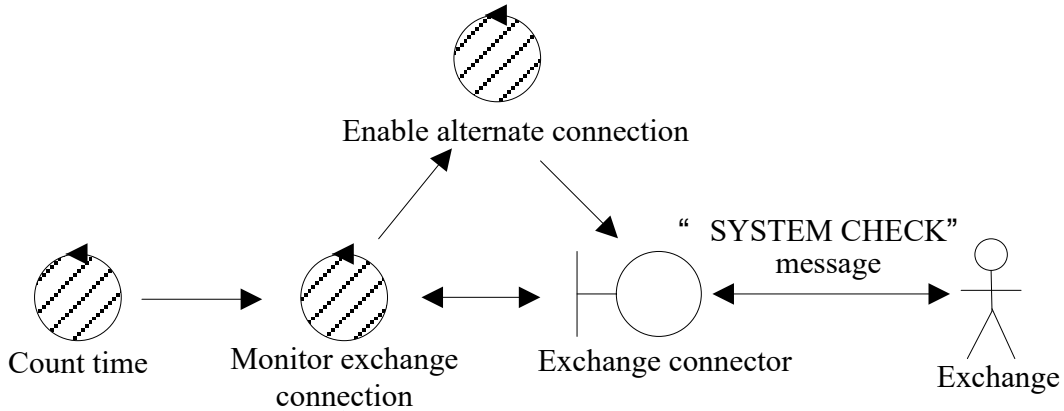
Online Stock Trading System: Submit Order Robustness Analysis Diagram with Static Controller



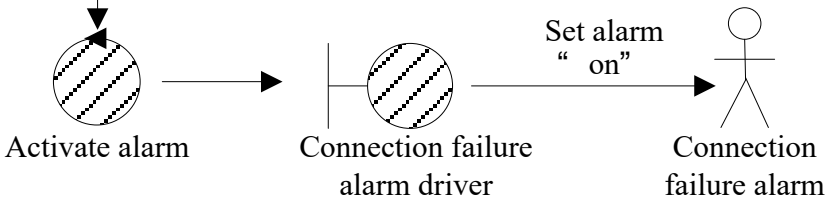
Online Stock Trading System: Submit Order Dynamic Controllers

Reduce Processing Time





Activate Alter Connection



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

Control cases: decrease order process time

Controls:

Alternative1: 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.

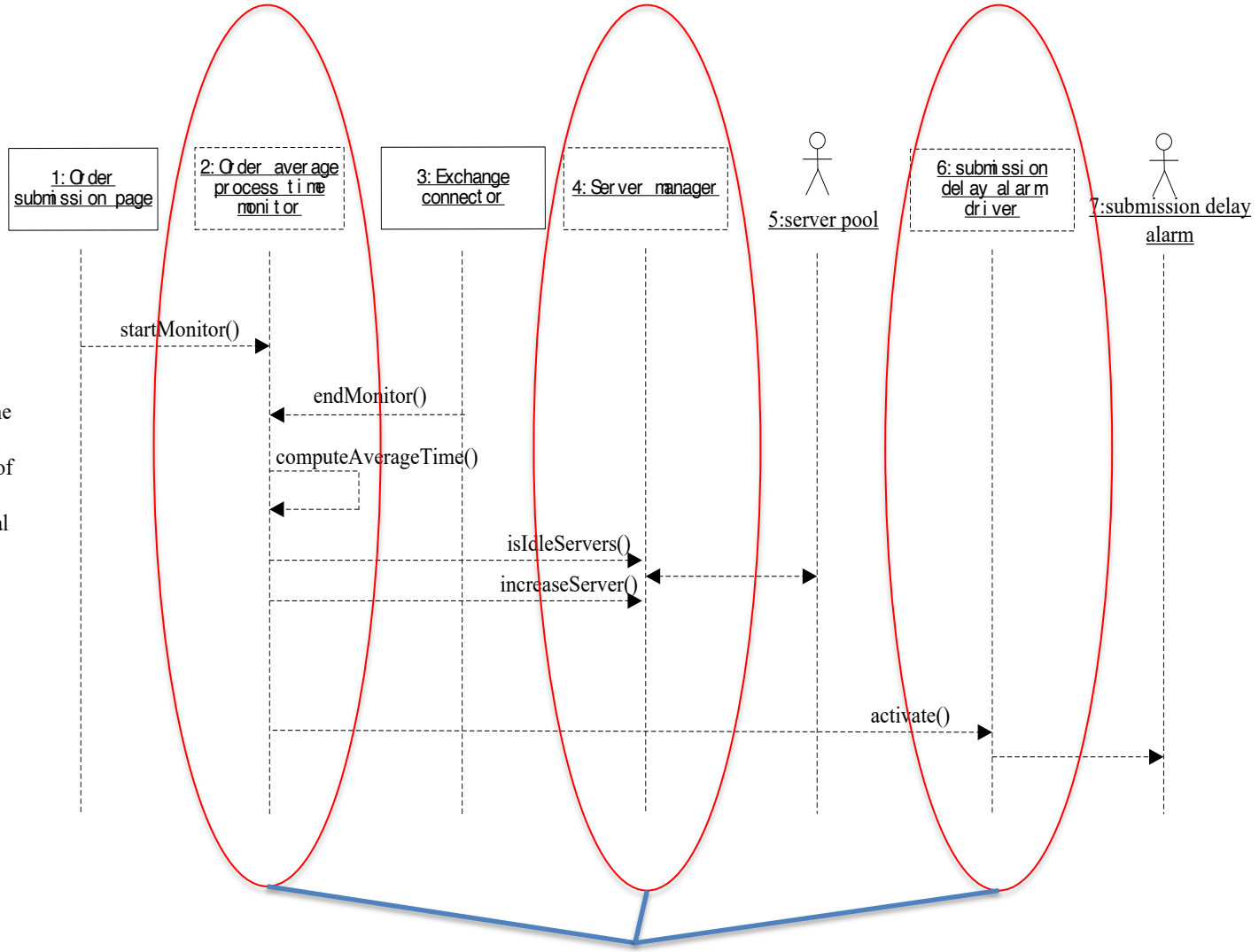
3, The system computes the average time of processing the order.

4, The system allocates more servers to deal with the orders accepted from the traders if the average time>0.8.

Alternative2: 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

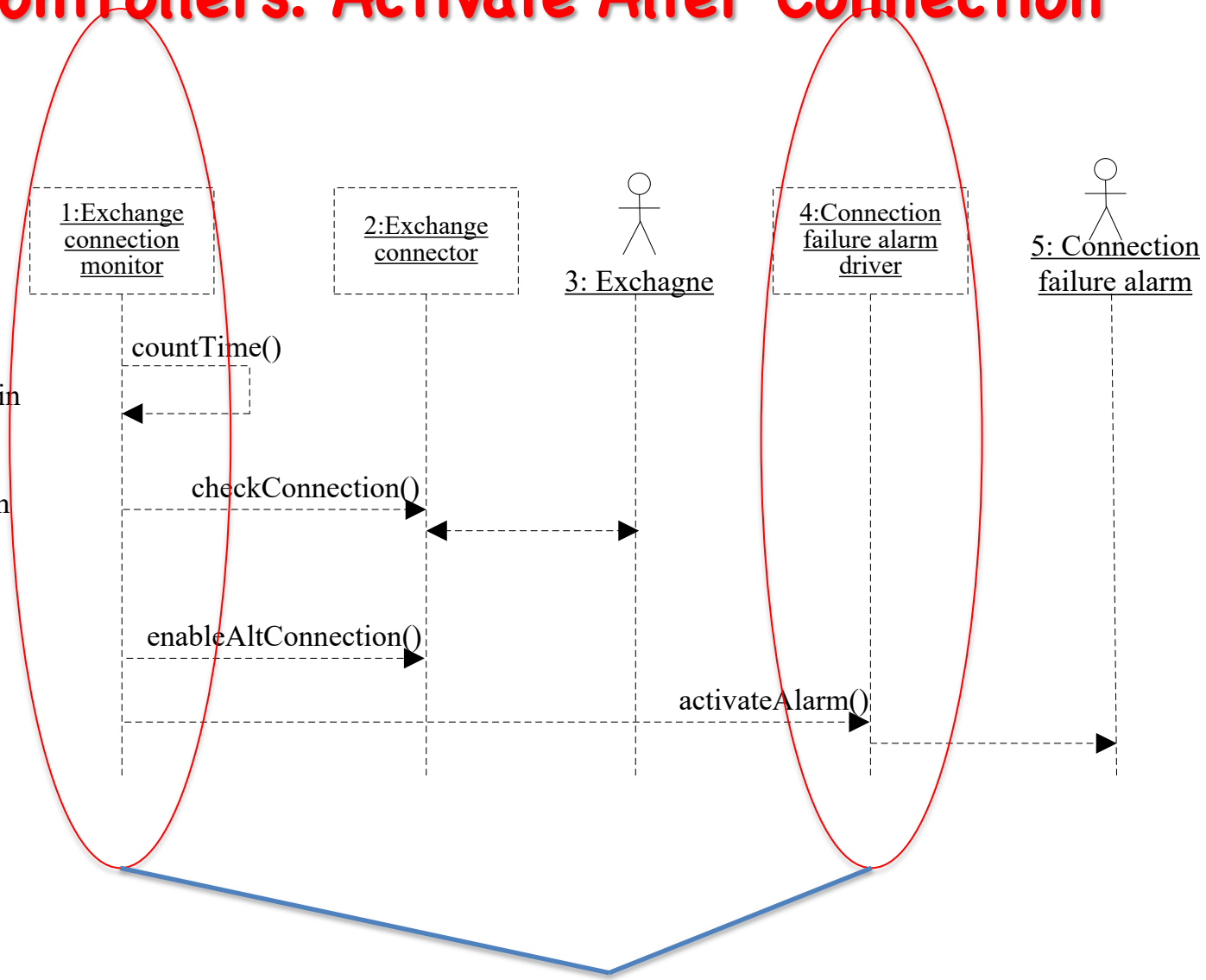
Online Stock Trading System: Submit Order
Dynamic Controllers: Activate Alter Connection

Control cases: enable alternate connection

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

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 interweaved 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)

Summarization

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 Attentions

IEEE

ReliabilitySociety

QRS 2017

The 2017 IEEE International Conference on
Software Quality, Reliability & Security

July 25-29, 2017 • Prague, Czech Republic
http://paris.utdallas.edu/qrs17