Separation of Concerns for Dependable Software Design

Daniel Jackson and Eunsuk Kang

MIT

Nov 7 • FoSER Workshop 2010
Achieving Dependability

Traditional approach
• process + testing: necessary, but not sufficient*
• reliance on *ex post facto* analysis: too late?

Static analysis & verification
• stronger guarantees, but need guidance
• correctness proof $\Rightarrow$ dependability

A Different Approach

Dependability case

• explicit, end-to-end argument

• $\text{ENV} \land \text{SPEC} \Rightarrow \text{REQ}$

Design for dependability

• most critical requirements first

• smaller trusted base $\Rightarrow$ simpler case, lower cost
Mixed-Criticality System

Many critical properties are *partial*
- factor out from full functional requirements
- ex. “perform good op X” vs. “prevent bad op Y”

Non-uniform allocation of resources
Example: Online Bookstore

Two requirements

• ordering: “fulfill a customer order”
• secrecy: “don’t leak a customer’s credit card”
Trusted Base for Secrecy

Customer
- id
- address
- credit card
- shopping cart
- orders

Order
- date
- books
- customer
- fulfill

ShoppingCart
- books
- customer
- makeOrder

CreditCardAPI
- charge

Card
- number
- name
- expiry date

Book
- ISBN
- title
- price
- reviews

Secrecy
Alternative Design

Coordinator
- makeOrder
- fulfill

Customer
- id
- address
- shopping cart

Orders
- orders: CustomerId -> Order
  - makeOrder fulfi

CreditCardAPI
- charge

Cards
- cards: CustomerId -> Card

ShoppingCart
- books
- customer

Book
- ISBN
- title
- price
- reviews

Order
- date
- entries
- customer id

Entry
- ISBN
- title
- price

Card
- number
- name
- expiry date
Discussions

Dependability case
• if you can’t say why it works, it probably doesn’t

Design for dependability
• untapped potential; shift in research focus?

Our on-going research
• design method for small trusted bases
• case studies: Tokeneer, radiation therapy, e-voting