Designing Software Systems to be Robust

Eunsuk Kang

Software Engineering Seminar
University of Waterloo
June 20, 2023
Environment Assumptions

To establish its requirements, a system relies on various assumptions about the environment.

Nurse performs actions in correct order
Network latency is at most 50 ms
Attacker doesn’t have access to secret key
Environment Deviations

What happens if the environment deviates from these assumptions?

Nurse inadvertently omits a critical action
Network experiences an unexpected disruption
Attacker obtains a secret through a side channel

Does the system still provide any guarantees?
Panama City Public Hospital (2001)

Therapy planning software by Multidata Systems
Theratron-780 by Theratronics (maker of Therac-25)

Shielding blocks
Inserted into beam path, protect healthy tissue
Therapist draws block shapes; SW computes dose
Therapist Interaction

dose = D
Accidents

28 patients overdosed; 21 deaths
Blame User or Software?

Multidata Systems

“Given [the input] that was given, our system calculated the correct amount, the correct dose. And, if [the staff in Panama] had checked, they would have found an unexpected result.”

Three therapists charged & found guilty for involuntary manslaughter; barred from practice
The environment will occasionally deviate from its expected behavior.

A robust system should ensure its critical properties even under such deviations.
Successful engineering products are designed with a margin of safety that provides layers of protection against abnormal events.
Robust Systems by Design

1) Devise an initial system design
2) Identify types of deviations in the environment
3) Analyze the design to check whether it’s robust against those deviations
4) If not, redesign the system to improve its robustness

But in software:
What exactly do we mean by “robust”?
How do we verify that a system is sufficiently robust?
How do we systematically improve its robustness?
Robust Software Design: Roadmap

**Specification**
What does it mean for our system to be robust?

**Analysis**
How robust is our system?

**Robustification**
How do we improve its robustness?
Robust Software Design: Roadmap

**Specification**
What does it mean for our system to be robust?

**Analysis**
How robust is our system?

**Robustification**
How do we improve its robustness?
Robust Software Design: Roadmap

**Specification**
What does it mean for our system to be robust?

**Analysis**
How robust is our system?

**Robustification**
How do we improve its robustness?
What exactly does it mean for software to be robust?
Robustness: High-Level Idea

The maximum amount of environmental deviations under which the system is capable of satisfying a desired property
Robustness: High-Level Idea

\[ M \parallel E \models P \]

Software Specification
System Property
Assumptions
Robustness: High-Level Idea

\[ M \parallel E \models P \]

\[ \delta \]

\[ E' \]

Deviations
Robustness: High-Level Idea

System $M$ is robust against a set of deviations $\delta$ with respect to environment $E$ and property $P$.
Robustness: Behavioral View

\[ M \parallel E \models P \]

LTS: Labelled transition system

Safety property
Simple but expressive formalism
Behaviors of an LTS ≡ Possible traces (event sequences)
Robustness: Behavioral View

\[ M \parallel E \models P \]

Key Idea: Represent & compute \( \delta \) as traces

- LTS
- LTS
- Safety property
- Traces (not in \( E \))
- \( E' \)
- \( E \) with additional behaviors (\( \delta \))
Robustness: Behavioral View

\[ M \parallel E \models P \]

\[ \Delta \]

\[ M \parallel E' \models P \]

Robustness \( \Delta(M, E, P) \)

The largest set of possible deviations under which the system can ensure property \( P \)
Example: Therac-25
Radiation Modes in Therac-25

Electron ("Ebeam") and X-ray modes
Insert collimator during X-ray for safe radiation level
Safety Hazard

Electron Mode
Low power
Wide area

X-Ray Mode
High power
Narrow area

THE PROBLEM
High power
Wide area

X-ray mode & collimator out → Possible overdose!
Caused several fatal injuries in Therac-25
Establishing Safety Property

\[ M \parallel E \models P \]

- **Software model** (interface + controller)
- **Safety property** “No radiation overdose”
- **Operator task description**

\[ M, E \quad \text{Labeled transition systems} \]
\[ P \quad \text{Safety property} \]
Therac-25 Design

\[ M = M_I \parallel M_B \parallel M_S \]
Modeling Operator Behavior

Possible traces in $E$:
$\langle X, \text{Enter}, B \rangle$
$\langle E, \text{Enter}, B \rangle$

Captures expected sequences of users actions
Typically specified in a training manual or user instructions
Establishing Safety Property

\[ M \parallel E \models P \]

Under “expected” operator behavior, system satisfies the safety property!
Deviations
Operator Error

“…[Therapist] noticed that for mode she had typed "x" (for X ray) when she had intended "e" (for electron)...the mistake was easy to fix; she merely used the cursor up key to edit the mode entry.”

An Investigation of the Therac-25 Accidents
Leveson & Turner, IEEE Computer, 1993
Modeling Operator Error

What if the operator commits an error?
1. Selects X-ray mode by mistake
2. Realizes error, presses UP to go back
3. Selects Ebeam mode and proceeds

Erroneous operator \( (E') \)
Deviation as Additional Behavior

A new trace (deviation) absent in $E$

Erroneous operator ($E'$)
System under Deviated Environment

\[ M \parallel E \models P \]

\[ M \parallel E' \models P ? \]

Perfect operator

Operator mistake

Erroneous operator
Safety Violation

What if the operator commits an error?
1. Selects X-ray mode by mistake
2. Realizes error, presses UP to go back
3. Selects Ebeam mode and proceeds
4. Collimator is removed for Ebeam
5. When user fires, radiation setting is still transitioning from X-ray to Ebeam
6. Safety violation!
System under Deviated Environment

\[ M \parallel E \models P \]

\[ M \parallel E' \not\models P \]

System is not robust; i.e., fails to be safe under the deviated environment!
Robustness: Another View

$\alpha I^*$

$M$ : Radiation therapy system
$E$ : Operator behavior
$P$ : “No overdose”
$\alpha I$ : Interface events
$\alpha I^*$ : All possible traces over $\alpha I$
Robustness: Another View

All environmental behaviors accepted by system

\[ \langle X, \text{Enter}, B \rangle \]
\[ \langle E, \text{Enter}, B \rangle \]
\[ \langle X, \text{Up}, E, \text{Enter}, B \rangle \]
\[ \langle E, \text{Up}, X, \text{Enter}, B \rangle \ldots \]
Robustness: Another View

Normative (i.e., expected) environment behavior

$\langle X, \text{Enter}, B \rangle$

$\langle E, \text{Enter}, B \rangle$

System satisfies its property ("no overdose") under these env. behaviors
Robustness: Another View

System may or may not satisfy its property under these deviations

\langle X, \text{Up, E, Enter, B} \rangle
\langle E, \text{Up, X, Enter, B} \rangle...
Robustness: Another View

System may violate property under these behaviors (called intolerable deviations)

⟨X, Up, E, Enter, B⟩
Robustness: Definition

Robustness $\Delta(M, E, P)$
System satisfies its property under these deviations

$\langle E, \text{Up}, X, \text{Enter}, B \rangle$...

Robustness ($\Delta$) is a computable, first-class property of a system!
Our definition enables new types of design analysis tasks
Robust Software Design: Roadmap

**Specification**
What does it mean for our system to be robust?

**Analysis**
How robust is our system?

**Robustification**
How do we improve its robustness?
Verification Problem

Env. ($E$) → Verification Tool

System ($M$) → Verification Tool

Property ($P$) → Verification Tool

Does $M$ satisfy $P$ under $E$?

Yes → Satisfied

No → Counter-example
Robustness Analysis

- Env. \((E)\)
- System \((M)\)
- Property \((P)\)

Robustness Analyzer

- \(\Delta(M, E, P)\)
- \(\tilde{\Delta}(M, E, P)\)

How robust is \(M\) w.r.t. \(E\) and \(P\)?

What are intolerable deviations?
Robustness Analysis

Operator errors that system can handle
$\langle E, \text{Up}, X, \text{Enter}, B \rangle$

Operator errors that result in a violation
$\langle X, \text{Up}, E, \text{Enter}, B \rangle$

Robustness Analyzer

Operator task

Therac-25 model

“No overdose”
Robustness Comparison

Is one design ($X$) more robust than another ($Y$)?

$\Delta_X = \Delta(M_X, E, P)$

$\Delta_Y = \Delta(M_Y, E, P)$
Robustness Comparison

Operator task

Therac-20 model

Therac-25 model

“No overdose”

Robustness Analyzer

Is Therac-20 more robust than Therac-25 against operator errors?

\[ \Delta_X - \Delta_Y \]

Operator errors that Therac-20 can tolerate but 25 can’t

\[ \Delta_X = \Delta(M_X, E, P) \]
\[ \Delta_Y = \Delta(M_Y, E, P) \]
Analyzing Robustness
Robustness Analysis

How robust is $M$ w.r.t. $E$ and $P$?

Technical challenges
1. Computing $\Delta(M, E, P)$
2. Choosing its representation
Computing Robustness

**Challenge #1**

Infinite number of possible deviations

How do we find a maximum set that the system is robust against?
Computing Robustness

Weakest assumption (WA)
Set of all environmental behaviors under which M satisfies P

Assumption generation for software component verification.
Giannakopoulou, Pasareanu, and Barringer. ASE 2003.
Computing Robustness

Weakest assumption \( (WA) \)
Set of all environmental behaviors under which \( M \) satisfies \( P \)

1. Compute \( WA \) using assumption generation method
2. Compute the difference over \( E \) (i.e., \( WA - E \))
Representing Robustness

Challenge #2

Δ is an infinite set of traces

How do we represent this information to the designer?
Trace Partitioning

Group $\Delta$ into a finite number of equivalence classes. Each class represents a particular type of deviation (e.g., omission, repetition, intrusion error...)

Deviation class #1
Deviation class #2
Deviation class #3…
Deviation Patterns in Human Errors


58
Analysis Process

More details in our paper!

A behavioral notion of robustness for software systems.
Analysis Case Studies

Safety-critical interfaces
Robustness against human errors

Network protocols
Robustness against unreliable network faults
Network Communication Protocols

Two versions
(1) “Naive” protocol
(2) Alternating bit protocol (ABP)

Property: “Message delivered in correct order”
Network Faults as Trace Deviations

What could happen in an unreliable channel?

Packet duplication
\[\langle \text{send}[0], \text{rec}[0], \text{ack}[0], \text{getack}[0] \rangle\]

\[\langle \text{send}[0], \text{rec}[0], \text{rec}[0], \text{ack}[0], \text{getack}[0] \rangle\]

Packet corruption
\[\langle \text{send}[0], \text{rec}[0], \text{ack}[0], \text{getack}[0] \rangle\]

\[\langle \text{send}[0], \text{rec}[1], \text{ack}[0], \text{getack}[0] \rangle\]
Comparing Network Protocols

Perfect Channel \((E)\)

Naive \((M_{\text{naive}})\)

ABP \((M_{ABP})\)

Robustness Analyzer

What types of faults ABP is more robust against?

\(\Delta_{ABP} - \Delta_{naive}\)

“Message delivered in correct order” \((P)\)

ABP is more robust against:
- message loss, duplication
- fabricated acknowledgments
Analysis Case Studies

Safety-critical interfaces
Robustness against human errors

Network protocols
Robustness against unreliable network faults

Our definition captures deviations in multiple domains
Robustness can be computed under several seconds
Robust Software Design: Roadmap

**Specification**
What does it mean for our system to be robust?

**Analysis**
How robust is our system?

**Robustification**
How do we improve its robustness?
Robustifying Systems
\( M \parallel E \models P \)

\( M \parallel E' \not\models P \)

Intolerable deviations
Can we generate suggestions for enhancing the original design to handle additional deviations?
Robustification

Can we generate suggestions for enhancing the original design to handle additional deviations?
Treat it as a model transformation problem!
LTS Transformation

Challenge
Infinite number of possible modifications!
Robustification as Supervisory Control

Robustification problem → reduce → Supervisory control problem
Supervisory control of a class of discrete event processes.

Supervisory Control

Given plant $G$, find supervisor $S$ such that $G$ under the control of $S$ satisfies property $P$. 

$S \parallel G \models P$
Supervisory Control

Based on plant’s past behavior, supervisor restricts the set of events that it is allowed to perform.

Disables some subset of plant events

enabled events

history of plant events
Supervisory Control: Applications

What’s safe to do in the current conveyor belt state?

\( P: \) “No incorrect assembly”

Others: Network security, concurrency control, protocol synthesis…

Diagram:

- **Supervisor**
- **Plant**
- **Manufacturing robot**
- **Past robot actions**
- **Pick up item, move arm, …**

Relationships:

\( h \) from **Past robot actions** to **Supervisor**

\( X(h) \) from **Pick up item, move arm, …** to **Manufacturing robot**
Robustification as Supervisory Control

Robustification problem \rightarrow \text{reduce} \rightarrow \text{Supervisory control problem}
Robustification as Supervisory Control

\[ M \parallel E' \not\models P \]

Deviated environment
Robustification as Supervisory Control

![Diagram](image)
Robustification as Supervisory Control

\[ S \parallel G \models P \]
\[ \iff S \parallel (M \parallel E') \models P \]
Therac-25 Example

Additional logic for handling operator errors

Past events

Therac-25 design

Enabled operator actions

$P$: “No overdose”
Recall: Safety violation under erroneous operator

Interface ($M_I$)

Collimator ($M_S$)

Mode setter ($M_B$)
Supervisor Behavior

Keep track of system state

Disable event(s) that could lead to a violation

Complications
Not all events are observable/controllable by supervisor
Non-determinism: Multiple possible states to keep track of
Therac-25 Example

\[ S \parallel G \models P \]
\[ \iff S \parallel (M \parallel E') \models P \]

Additional logic for handling operator errors

Diagram:
- Supervisor \( S \)
- Software \( M \)
- Environment \( E' \)
- Plant \( G \)
- \( h \) and \( X(h) \) connections
Robustification as Supervisory Control

\[ S \parallel (M \parallel E') \models P \]

\[ \iff (S \parallel M) \parallel E' \models P \]

\[ \iff M' \parallel E' \models P \]
Synthesized Solution for Therac-25

Additional check to ensure mode transition before beam firing

Redesigned Interface ($M_I$)
Quality of Redesign

Multiple possible solutions, not all desirable!
Some supervisors may disable more events than needed
Ideally, find a solution that
(1) Preserves as many existing behaviors in $M$ as possible
(2) Picks the simplest supervisor possible
Optimal Robustification

Permissiveness
(Amount of preserved behaviors)

Redesign Complexity
(# events used by supervisor)

Trade-offs between these dimensions!
Multiple, possible Pareto-optimal solutions

Pareto-optimal
Robust Design Framework

More details in paper!

Robustification of Behavioral Designs against Environmental Deviations
CJ Zhang et al., ICSE 2023
Robustification Case Studies

Electronic voting
Voter errors
Malicious officials

Infusion pump
Therapist errors
Power and alarm failure
FRANKFORT — A former Clay County precinct worker testified Friday that top election officers in the county taught her how to change people’s choices on voting machines to steal votes in the May 2006 primary.

ES&S iVotronic, Kentucky
Voters exits the voting booth before pressing “confirm” Malicious official enters booth, press “back” & modify the vote
Electronic Voting Interface

Property: The machine must record the vote as selected by the voter

Introduce a deviation to capture voters omitting “confirm”
Use robustification to generate suggested enhancements
**Synthesizing Solutions**

**Redesign #1**
Disables “back” action
**Simple**, but **not permissive**
Does not allow the voter to modify their selection

**Redesign #2**
Disables confirm while the official is in the booth
**More permissive**: Allows vote change
But also more **complex**: Requires keeping track of booth occupant
Robustification Case Studies

Electronic voting

Infusion pump

Our method can automatically synthesize optimal robustification solutions
For complex models (~760 states), < 20 secs
Other On-going Works
Can we enhance the original design to tolerate additional deviations in the environment?
Robustness through Req. Weakening

\[ M \parallel E \models P \]

\[ M \parallel E' \models P' \]

**Self-adaptive framework**
Temporarily weaken \( P \) to a weaker variant \( (P') \) that is (1) acceptable to the user & (2) satisfiable in the deviated environment
Robustness of AI-based Systems

System as a composition of AI & “traditional” SW

How robust is the overall system against:
(1) Deviations in the environment?
(2) Mistakes in the learning-based components?

How do we validate & measure robustness in such systems?
Takeaway

Software-intensive systems depend on various assumptions about the environment.

The environment may deviate from its expectations due to misbehavior or changes.

To ensure critical properties, systems should be designed to be robust against possible deviations.
Robust Software Design: Roadmap

**Specification**
What does it mean for our system to be robust?

https://github.com/cmu-soda/Fortis

**Analysis**
How robust is our system?

**Robustification**
How do we improve its robustness?