Analyzing Dependability in Embedded Systems From the User Perspective

Christopher Martin **Beth Latronico** **Philip Koopman**

Workshop on Reliability in Embedded Systems

20th Symposium on Reliable Distributed Systems

October 28, 2001

GM

RoSES

Institute for Complex Engineered Systems

Carnegie Mellon

Electrical & Computer ENGINEERING

Outline

- **Introduction & Problem Statement**
- **RoSES Project**
- **Motivation**
  - People are a natural part of redundancy
- **Related Areas**
  - Expand beyond realm-specific techniques
- **Approach**
  - Introduce the User Mission Graph concept
- **Example**
  - Apply techniques to a sample elevator subsystem
- **Conclusion**
  - User flexibility can be a part of dependability assessment

Introduction & Problem Statement

- **Aim is to examine design methodologies that increase dependability**
  - For our purposes, we take *dependability* to be defined as:

    "Trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers." [Laprie92]

- **Coming up with a single 'dependability number' for a complex system is hard**
  - Confluence of hardware, software and HCI makes life difficult
  - Go beyond composing individual component reliability estimates

- **What can we do differently than existing approaches to better evaluate dependability?**

RoSES Project

- **Robust Self-configuring Embedded Systems (RoSES)**
- **Robustness gained with automatic graceful degradation**
  - Must not require human intervention to specify or guide
- **First shot → automated reconfiguration when fault detected**

- **Domain -- Distributed Embedded Systems**
  - Distributed – functionality remains after most failures
  - Smart sensors – general compute capability
  - Most functionality is optimization

- **Examples: elevators, autos, copiers, plant control, …**
  - Not Internet toasters

Motivation (What do we want?)

- **People can be a natural part of redundancy**
  - How can we take into account the ability of a user to interact with a system in light of partial system failures?
  - People could take advantage of global workarounds that enhance dependability

- **User perspective is important because reliability is measured from user's perspective!**
  - Complete path is important, not just individual functions
  - Implicit state information in people that system won't know about

- **Need something that works at design time and incorporates system view**
  - Relative vs. absolute dependability

Related Areas

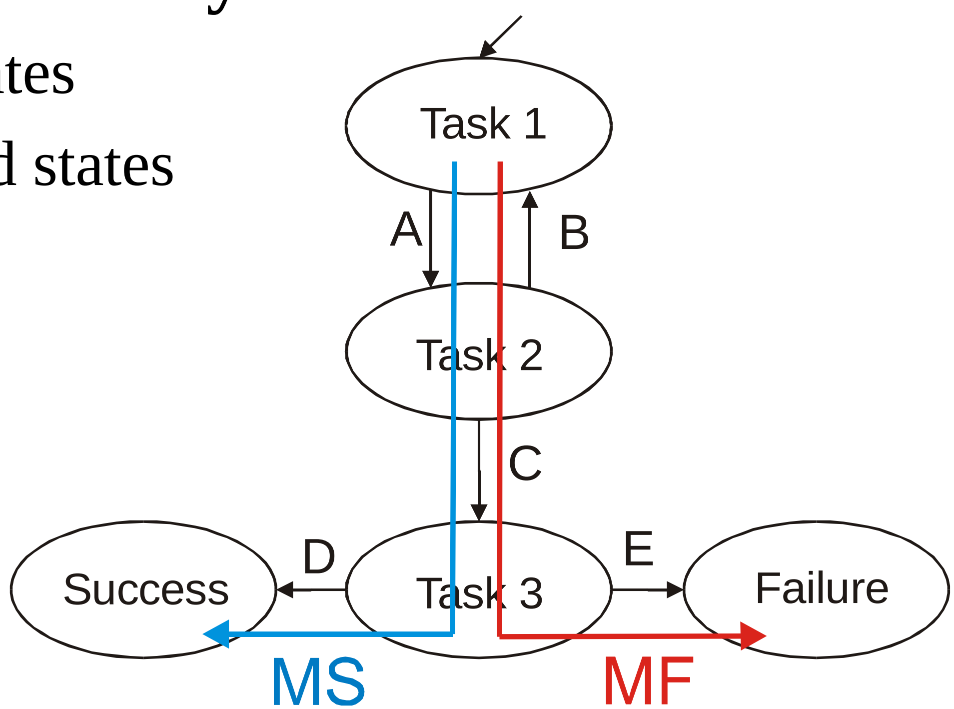
- **Why is dependability so hard to measure absolutely?**
  - FMEA, software FMEA, human error
    - Primarily realm-specific techniques

- **What attempts have been made to assess and improve relative dependability?**
  - Safety analysis: FTA, process improvements to reduce errors in requirements & human interfaces
    - Still realm specific - would like something more global

- **What other concepts are similar to the graph-based concepts that we shall introduce?**
  - Statecharts (usually per object), part-whole statecharts

Proposed Approach (1)

- **Dependability can be seen as a user successfully completing a series of tasks**
- **User's interaction with the system is modeled as a directed graph (*User Mission Graph*)**
  - Nodes are tasks, arcs are conditionally traversed
- **Dependability can be improved by:**
  - Adding paths toward good states
  - Also add paths away from bad states



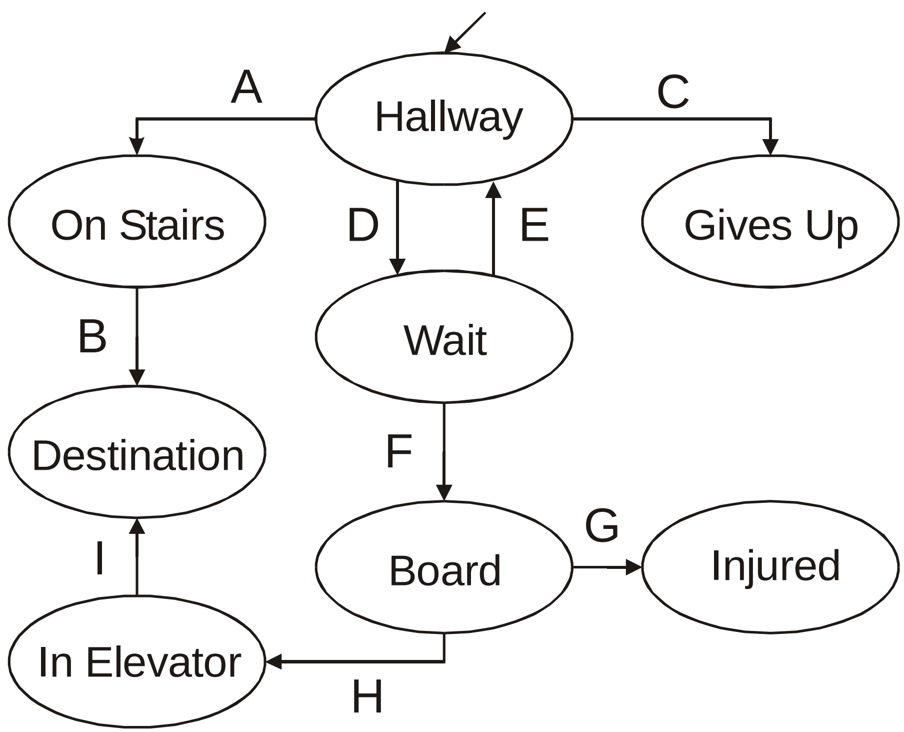
Proposed Approach (2)

- **Definitions:**
  - A *mission* is a complete path from a start node to an end node through the system
  - A *mission success* is a path that achieves a desired goal
  - A *mission failure* is a path that does not achieve a desired goal

- **Important qualities of User Mission Graph approach:**
  - Integrates user's contribution to dependability
  - Describes complete path through the system
  - Can be applied at design time (relative comparison)
  - Incorporates system view (HW, SW, HCI)

Embedded System Example

- Example user mission graph is a high level description of a user attempting to reach another floor in a building



Arc	Description
A	User times out (impatient)
B	User arrives at destination (walks)
C	User times out (frustrated)
D	User presses call button
E	User times out (excessive wait)
F	Doors open / lanterns activate
G	Doors close on user
H	User boarding time elapses
I	Doors close, elevator travels to destination

How do we apply our approach to the system?

- **Maximum dependability can be achieved by maximizing the probability of a mission success**
  - User can succeed even in light of partial failures

- **General idea: make it easy for user to achieve success**
  - Provide a rich set of possible ways to succeed
  - Multiple chances to divert from failure toward success

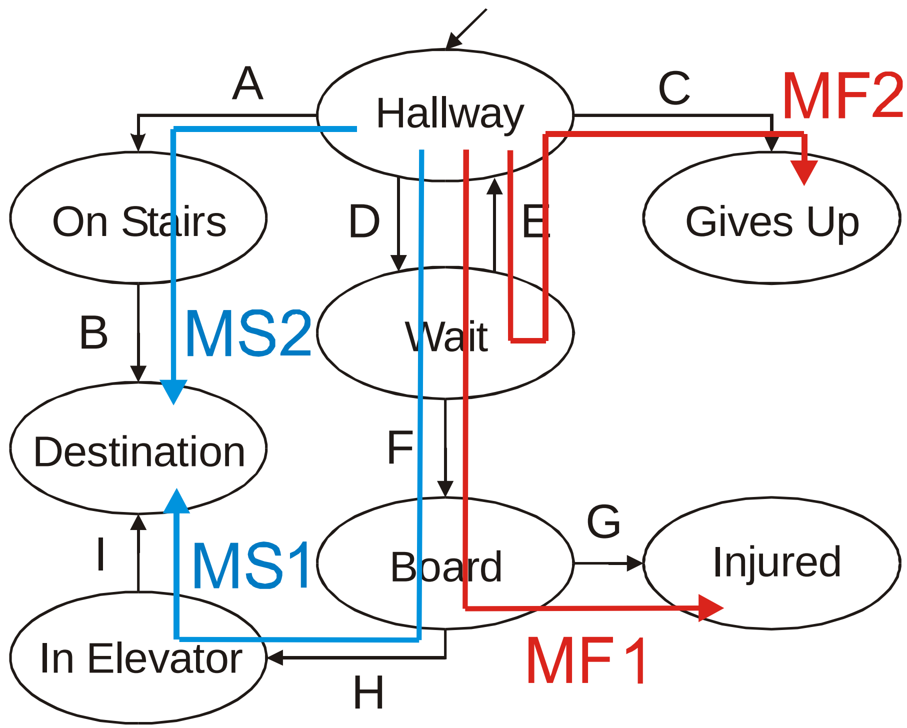
- **Simple heuristics can help us apply these strategies to the user mission graph**
  - More mission successes, fewer mission failures
  - More arcs toward good nodes, fewer toward bad
  - Increase path length to bad nodes, decrease to good

How can we analyze / transform the graph?

- **Three questions to ask while applying approach:**
  - Given start and end states, how many complete, distinct paths exist between them?
    - Determine ALL user missions in this step
  - Given a user state, what transitions exist to subsequent states?
    - Examine number and character of arcs out of each node in the graph
  - Given two mission paths, which portions are identical?
    - Focus attention on making failures more difficult to achieve without affecting the normal operation of the system

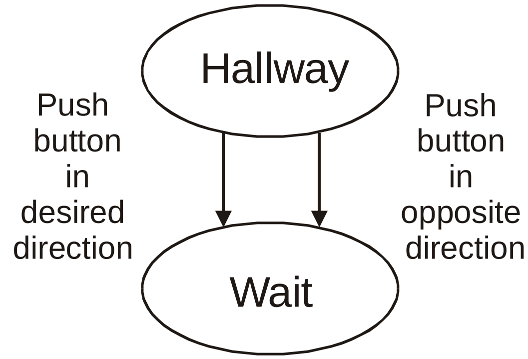
Mission Success / Failure Paths

- **Given start and end states, how many complete, distinct paths exist between them?**
  - Two mission successes, two mission failures
- **Focus:**
  - Many mission success scenarios suggest high dependability
  - Many mission failure scenarios suggest low dependability

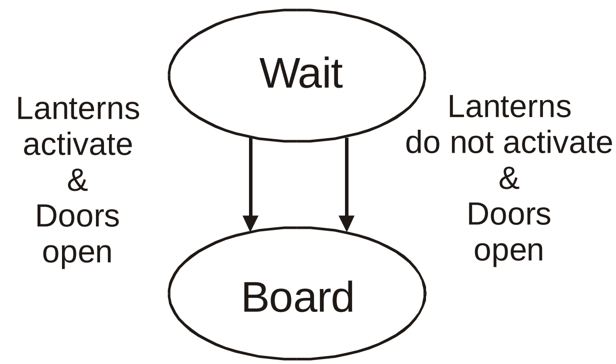


Hardware Redundancy & Human Interface

- **Given a user state, how many transitions exist to subsequent states?**
- **Focus:**
  - Additional arcs toward mission success increase dependability
- **EXAMPLE: Hardware redundancy**
  - Exploit heterogeneous redundancy to provide alternate paths
  - Move beyond brute force redundancy, traditional reliability measures



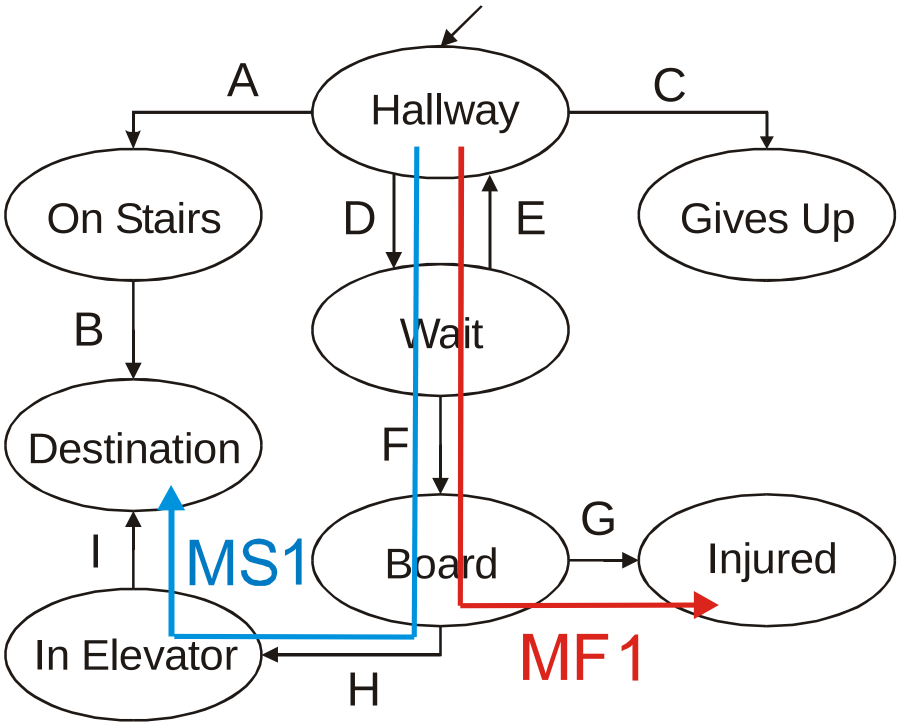
- **EXAMPLE: Human interface**
  - Elevator lantern enhances the system performance component of dependability
    - Malfunctioning lanterns (non-essential functionality) don’t put elevator out of service
    - Provides paths that correspond to user flexibility



Undesirable States (1)

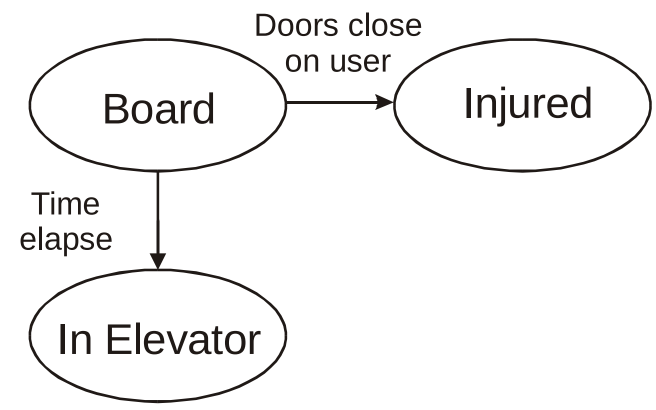
- **Given two mission paths, which portions are identical?**
  - Graph sub-sequences that are shared between success and failure paths are inherently risky

- **Safety vs. performance requirements are highlighted**
  - Example mission success and mission failure share a common path subset (***Hallway***, ***Wait***, ***Board***)
  - We DO NOT want to decrease performance during nominal operation
    - Difficulty in reaching ***Wait*** and ***Board*** should NOT be increased

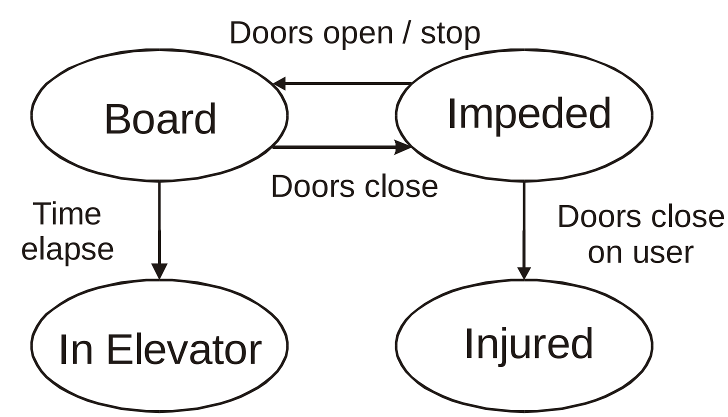


Undesirable States (2)

- **EXAMPLE: Transition from *Board* to *Injured* is a dependability tradeoff**
  - Safest system never allows passenger to be injured
    - If the doors are never closed, the passenger can never be injured
  - However, the safest system would have zero utility and thus zero dependability!
    - Note: in this example, the elevator cannot move if the doors are not closed
- **Focus:**
  - Longer paths towards failure increase dependability
- **Change the system to a useful point between complete safety and maximum performance**
  - Add intermediate state before failure



Original



Added state

Example Summary

- **Enumerate missions**
  - Give user more chances to succeed, fewer opportunities to fail
  - More mission successes, fewer mission failures increase overall system dependability
- **Change arcs**
  - Try to eliminate dependability 'bottlenecks'
  - More arcs toward success states give the user increased opportunities for success, and hence increase dependability
- **Change nodes**
  - Give the user a chance to work around partial system failures
  - Longer paths towards failure help increase dependability

Conclusion

- **Users are a part of improving dependability**
  - Systems can help users work around component failures
  - AND users can help systems work around component failures
- **Dependability can be enhanced by seeking to modify some formal properties of proposed graph constructs**
  - Number of paths / missions, number of arcs, number of nodes
- **Relative dependability assessment based on user mission graphs**
  - With some assumptions, can be useful at design time