RESISTING RELIABILITY DEGRADATION THROUGH PROACTIVE RECONFIGURATION

Summarized by Andeep Toor

By Deshan Cooray, Sam Malek, Roshanak Roshandel, and David Kilgore

Definitions

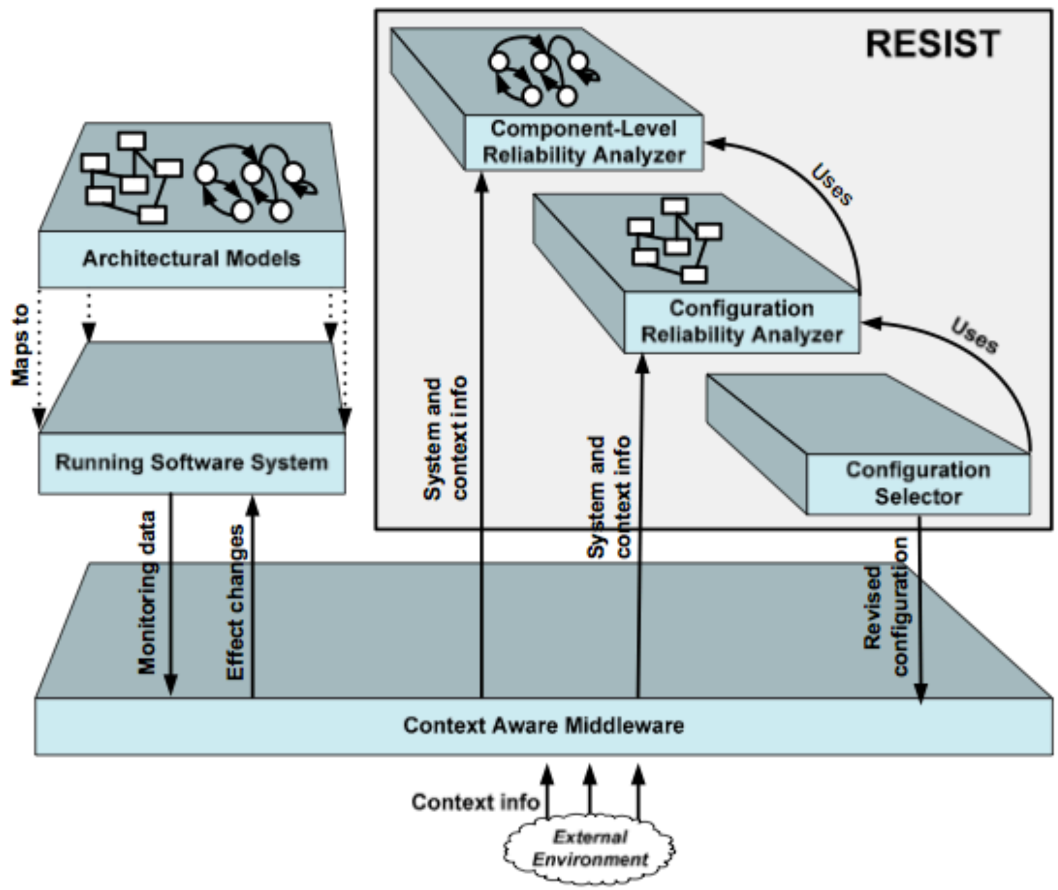
- *RESIST*
  - ***RE**silient **SI**tuated **S**of**T**ware system*
  - *"A framework for mission-critical systems"*
- *Situated Systems (SS)*
  - Embedded
  - Mobile
  - Pervasive
  - Ex. Mobile devices, robots
- Mission Critical
  - Ex. Emergency response, disaster recovery

Core Problem

- Mission critical systems require high reliability
  - Situated systems are inherently unreliable
  - External factors play a huge role in this
- The best configuration for a system is known only at runtime
  - Need to update configuration to improve reliability
- How do we design such a system?

What does RESIST do?

- Self-healing / Self-optimizing



What assumptions does RESIST make?

- Errors are assumed to be between components
  - Errors internal to the component are not handled by this error model
- Configurations may have replicas of components
- Replicas of different components fail independently

How is RESIST different?

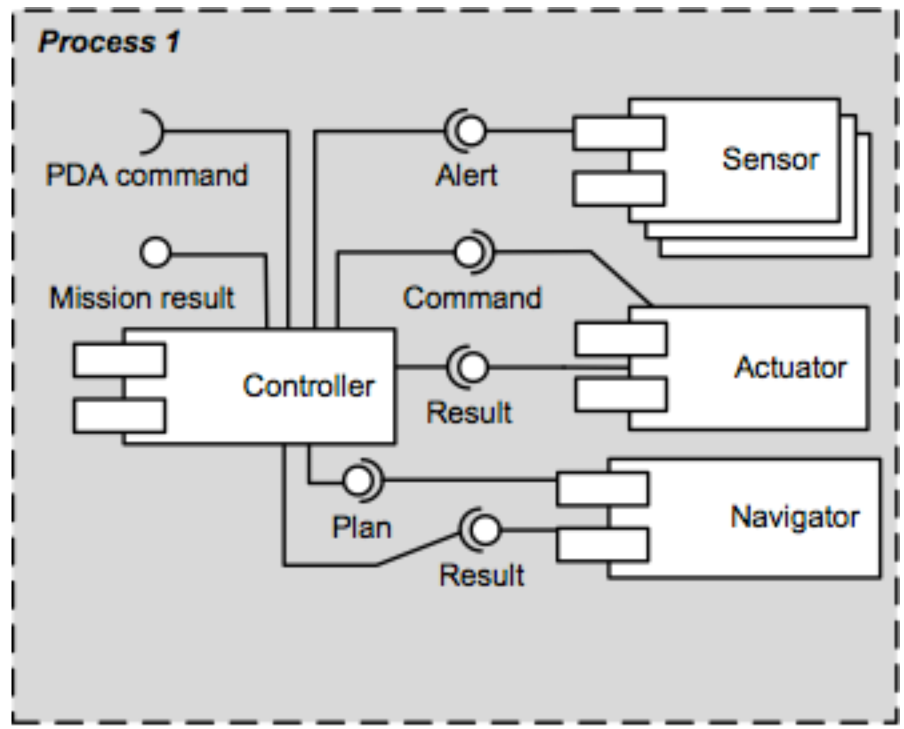
- Optimizes proactively
  - Uses predictive models to optimize ahead
  - Focuses on where system is expected to be
    - Different from other systems that focus on current state
    - Note: Can only focus on near-future
- Considers external factors (Context)
- Other related work not appropriate
  - Expects apriori knowledge of reliability
  - Do not consider context

How does RESIST work?

- Determine optimal configuration of components for SS
- Optimal = most reliable
  - Calculate individual component reliabilities
  - Calculate total system reliability
    - This is based on individual component reliabilities
  - Consider architectural factors
    - Redundant components
    - Assignment of components to processes

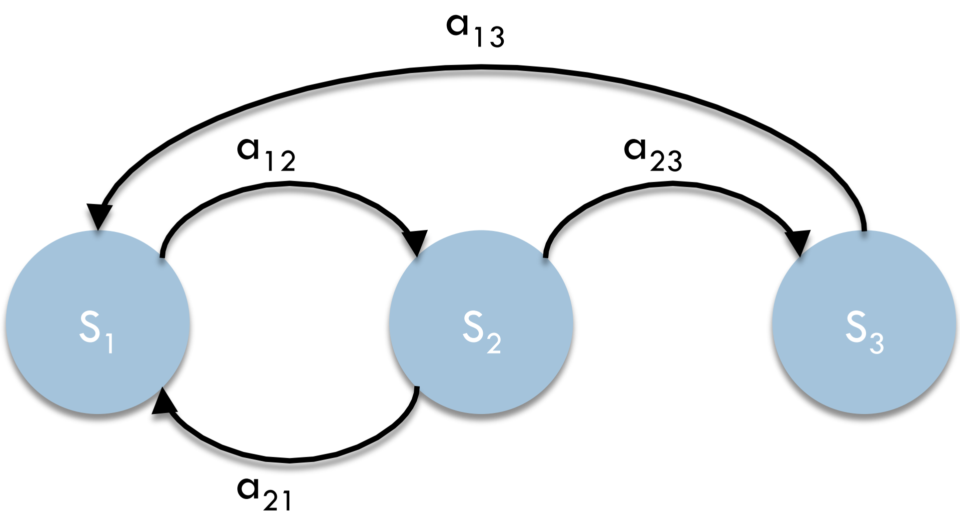
Scenario

- Emergency response
  - Firefighters
  - Central dispatcher
- Robots
  - Components
    - Sensors
    - Actuators
    - Controllers



Calculating Component Reliability

- Uses Hidden Markov Models (HMMs)
  - Normal Markov Model

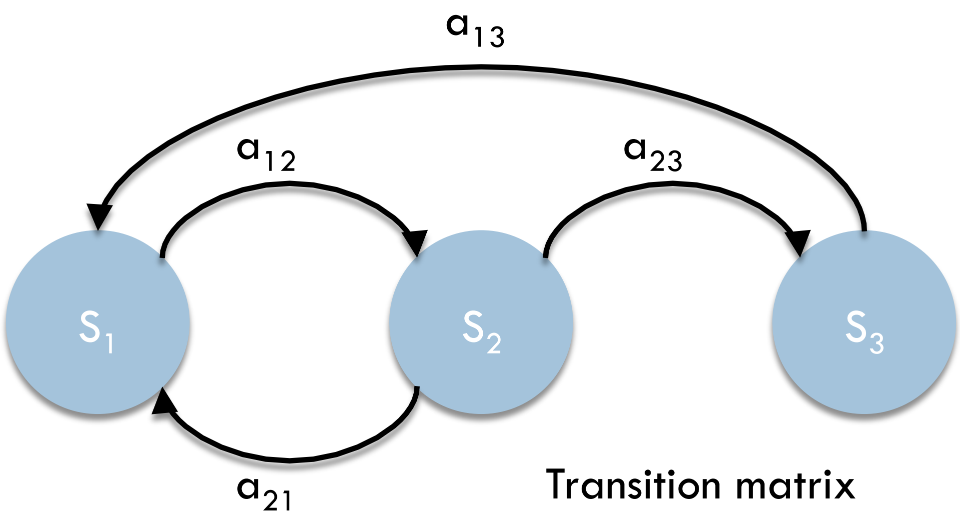


Calculating Component Reliability

- Normal Markov Model
  - Can predict next state based purely on current state

Set of states

$S = \{S_1, S_2, ..., S_N\}$



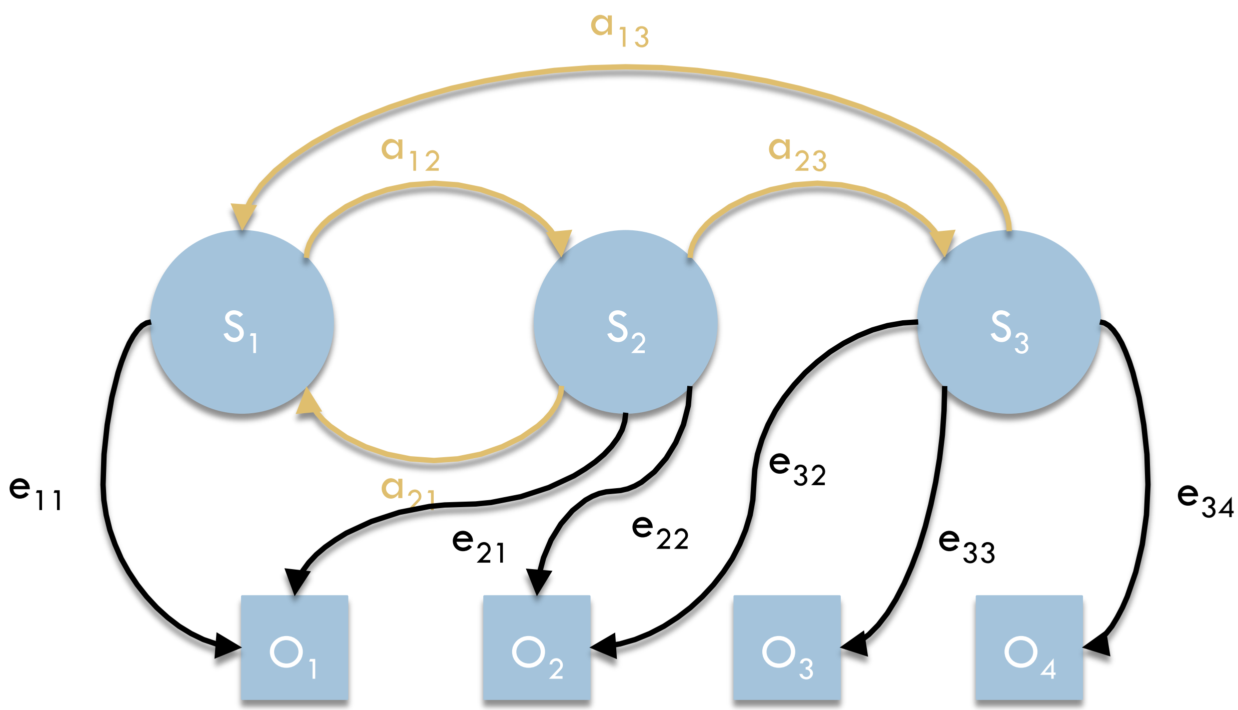
Transition matrix

$A = \{a_{ij}\}$

This gives the probability of transitioning from one state to another

Calculating Component Reliability

- Hidden Markov Models (HMMs)
  - HMMs extend this idea by adding hidden states



Calculating Component Reliability

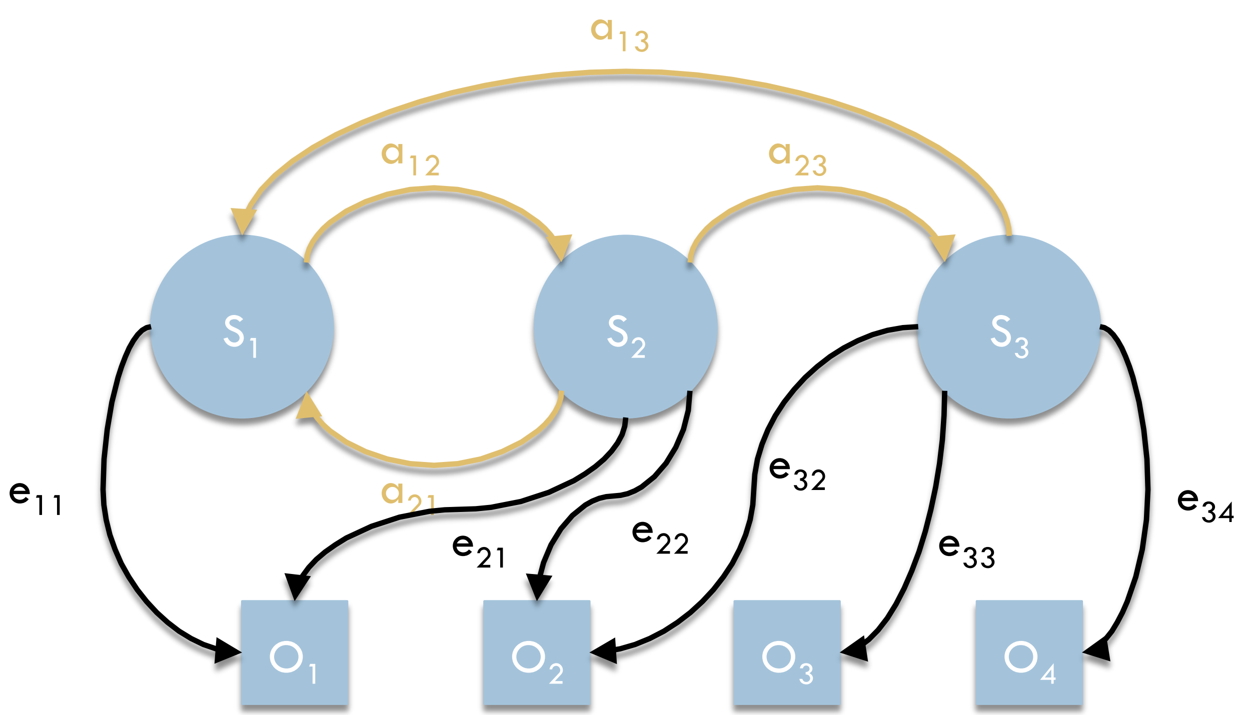
Set of observations

$O = \{O_1, O_2, \ldots, O_N\}$

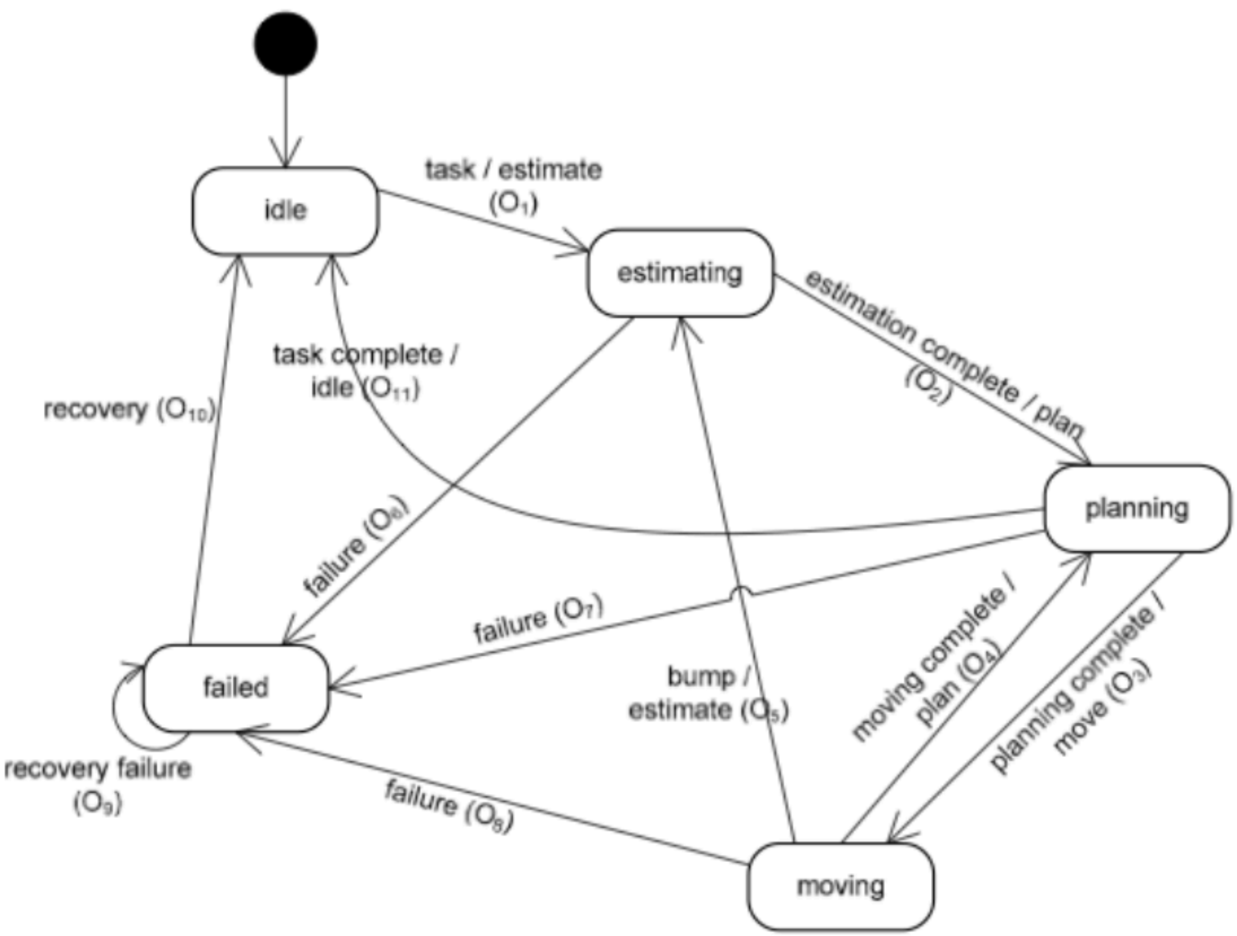
Observation probability matrix

$E = \{e_{ik}\}$

This represents the probability of observing an event in a particular state

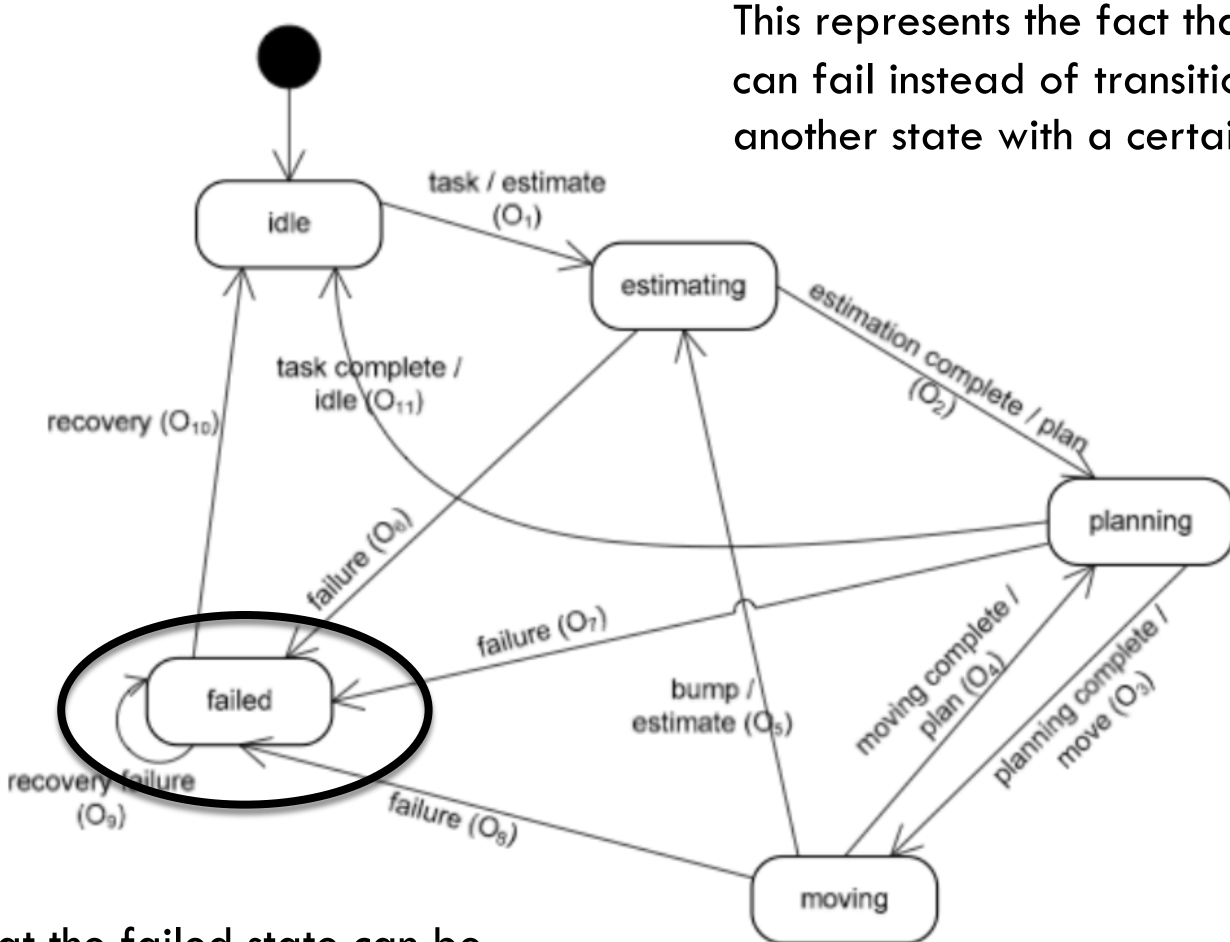


Real State Transitions



Real State Transitions

This represents the fact that a system can fail instead of transitioning to another state with a certain probability



Note that the failed state can be reached from most other states

Training the HMM

- States are known
  - Ex. Monitoring, moving

- Need to determine transition probability matrix
  - Can learn this from monitoring data
    - This gives us observations

  - Train using sample data
    - Baum-Welch algorithm
      - Method for finding the hidden parameters in an HMM
      - Uses expectation-maximization

Predictive Calculations

- Calculating reliability at runtime before failure
  - Involves the use of "context"
    - These are events or processes outside of the system that affect it
    - Must be included in calculations for situated systems
    - Introduce a new set of parameters:

Set of contextual parameters

$$C = \{C_1, C_2, \ldots, C_N\}$$

Using Context in Reliability Calculations

- $a'_{kj} = u(a_{kj}, \Delta C_n)$
  - a_{kj} – transition probability
  - u – a function that based on context
    - Encapsulates the effect that C_n has on a_{kj}

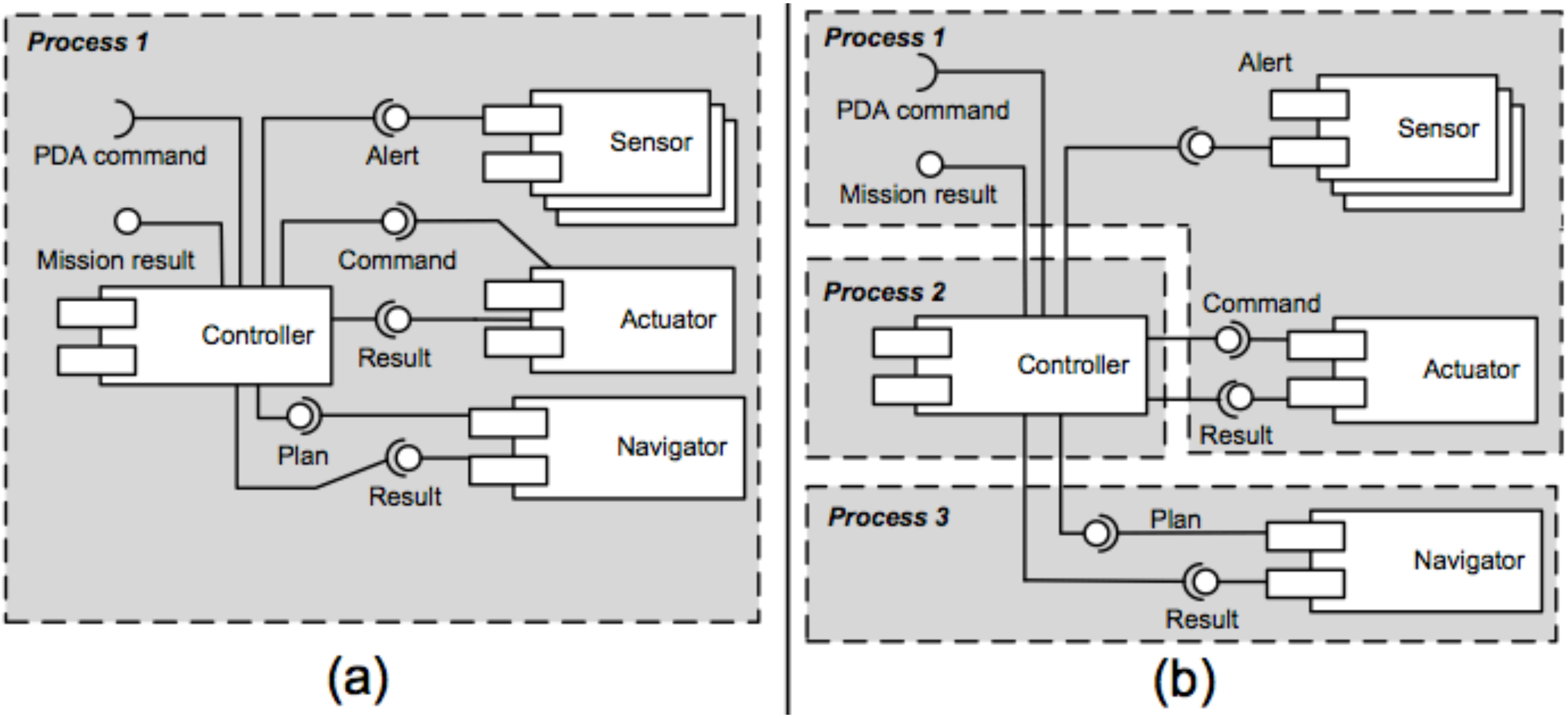
Calculating Total System Reliability

- Based on individual component reliability

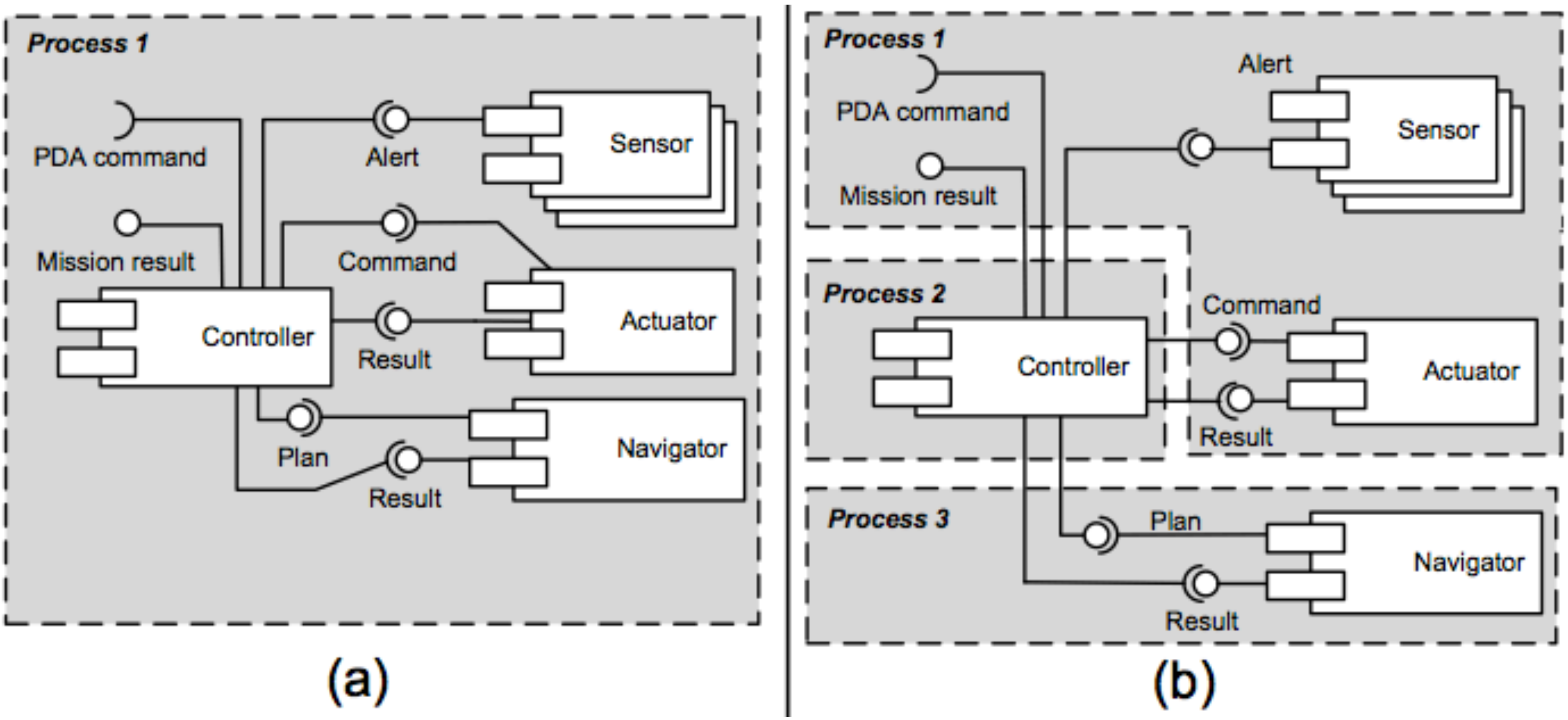
$$R = (-1)^{k+1} R_k \frac{|E|}{|I-M|}$$

  - k = Number of states
  - R_k = Reliability of exit state
  - M = Matrix of size k x k
  - |I-M| = Determinant of M
  - |E| = Determinant of everything but the first column and row of |I-M|

Considering Architectural Factors



Considering Architectural Factors



More efficient architecture

More reliable architecture

Finding Optimal Configuration

- Reliability is the goal
  - In practice, other factors may influence calculation

$$C^* = argmax_{(C)} \sum_{\forall q \in QualityObjectives} U_q(C)$$

$$Subject\ to\ \ R(C) \geq \delta, \delta \in \mathbb{R}, 0 < \delta \leq 1$$

  - U_q = Utility function
  - Can take on any format

Finding Optimal Configuration

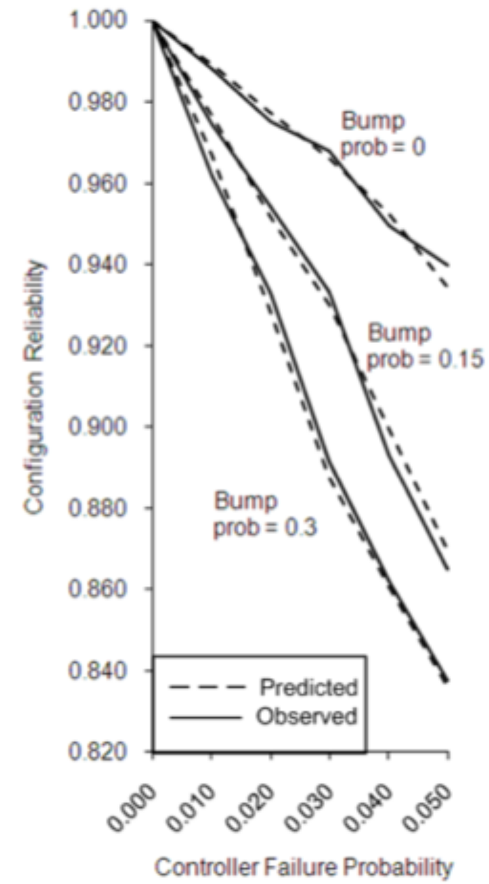
- Configurations have constraints
  - Must be assigned to at least one process
  - Can have a bounded number of replicas
  - Cannot share a process and have a replica
    - Components and replicas should be on separate processes

Experimental Results

- Robot example from earlier
- Context - probability of hitting an obstacle
  - Bump probability (BP)
- Controller failure is examined with respect to different BP
  - This is because the transition from one state to another can fail with a certain probability

Experimental Results

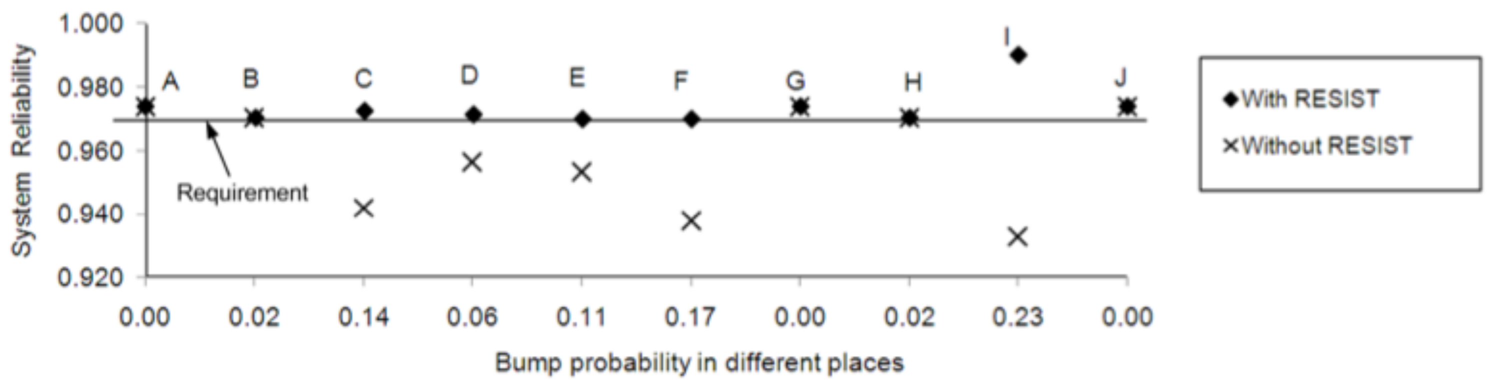
- Observed and predicted reliability
  - Shows accuracy of predictive model
- Reliability degrades with context
  - Increased BP = lower reliability



Experimental Results

- Real robotic results
- RESIST sees a increase in BP
  - This is predicated to result in a drop in reliability
  - Before this degradation in reliability, RESIST

    "adapts the system to maintain its reliability above 97%. As a result, the *Navigator is replicated and the Controller is redeployed to a separate process.*"



Conclusion

- Overall, the paper covers a lot of ground
- Offers an interesting, predictive approach
- Questions
  - What other machine learning techniques can be used to aid prediction?
  - Does the system's accuracy improve with more data / examples?

References

1. Deshan Cooray , Sam Malek , Roshanak Roshandel , David Kilgore, RESISTing reliability degradation through proactive reconfiguration, Proceedings of the IEEE/ACM international conference on Automated software engineering, September 20-24, 2010, Antwerp, Belgium

- http://en.wikipedia.org/wiki/Baum-Welch_algorithm