A Model-Based System Supporting Automatic Self-Regeneration of Critical Software

Paul Robertson & Brian Williams

Model-Based and Embedded Robotic Systems

http://mers.mit.edu

MIT

Computer Science and Artificial Intelligence Laboratory

What we are trying to do

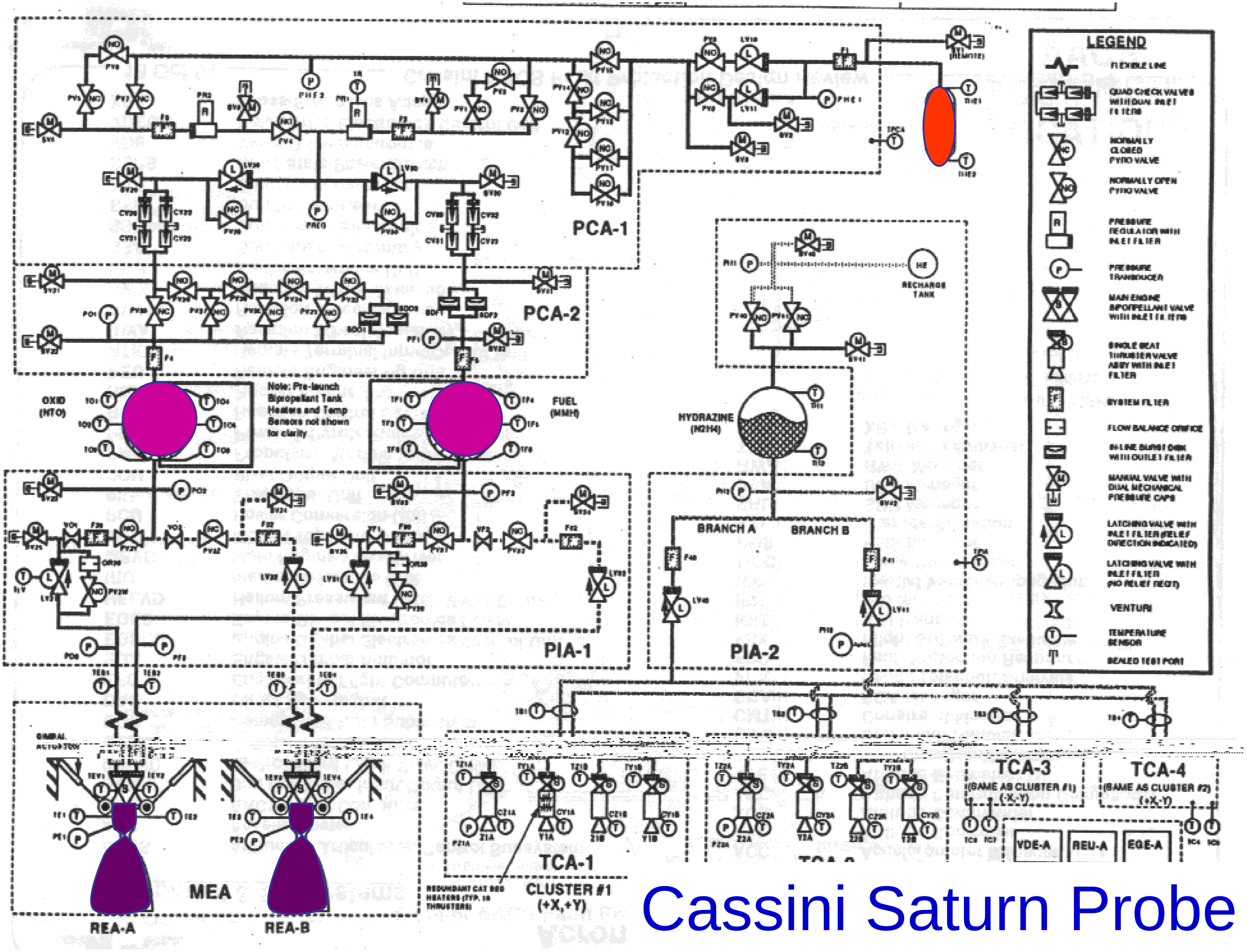
- Why software fails:
  - – Software assumptions about the environment become invalid because of changes in the environment.
  - – Software is attacked by a hostile agent.
  - – Software changes introduce incompatibilities.
- What can be done when software fails:
  - – Recognize that a failure has occurred.
  - – Diagnose what has failed – and why.
  - – Find an alternative way of achieving the intended behavior.

Runtime Models

Self repairing explorer: Deep Space 1

Flight Experiment, May 1999.

courtesy ARC & JPL


PCA-1
PCA-2
OXID (NTO)
FUEL (MMH)
Note: Pre-launch Bipropellant Tank Heaters and Temp Sensors not shown for clarity
HYDRAZINE (N2H4)
HE RECHARGE TANK
BRANCH A
BRANCH B
PIA-1
PIA-2
GIMBAL ACTUATOR
MEA
REA-A
REA-B
TCA-1
CLUSTER #1 (+X,+Y)
REDUNDANT CAT BED HEATERS (TYP. 16 THRUSTERS)
TCA-3 (SAME AS CLUSTER #1) (-X,-Y)
TCA-4 (SAME AS CLUSTER #2) (+X,-Y)
VDE-A
REU-A
EGE-A
LEGEND
FLEXIBLE LINE
QUAD CHECK VALVES WITH DUAL INLET FILTERS
NORMALLY CLOSED PYRO VALVE
NORMALLY OPEN PYRO VALVE
PRESSURE REGULATOR WITH INLET FILTER
PRESSURE TRANSDUCER
MAIN ENGINE BIPROPELLANT VALVE WITH INLET FILTERS
SINGLE SEAT THRUSTER VALVE ASSY WITH INLET FILTER
SYSTEM FILTER
FLOW BALANCE ORIFICE
IN-LINE BURST DISK WITH OUTLET FILTER
MANUAL VALVE WITH DUAL MECHANICAL PRESSURE CAPS
LATCHING VALVE WITH INLET FILTER (RELIEF DIRECTION INDICATED)
LATCHING VALVE WITH INLET FILTER (NO RELIEF REQT)
VENTURI
TEMPERATURE SENSOR
SEALED TEST PORT

Cassini Saturn Probe

Project Status

Funding: DARPA (SRS), NASA (Ames)

Current State: Prototype System Operational

Project Premise:

- Extend proven approach to hardware diagnosis and repair as used in DS-1 to critical software.

Principle Ideas:

- Model-Based Language Approach
- Redundant Methods
- Method Deprecation
- Model-Predictive Dispatch
- Hierarchical Models
- Adjustable Autonomy

Overview

Technical Objective:

When software fails because (a) environment changes (b) software incompatibility (c) hostile attack, (1) recognize that a failure has occurred, (2) diagnose what has failed and why, and (3) find an alternative way of achieving the intended behavior.

Technical approach:

By extending RMPL to support software failure, we can extend robustness in the face of hardware failures to robustness in the face of software failures. This involves:

(1) **Detection**

(2) **Diagnosis**

(3) **Reconfiguration**

(4) **Utility Maximization**.

RMPL Models of:
Software Components,
Component Hierarchy & Interconnectivity,
and Correct Behavior.

Expected Benefits

- Software systems that can operate autonomously to achieve goals in complex and changing environments.
  - Modeling environment
- Software that detects and works around “bugs” resulting from incompatible software changes.
  - Modeling software components
- Software that detects and recovers from software attacks.
  - Modeling attack scenarios
- Software that automatically improves as better software components and models are added.

What can go wrong?

1. Hardware: A problem with robot hardware.
2. Software: A problem with the environment.
    1. A mismatch between a chosen algorithm and the environment such as there not being enough light to support processing of a color image.
    2. An unexpected imaging problem such as an obstruction to the visual field (caused by a large obscuring rock).

Solution to 2.1

Reconfigure the software structure:

1. Redundant Methods
2. Mode Estimation
3. Mode Reconfiguration

Solution to 2.2

Switch to a contingent plan:

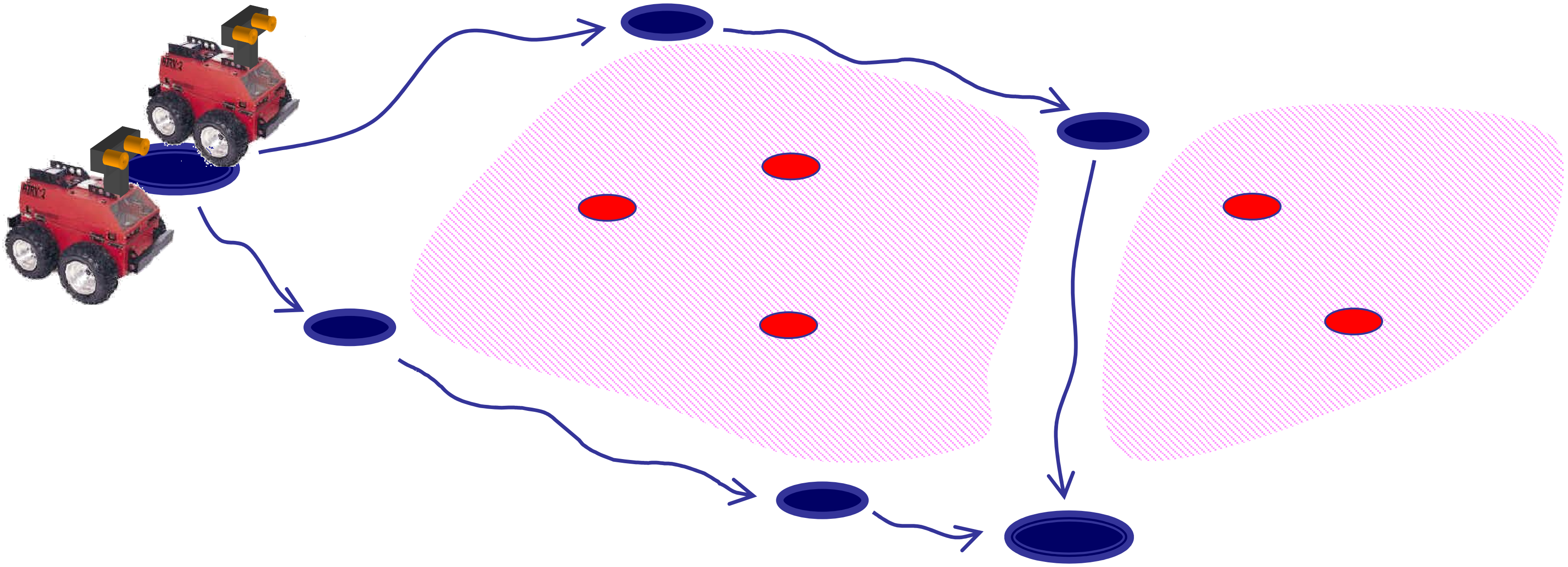
1. Exception
2. Model Predictive Dispatch
3. Replanning

Test Bed Platform

Involves:

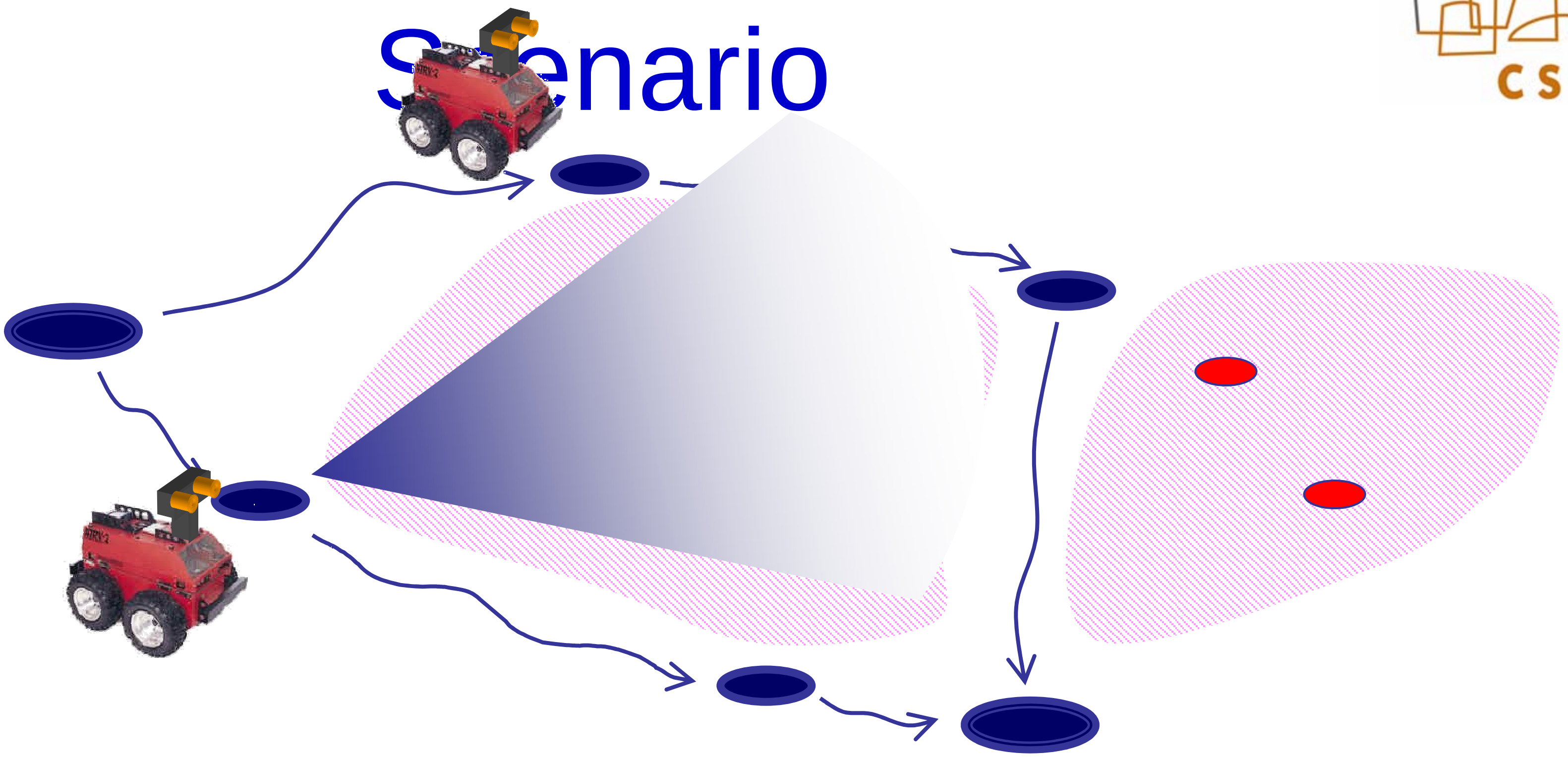
- Cooperative use of multiple robots.
- Timing critical software.
- Reconfiguration of Software Components.
  - Multiple Redundant Methods
- Continuous Replanning
  - Multiple Redundant Methods

Science Target Search Scenario

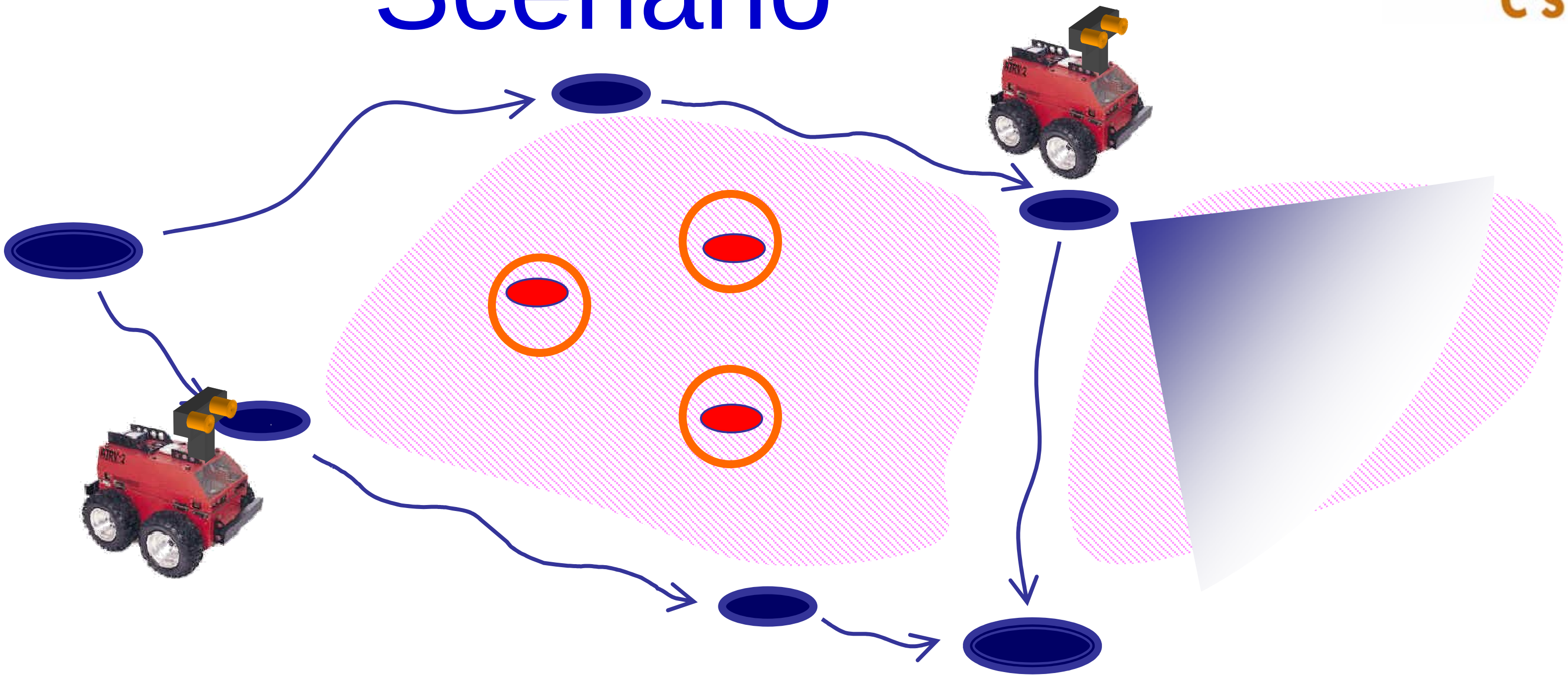


- Cooperatively search for targets in the predefined regions
- Search from predefined viewpoints
- Search for the targets using stereo cameras and various visualization algorithms

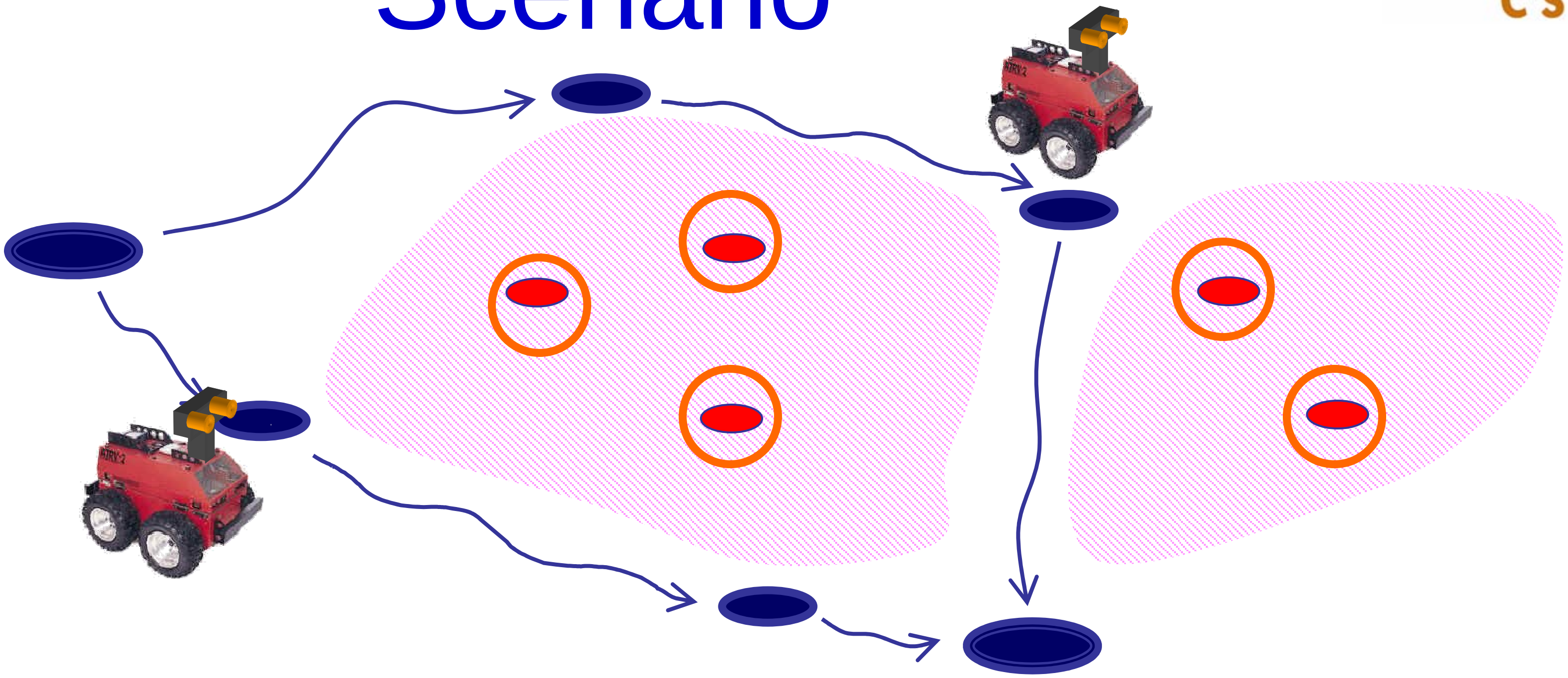
Science Target Search Scenario



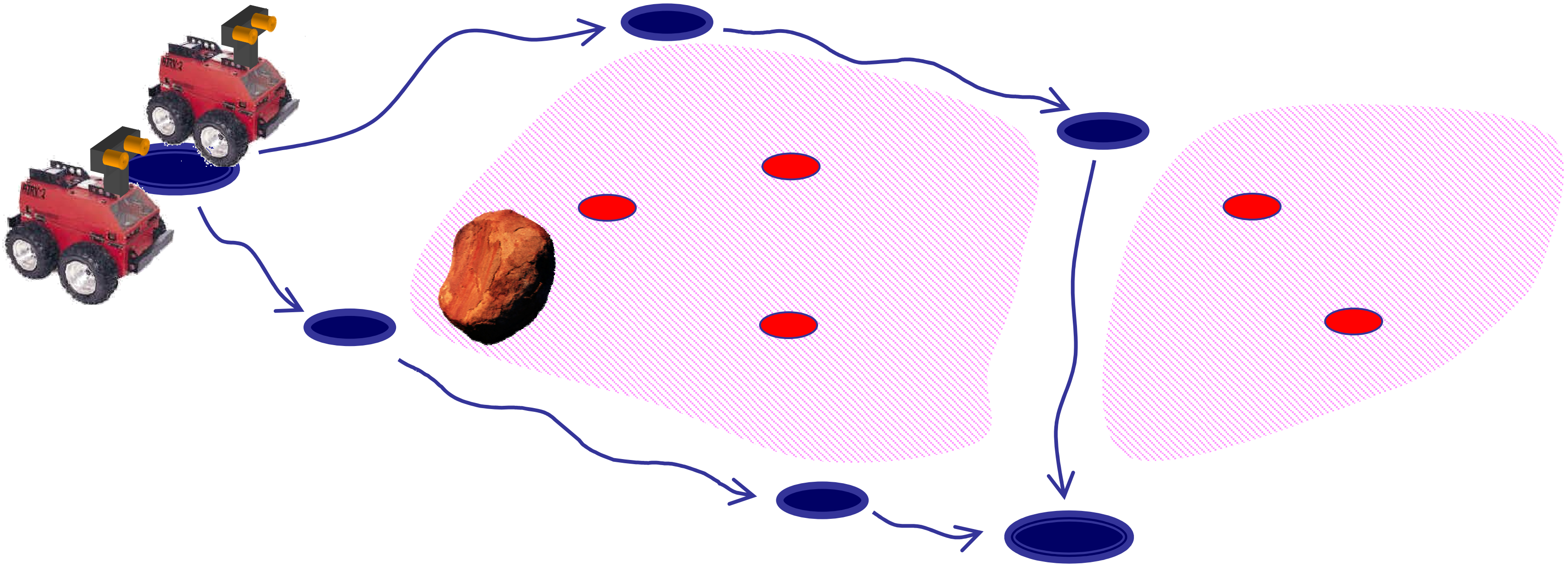
Science Target Search Scenario



Science Target Search Scenario

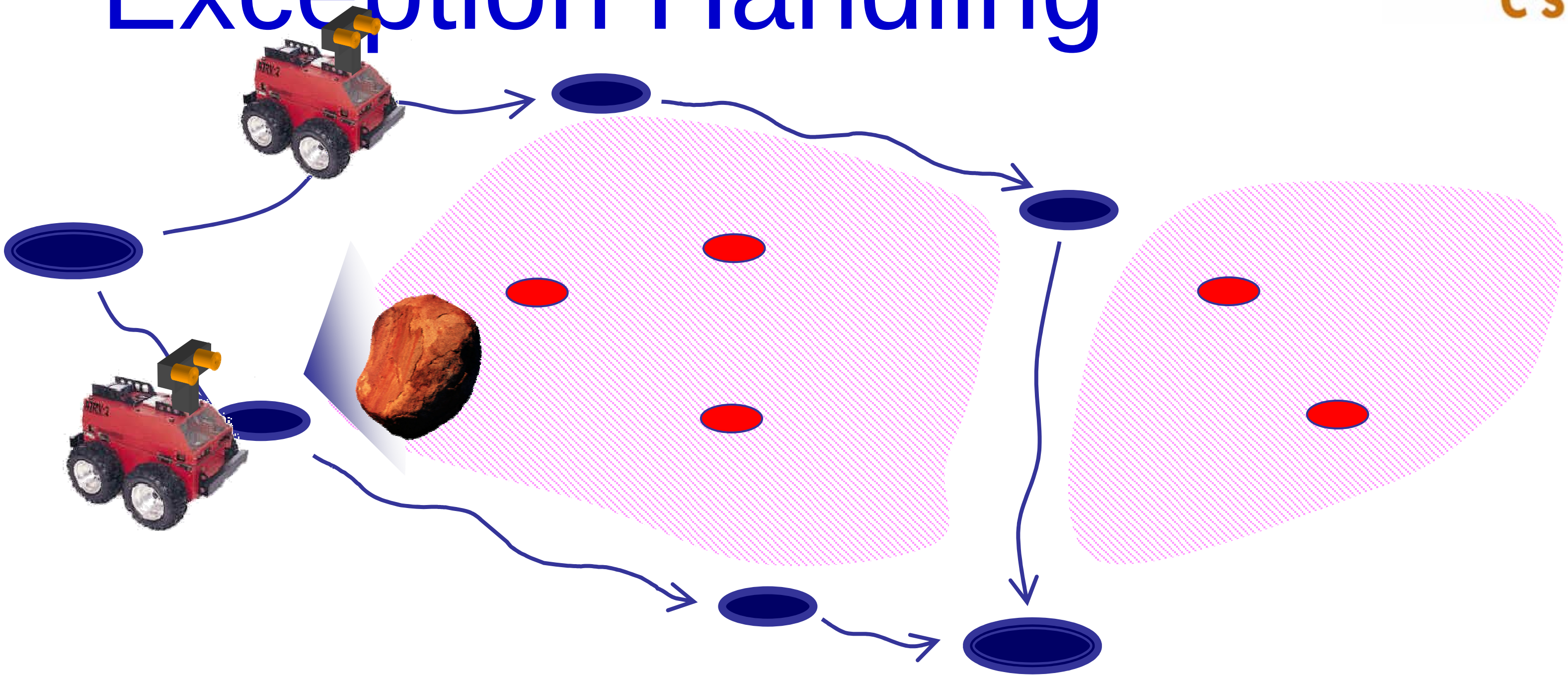


Method Regeneration: Exception Handling

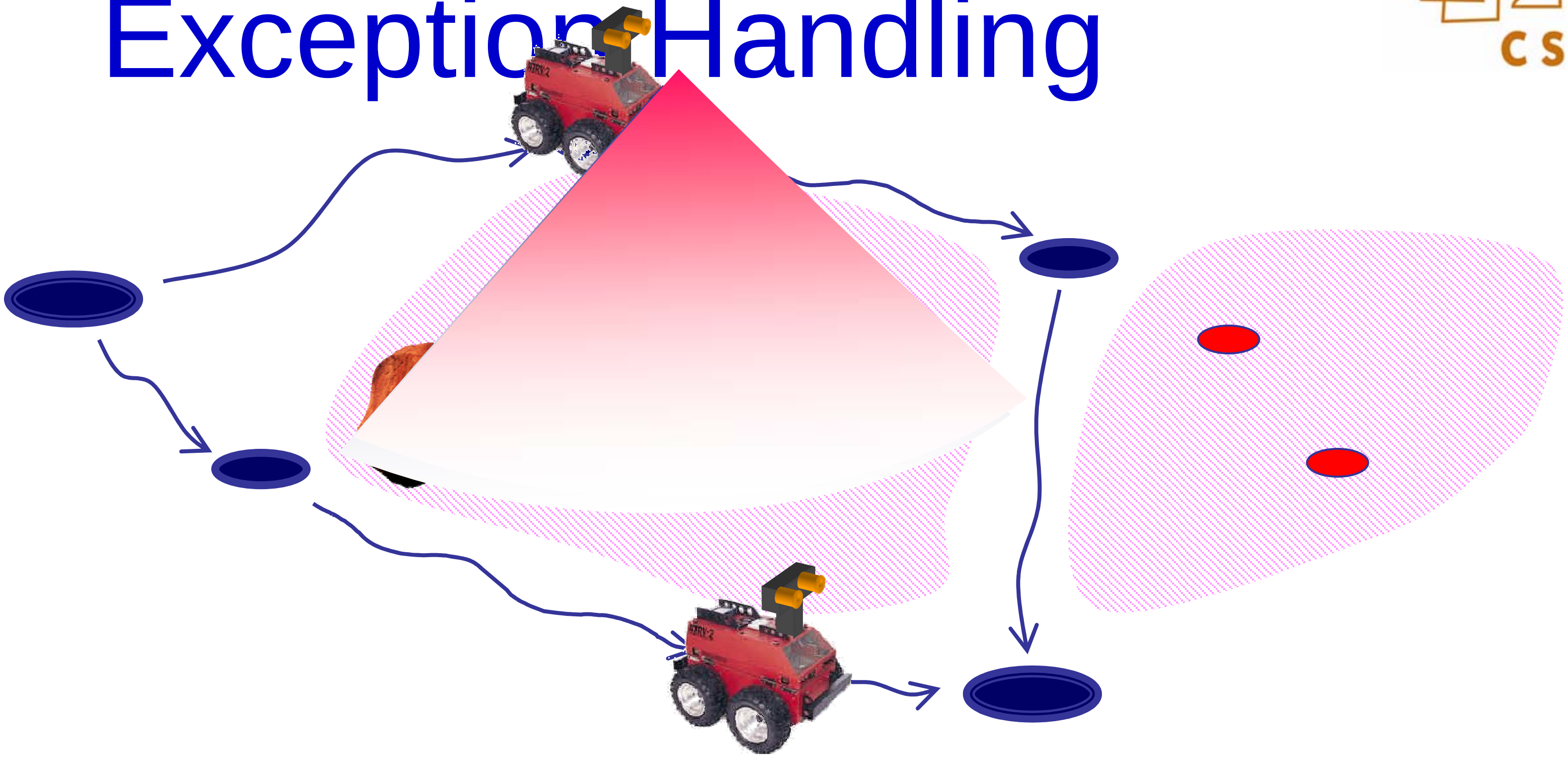


- A rock blocks the view
  - Recover by taking the image from a different perspective (i.e. change the strategy)
- The shadow cast by the rock fails the imaging code from identifying the objects in view
  - Reconfigure the imaging algorithm to work under these conditions

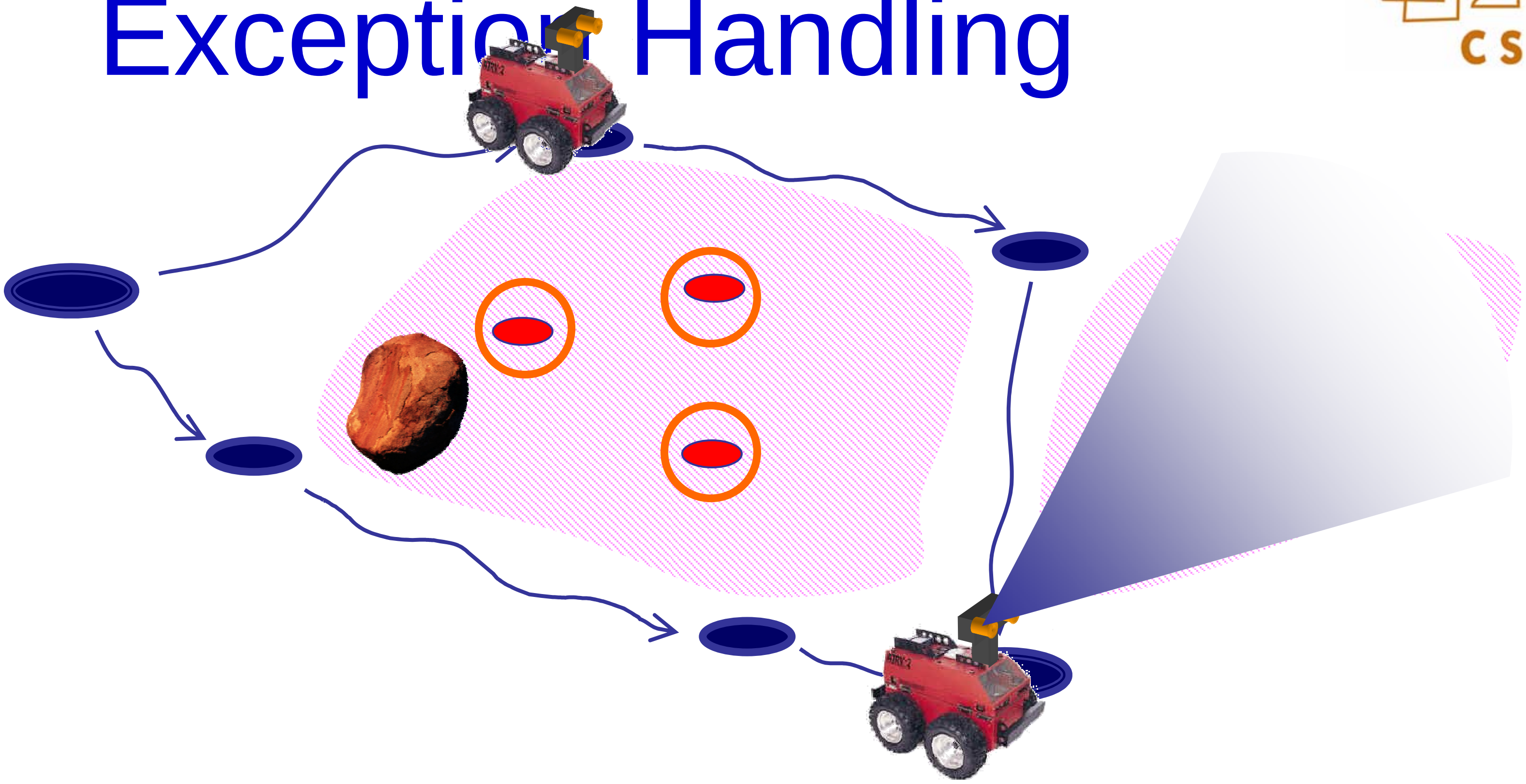
Method Regeneration: Exception Handling



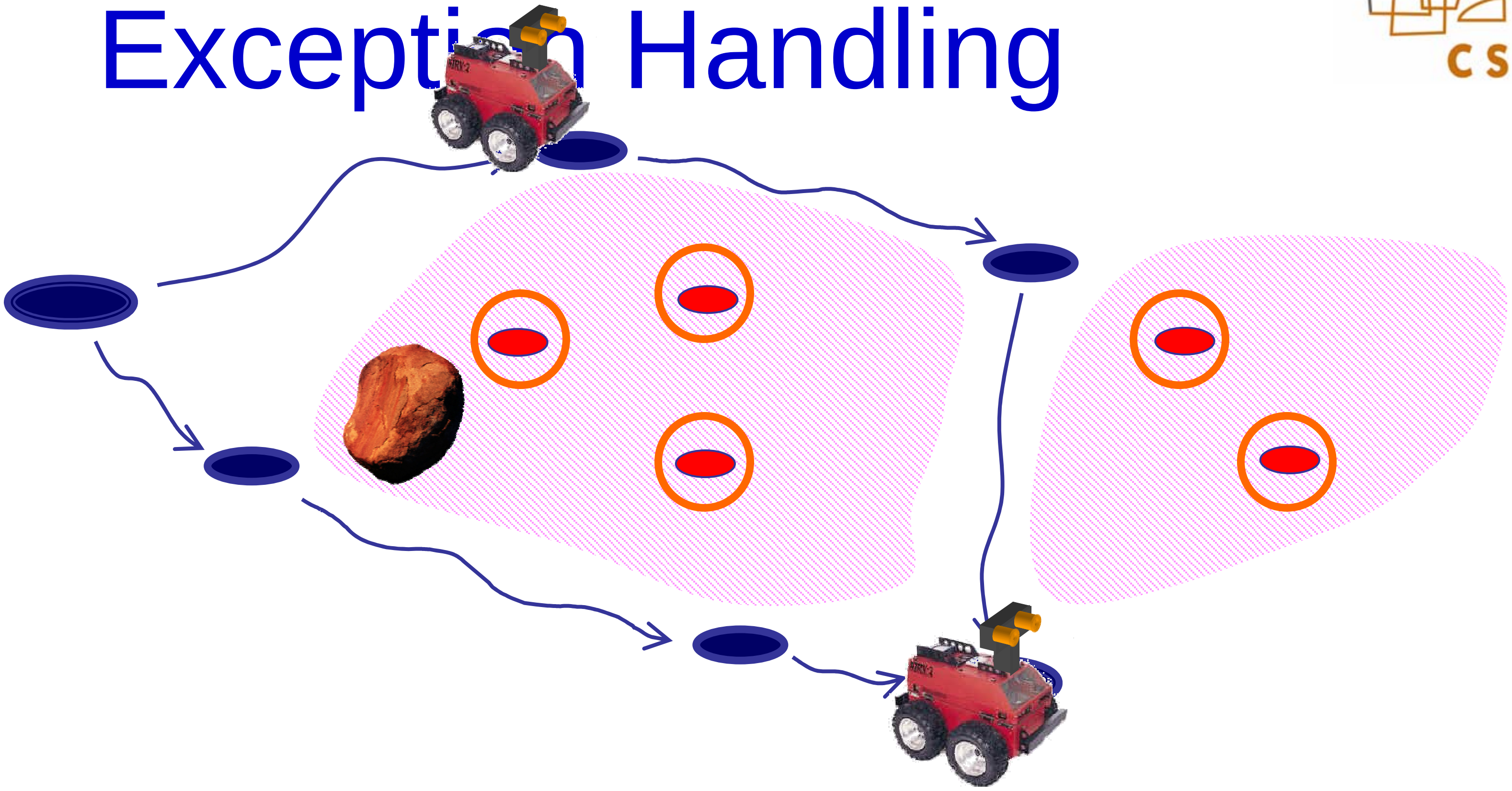
Method Regeneration: Exception Handling



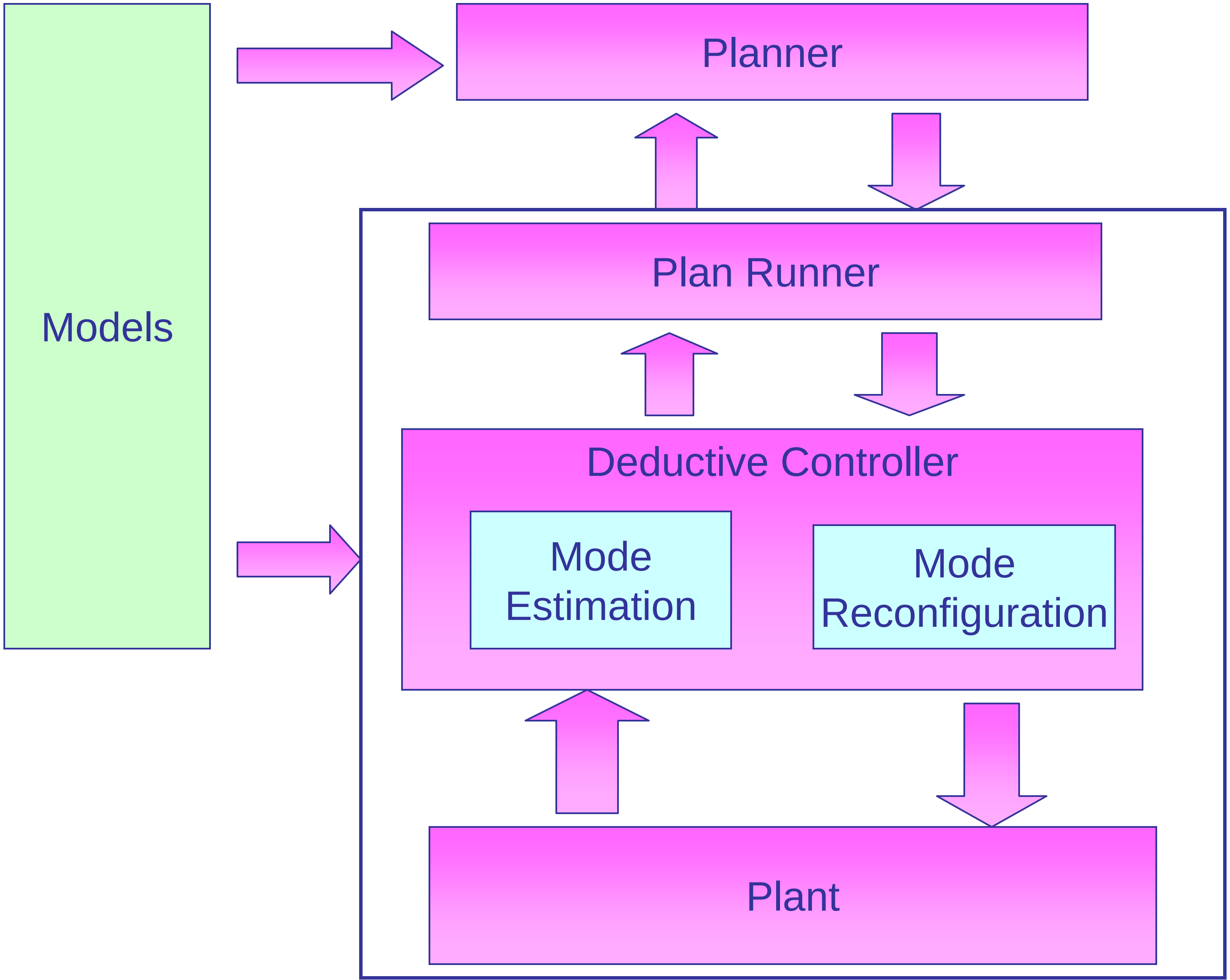
Method Regeneration: Exception Handling



Method Regeneration: Exception Handling

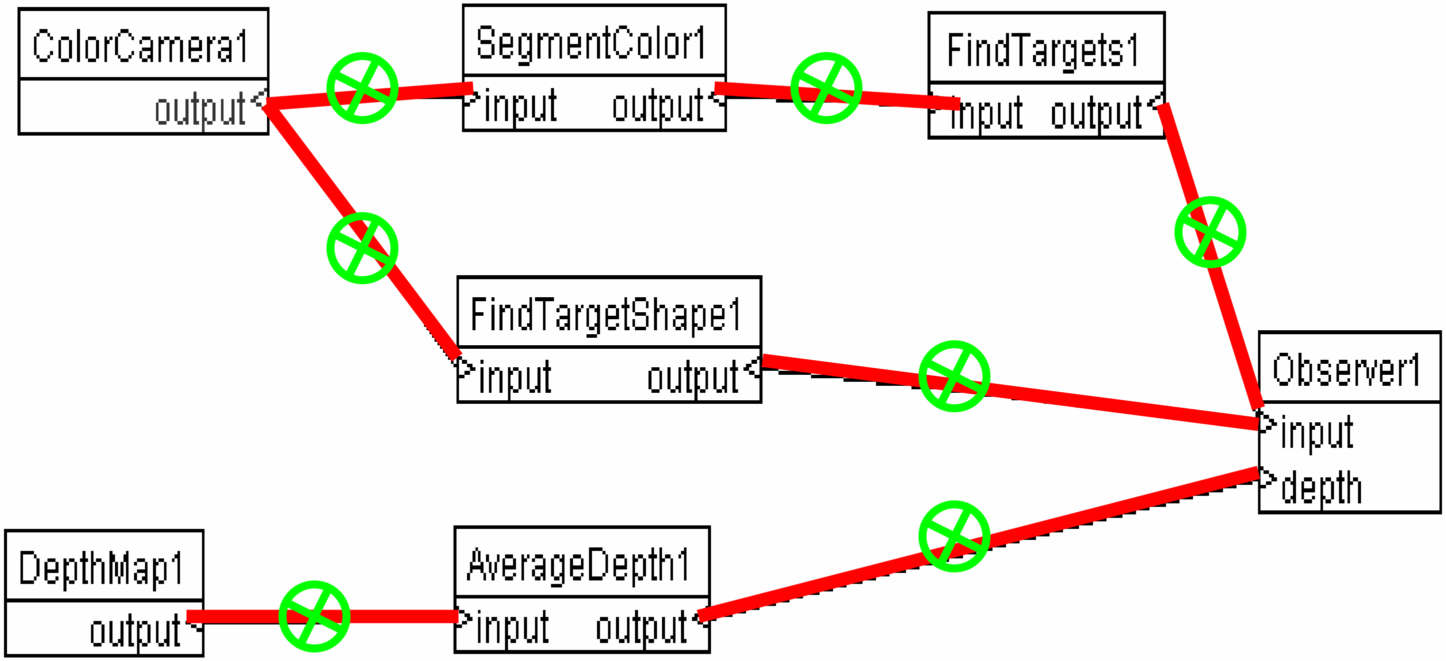


Overall Architecture

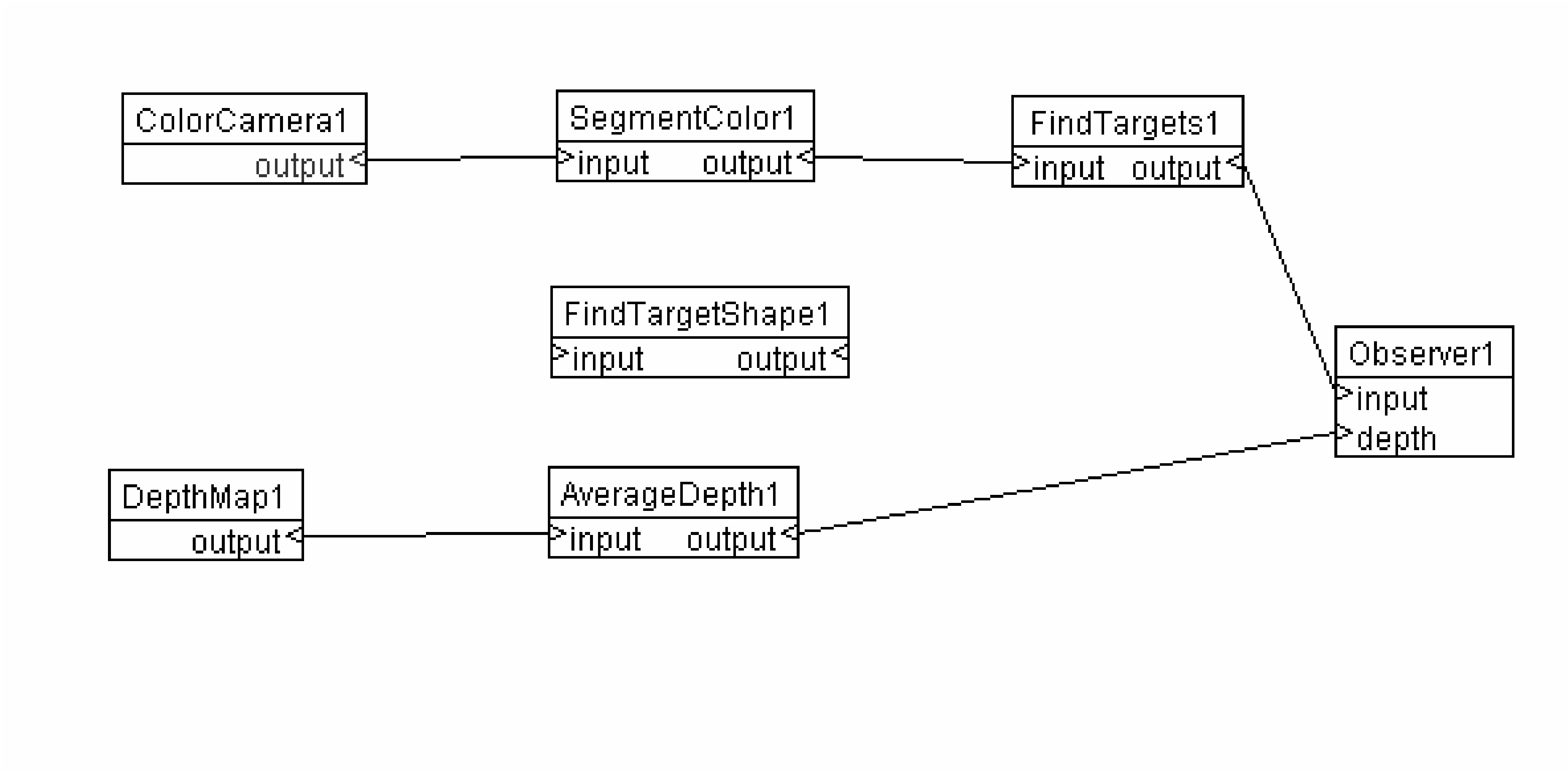


Reconfigurable Vision for Robust Rover Mapping

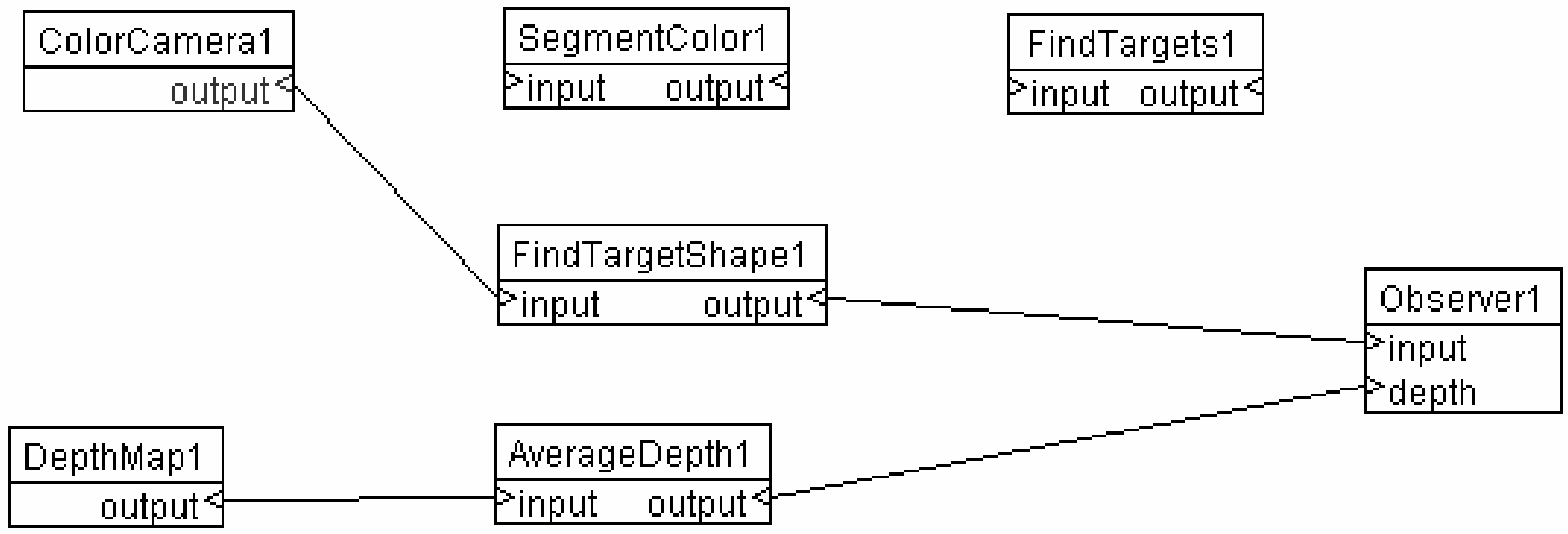
Reconfigurable Vision Plant Model



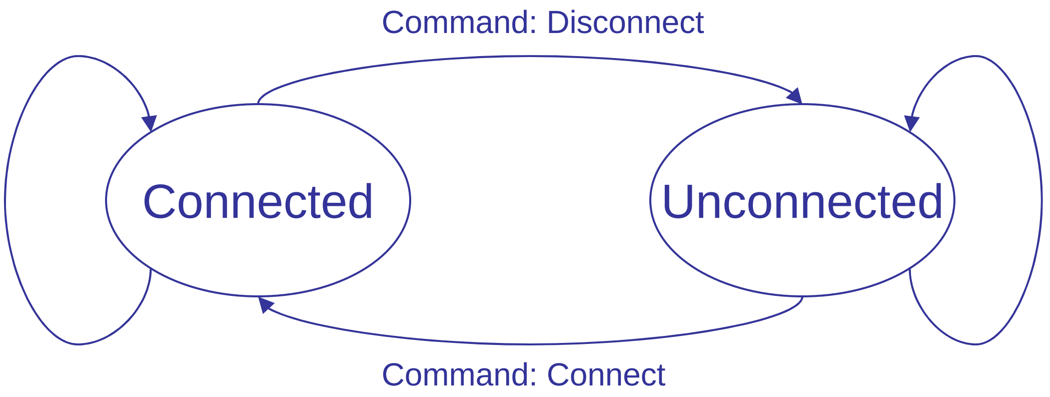
Nominal Configuration



Contingent Configuration



Connection



Command: Disconnect

Connected

Unconnected

Command: Connect

Inputs: x

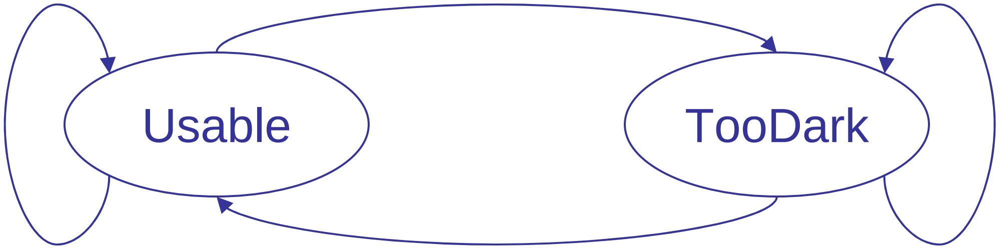
Outputs: x

Models simplified for clarity in this and following slides

```
class Connection ()
{
    RawImage image_in;
    SegmentedImage image_out;

    mode Connected (…) {
      primitive method disconnect () => Unconnected; }
    mode Unconnected (…) {
      primitive method connect () => Connected; }
   failure mode Failed (…) { … };
}
```

SegmentColor


Usable

TooDark

Inputs: RawImage

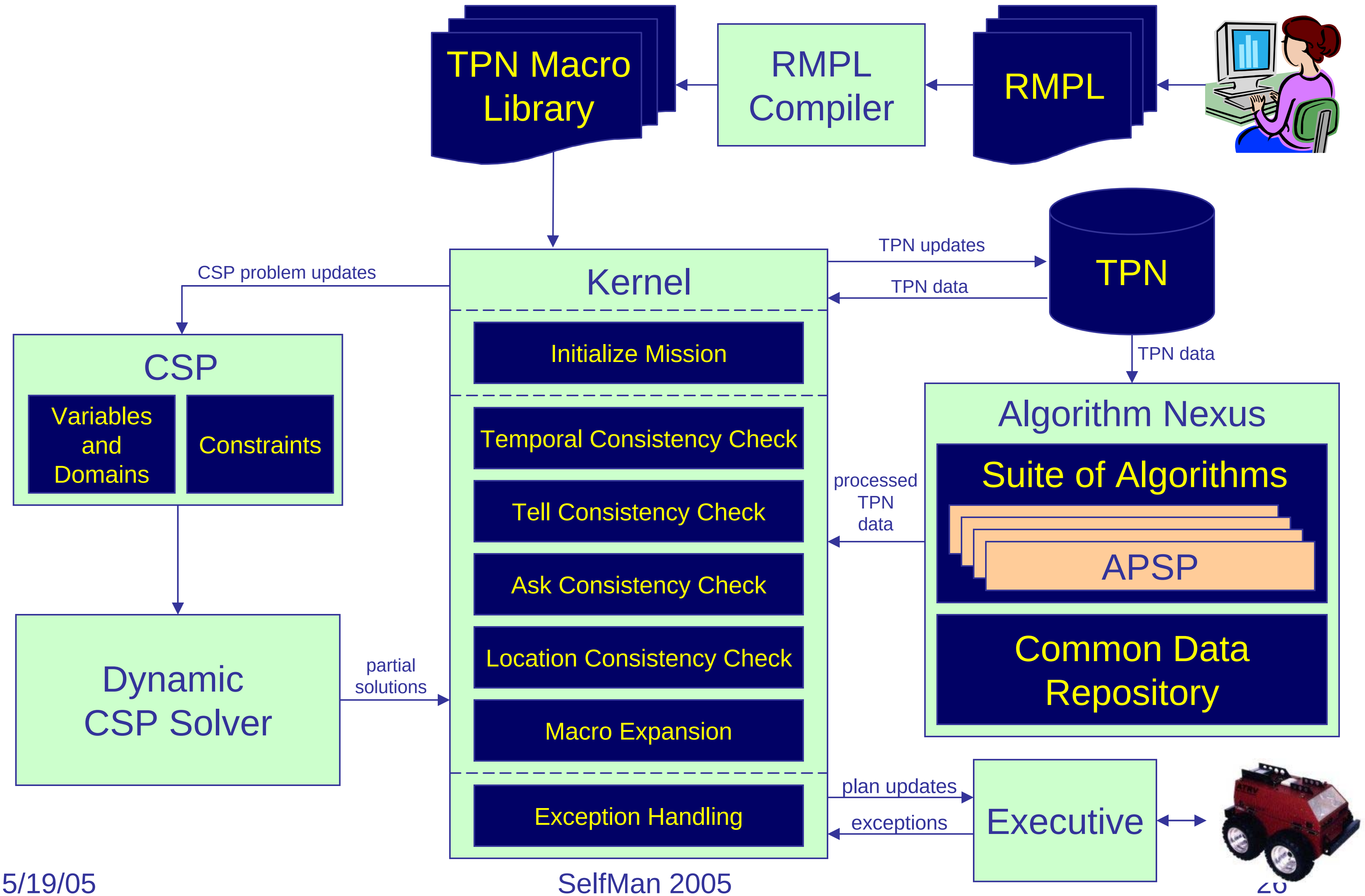
Outputs: SegmentedImage

```
class SegmentColor ()
{
    RawImage image_in;
    SegmentedImage image_out;

    mode Usable ((image_in = Nominal)) { …    }

    mode TooDark ((image_in = Dark)) { … }
}
```

Block Diagram



TPN Macro Library

RMPL Compiler

RMPL

Kernel

Initialize Mission

Temporal Consistency Check

Tell Consistency Check

Ask Consistency Check

Location Consistency Check

Macro Expansion

Exception Handling

CSP problem updates

CSP

Variables and Domains

Constraints

Dynamic CSP Solver

partial solutions

TPN updates

TPN data

TPN

TPN data

Algorithm Nexus

Suite of Algorithms

APSP

Common Data Repository

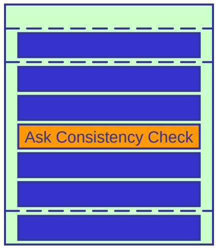
processed TPN data

plan updates

exceptions

Executive

Solution Analysis: Exception Handling

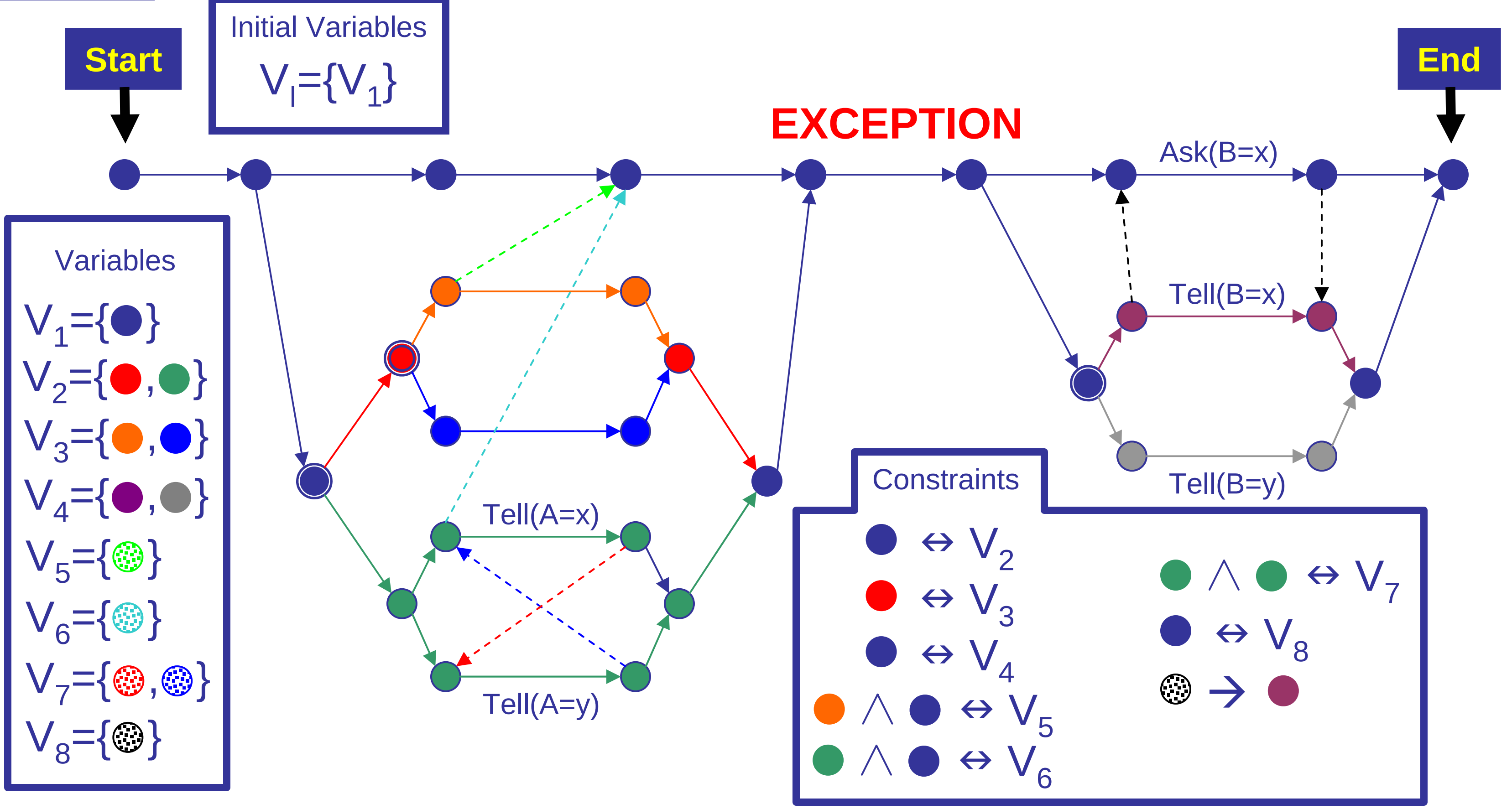


Partial Solution

V_1={🔵} V_2={🔴} V_3={🟠}

V_4={🟣} V_5={▨} V_8={▨}

1. Execution begins…
2. An error occurs, and an exception is thrown



Initial Variables

V_I={V_1}

Variables

V_1={🔵}

V_2={🔴,🟢}

V_3={🟠,🔵}

V_4={🟣,⚪}

V_5={▨}

V_6={▨}

V_7={▨,▨}

V_8={▨}

Constraints

🔵 ⇔ V_2

🔴 ⇔ V_3

🔵 ⇔ V_4

🟠 ∧ 🔵 ⇔ V_5

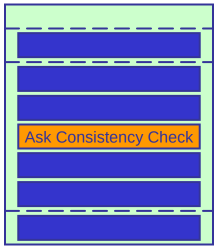
🟢 ∧ 🔵 ⇔ V_6

🟢 ∧ 🟢 ⇔ V_7

🔵 ⇔ V_8

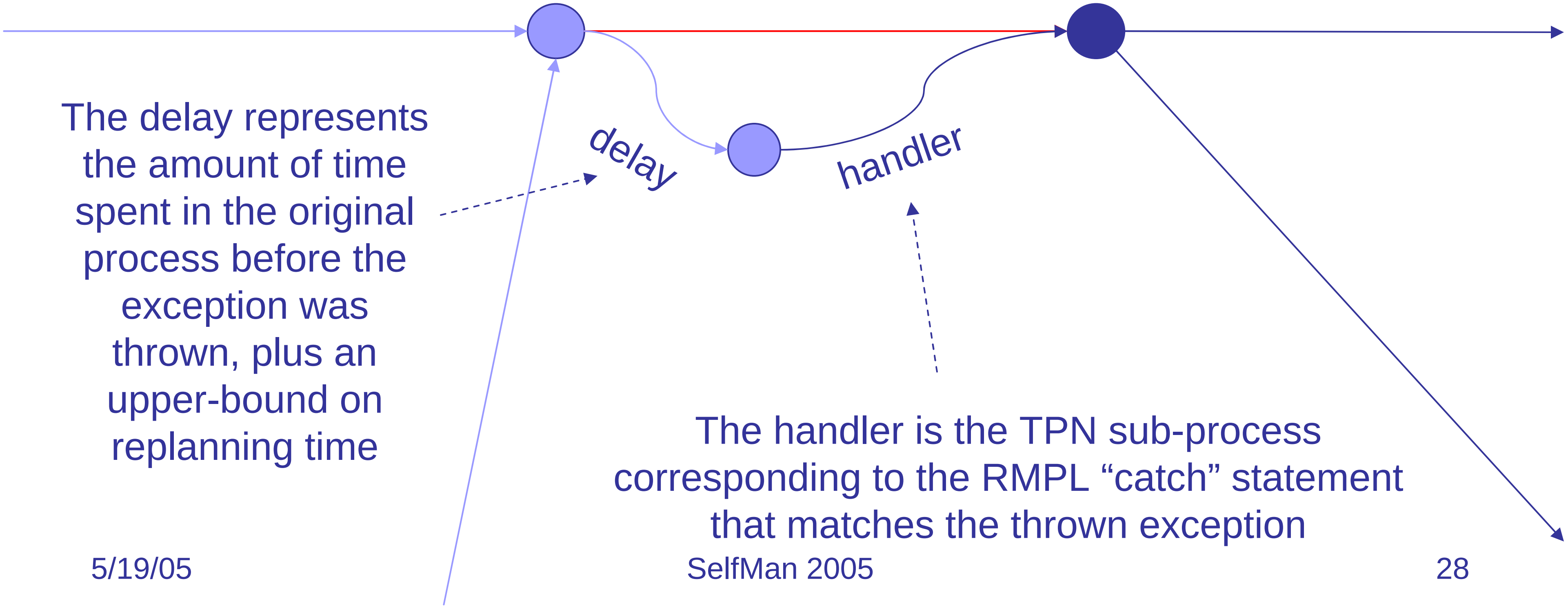
▨ → 🟣

Solution Analysis: Exception Handling



1. Execution begins…
2. An error occurs, and an exception is thrown
3. The exception-handling code is inserted

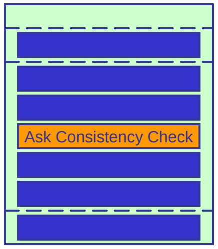
EXCEPTION



The delay represents the amount of time spent in the original process before the exception was thrown, plus an upper-bound on replanning time

The handler is the TPN sub-process corresponding to the RMPL “catch” statement that matches the thrown exception

Solution Analysis: Exception Handling



Partial Solution

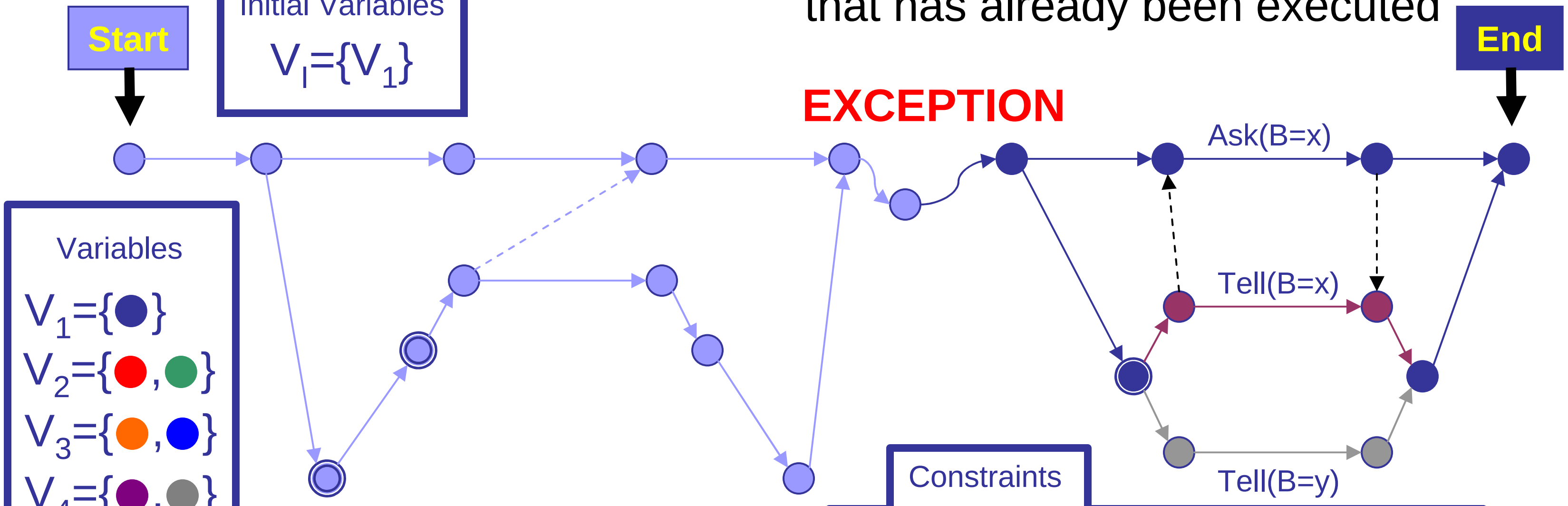
V_1={●} V_2={●} V_3={●}

V_4={●} V_5={◍} V_8={◍}

1. Execution begins…
2. An error occurs, and an exception is thrown
3. The exception-handling code is inserted
4. Replanning begins, pre-selecting anything that has already been executed

Initial Variables

V_I={V_1}



Variables

V_1={●}

V_2={●,●}

V_3={●,●}

V_4={●,●}

V_5={◍}

V_6={◍}

V_7={◍,◍}

V_8={◍}

Constraints

● ⇔ V_2

● ⇔ V_3

● ⇔ V_4

● ∧ ● ⇔ V_5

● ∧ ● ⇔ V_6

● ∧ ● ⇔ V_7

● ⇔ V_8

◍ → ●

Conclusions

- Models of correct operation permits:
  - Detection and Diagnosis of failed components.
  - Reconfiguration of Software/Hardware components to achieve high-level goals
  - Describe goals as abstract state trajectories.
- Software can be handled by adding:
  - Hierarchy to component organization
  - Models of the environment