



IDEA: Intelligent Distributed Execution Architecture

PI: Nicola Muscettola

Team Members: Chuck Fry, Rich Levinson, Chris Plaunt,
Greg Dorais, Vijai Baskaran, Felix Ingrand (LAAS/CNRS),
Mary Bernardine Dias (CMU), Solange Lemai (LAAS/CNRS)

NASA Ames Research Center

Intelligent Distributed Execution Architecture (IDEA)

Nicola Muscettola, NASA Ames

Problem: Advanced autonomous agents are traditionally multi-layered with each layer using different technology with different semantics. This increases software integration and validation costs.

Objectives:

- Simplify an autonomous agent architecture by using the same execution machinery (semantics and implementation) at any layer
- Simplify the integration of autonomous agent software by using a single communication protocol between all layers

Key Innovation:

- IDEA virtual machine implementing a sense-plan-act loop with real-time guarantees
- Use of a reactive model-based planner in the inner loop of execution, with each IDEA agent possibly using a different planner technology provided it is compatible with the IDEA virtual machine
- IDEA communication protocol consisting of the interchange of task networks

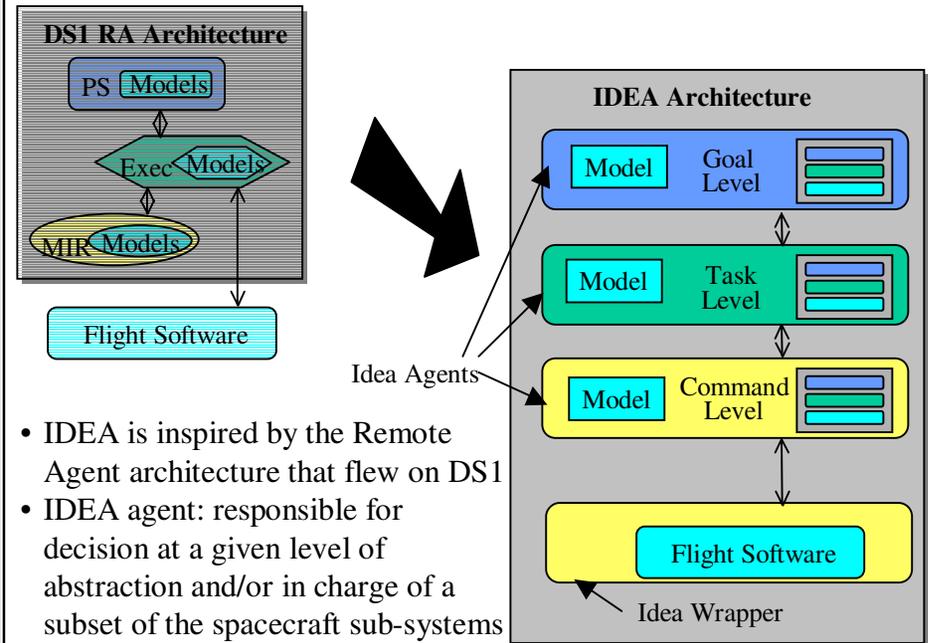
NASA Relevance:

- Software architecture for real-time autonomous systems is key to robust autonomous planetary explorers (Mars Exploration, Europa Lander, Titan aerobot)
- Distributed nature of architecture supports multi-spacecraft missions (earth observing, formation flying interferometry)
- Potentially usable as well-founded fault-protection engine in traditional flight software

Accomplishments to date:

- Full implementation of IDEA architecture
- Identified basic services that are planning technology independent
- Several multi/single agent applications programmed in IDEA
 - Planner and Executive layers for Remote Agent
 - DS1 launch sequence
 - K9 on board rover controller

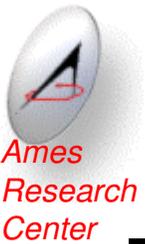
Description/Schedule



- IDEA is inspired by the Remote Agent architecture that flew on DS1
- IDEA agent: responsible for decision at a given level of abstraction and/or in charge of a subset of the spacecraft sub-systems
- IDEA agent relay allow: “legacy” software to interact with IDEA agents by supporting the IDEA communication protocol.

Schedule:

- FY 03: theoretical paper on IDEA computational model, reference implementation ready for distribution, completion of current reference application (Remote Agent)
- FY 04: mapping between procedural execution and IDEA computational model, IDEA agent applications with different planning technologies; translators/analyzers between different procedural execution/planning paradigms and IDEA (in both directions)



Problems with Large Scale Autonomy Systems



- Difficulty in programming
 - Each subsystem has a distinct “virtual machine” and method of programming
 - A domain specialist needs help from reasoning engine specialists to program something like Remote Agent
- Difficulty in validation
 - At times information needs to be duplicated at different levels
 - Different encoding makes validation difficult, e.g., it is not easy to envision a unified approach to validation applicable to all levels
- Lack of uniformity with flight software
 - Autonomy seen as disposable add on
 - What about flight software fault protection?



Fundamental IDEA Hypothesis

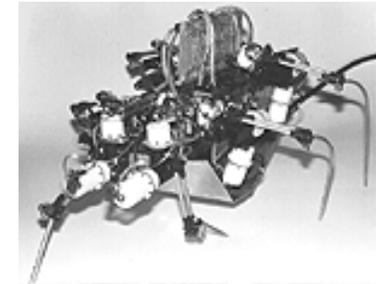
- The only coordination mechanism needed in a complex control system is:
 - **An explicit representation of a plan in a plan database**
 - **An interpreter that integrates sensor information and goals in the plan database and starts new tasks**
 - **A planner based on sub-goaling (and constraint propagation)**
- This is true at any level of abstraction

Intelligent Distributed Execution Architecture (IDEA)

“ Now, wait a minute ...”



Shakey the robot
1966-1972



Behavior-based robots
1986

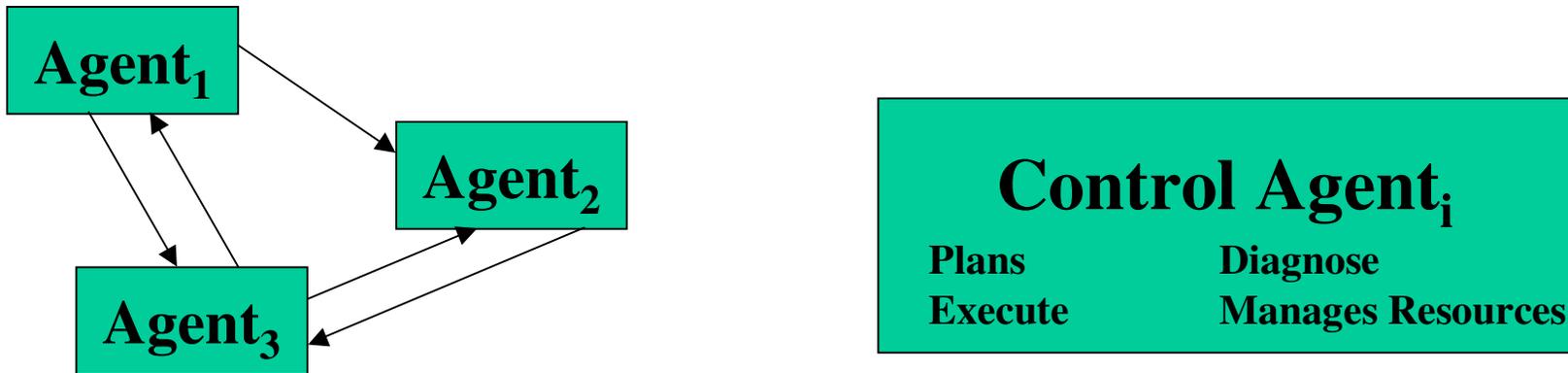


Traditional Arguments against Planning



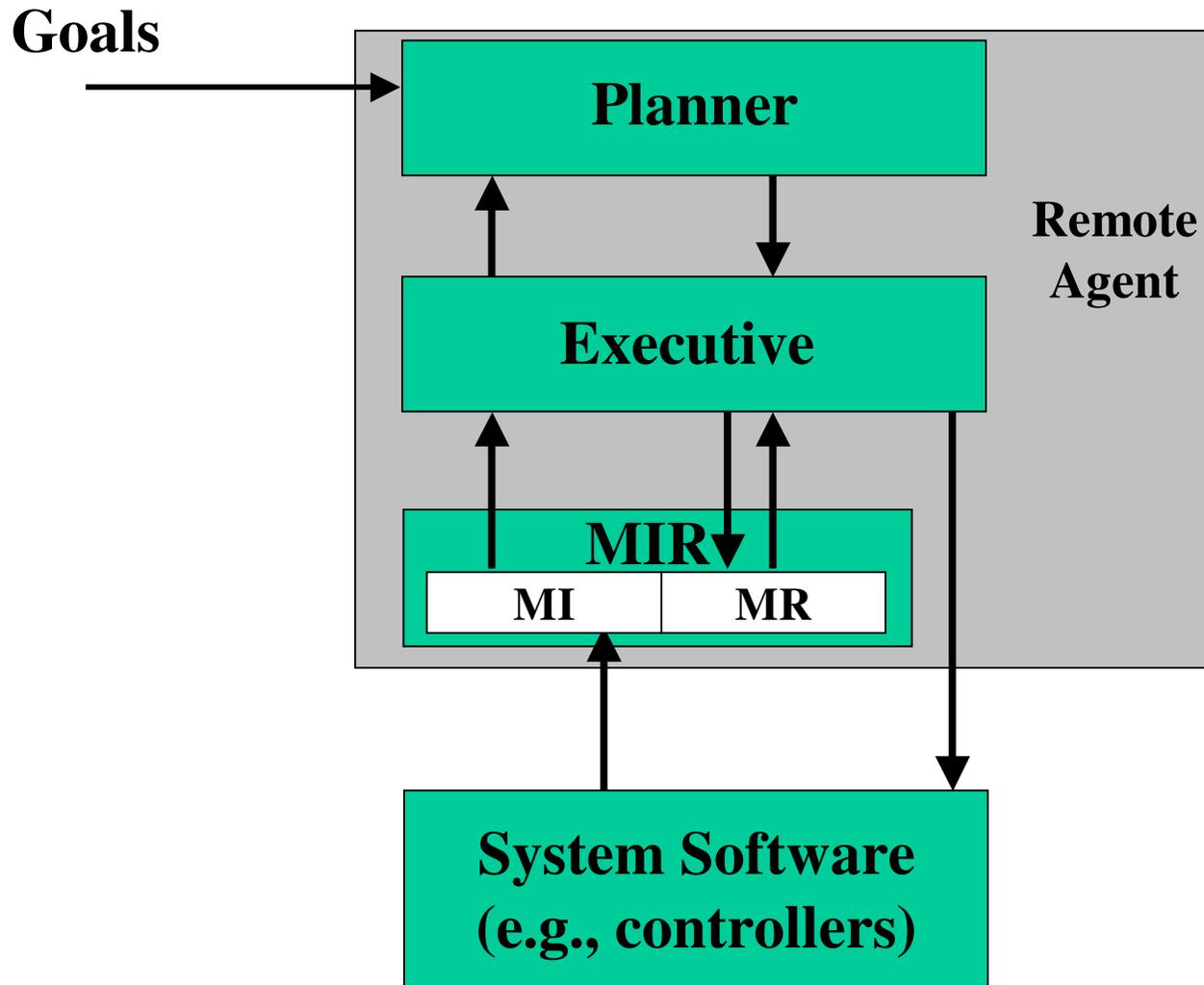
- Some truths
 - **Combine simple reactive controllers to build complete autonomous systems**
 - **Focus on reactivity, i.e., the ability to keep up with a dynamic world**
 - **Minimally constraining architectural assumptions. Design behaviors internally as most fit to task.**
 - **Keep architecture “light weight”.**
- Some nonsense
 - **"The world is its own best model."**
- Some problems with the alternative
 - **Approaches to programming Artificial Insects have not scaled up to higher cognitive functionalities**

Building control systems as communicating agents

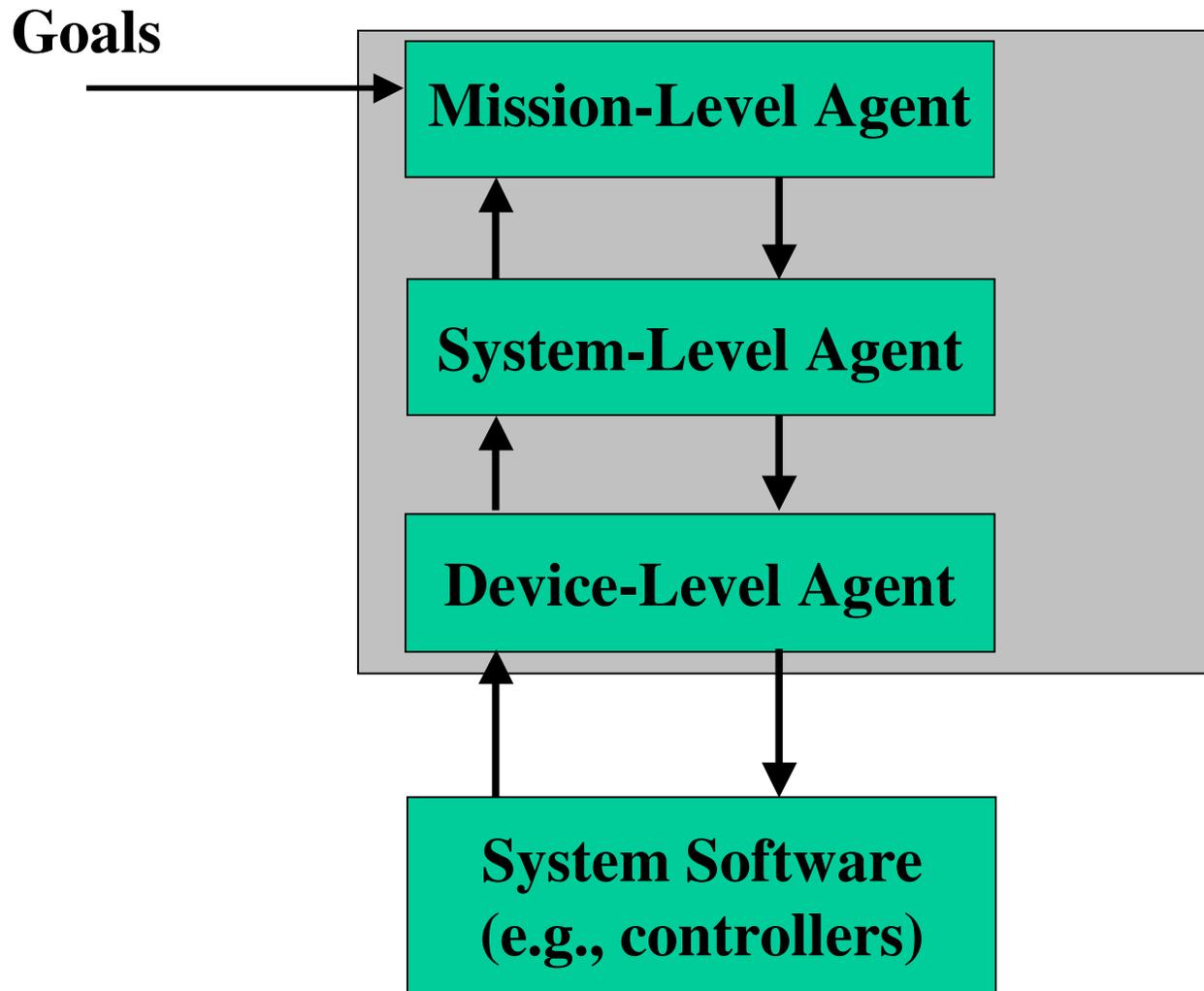


- Complex control systems designed as networks of interacting agents
- Each agent displays all functionalities found in Remote Agent: planning, execution, diagnosis, resource management.
- All agents will use a uniform, declarative communication protocol

Remote Agent

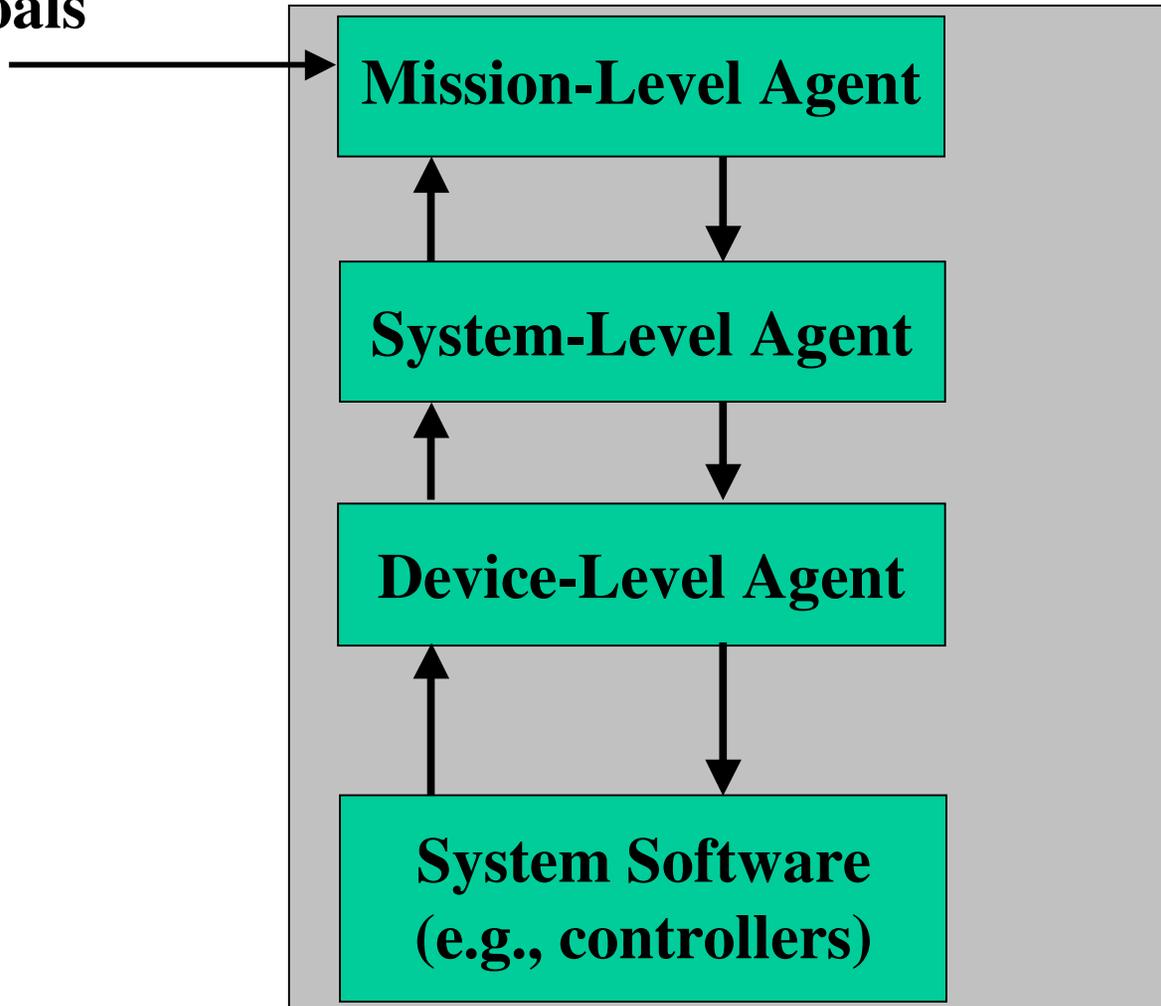


Multi-agent perspective

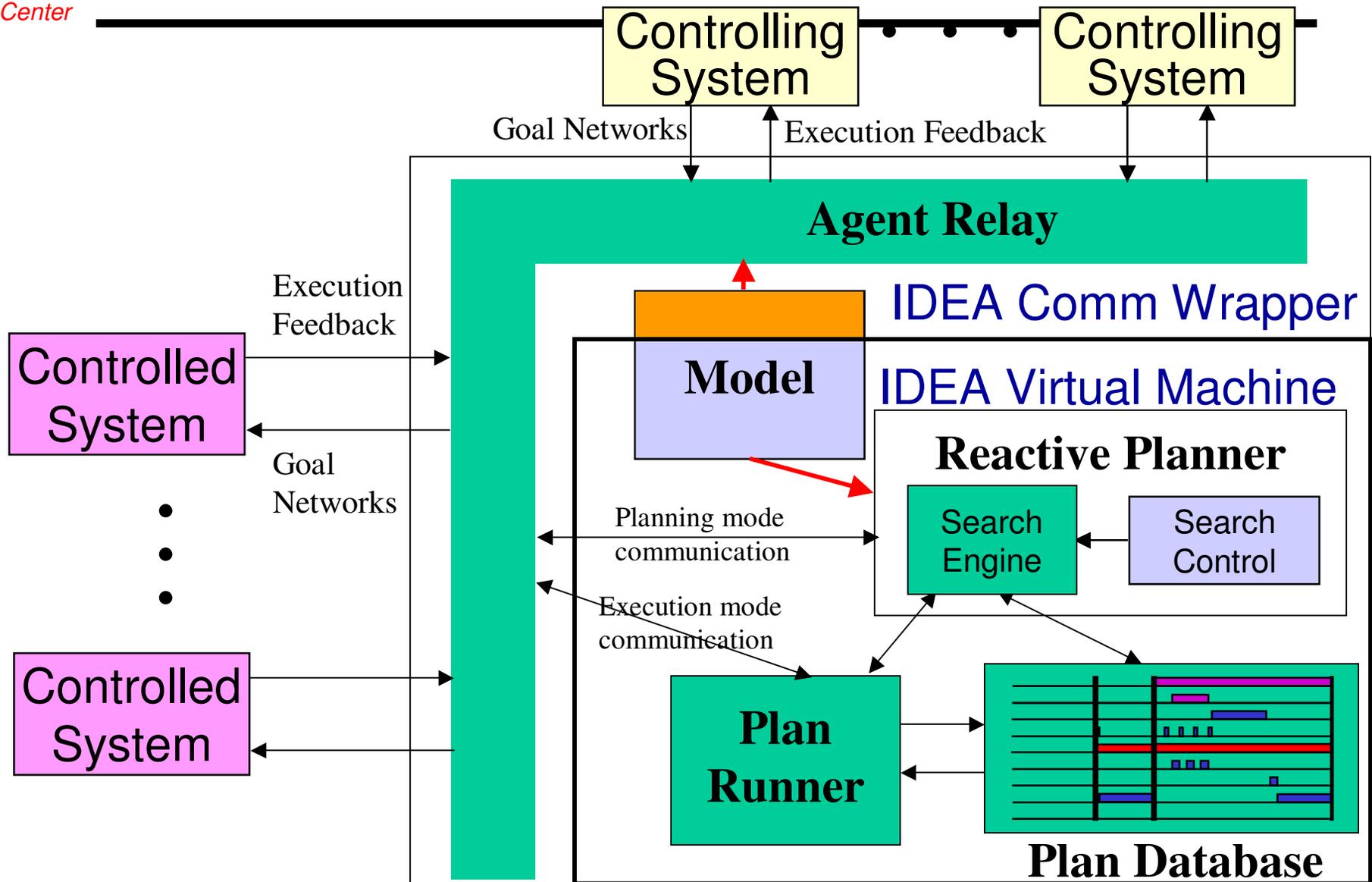


Multi-agent perspective

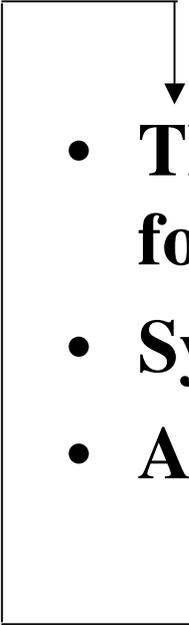
Goals



Internal Agent Architecture



Method of Investigation

- 
- **Theoretical architectural hypothesis formulation**
 - **System architecture services implementation**
 - **Application to non-trivial domains**

New Theoretical Insights

- **Reactive behavior depends on the state of the reactive planner during the reaction cycle**
 - Model this explicitly as *internal timelines*
- **Every return value has an associated communication flag**
 - Under “reliable and instantaneous communication” hypothesis, allows to explicitly model response to “too slow” failure state of other control agents
- **“Latency matching” in multi-latency control systems**
 - Determines how much look-ahead is needed from controlling agent to correctly control a controlled agent
 - Needed for studying “stability” of multi-agent system (“Under which conditions can a controller keep up with a controlled system?”)
- **IDEA is a model of a hybrid synchronous/asynchronous control/computational system**
 - Possibility of deep connections with formal models approaches to same problem
 - This is currently at the frontier of what is known in the field of formal models for distributed systems

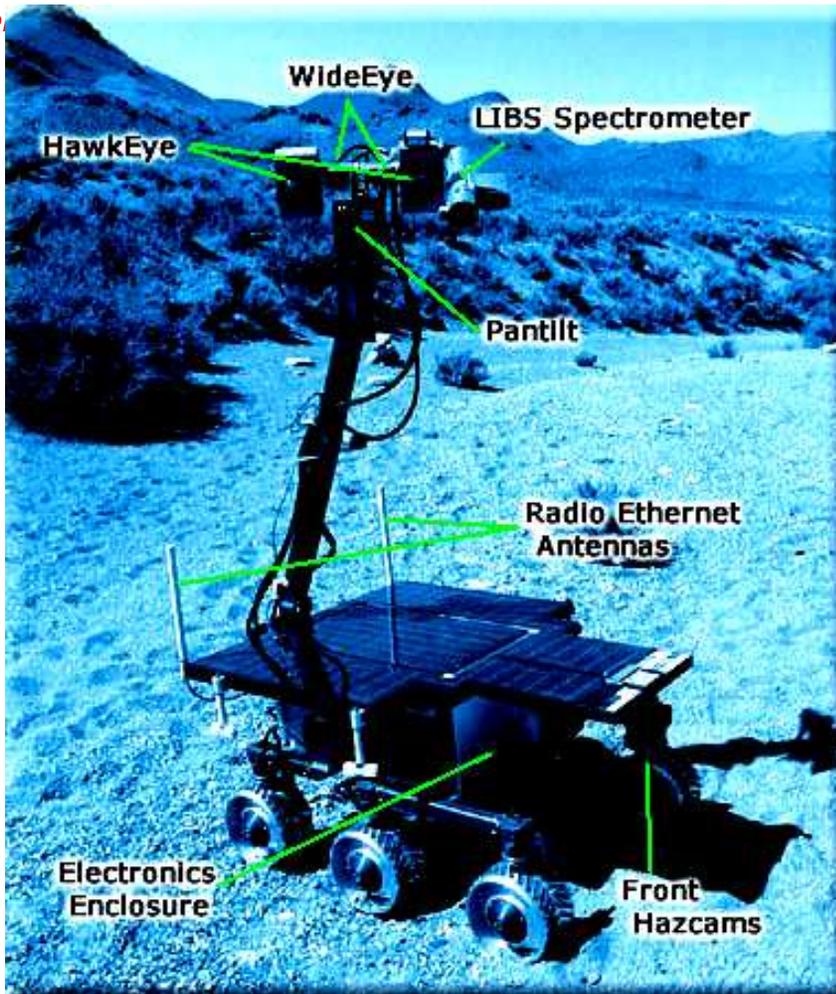
- **Plan Service Layer (PSL)**
 - API providing “glue” between plan runner and plan database/planner
 - Implemented services to ensure that a constraint-based planner (EUROPA based) always operate consistently with execution information
 - Ready to generalize PSL to connect with much wider range of planners
 - Non-timeline based temporal planners (Kirk)
 - Contingent planners (Dave Smith)
 - Repair-based planners (ASPEN)
 - Non-temporal planners
 - Diagnosis and reconfiguration engines (L2, Titan)
- **Agent Relay design**
 - Independent from “message-level layer”
 - Two implementations: CORBA and Reid Simmon’s IPC
 - Does not assume that recipient or sender is an IDEA agent



Applications

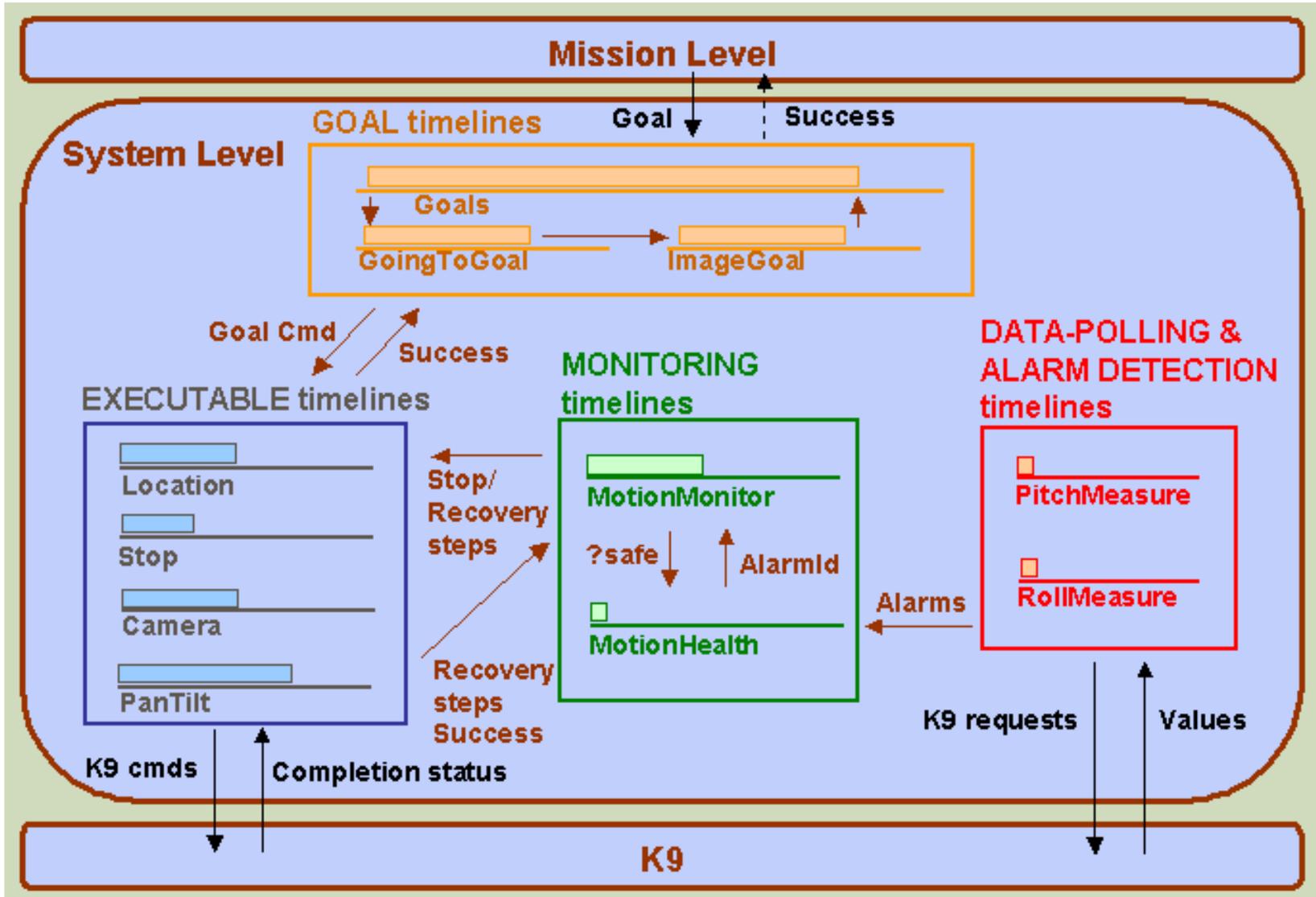
- **Reimplementation of Remote Agent**
 - Demonstrated interleaving of deliberative planning and execution at mission-level (Planner/Plan Runner in Remote Agent)
 - Planning to standby at the mission level (model-based solution of ad-hoc Executive script in Remote Agent)
 - Finishing up System level model (Executive in Remote Agent)
 - Coverage of all subsystems in EXEC working nominal behavior and some faults
 - Preliminary inter-agent Mission/System level communication tests
- **PSA**
- **DS1 launch sequence**
 - Collaborative work with JPL (Abdullah Aljabri's group)
- **On-board executive for K9 rover**
 - Summer project by Bernardine Dias (CMU) and Solange Lemai (LAAS/CNRS)

K9 Rover

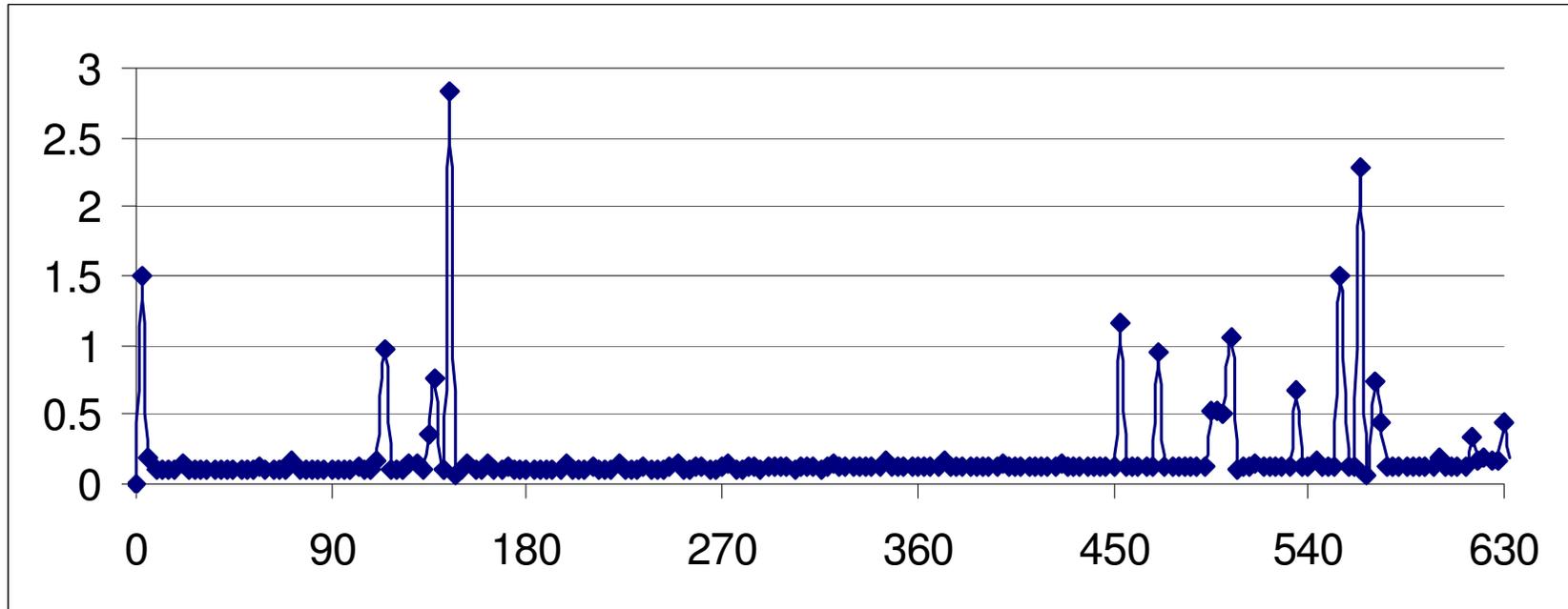


- Pilot
- Obstacle avoidance
- Pantilt
- Vision
- Power
- Fans
- Arm

Implementation of IDEA Model

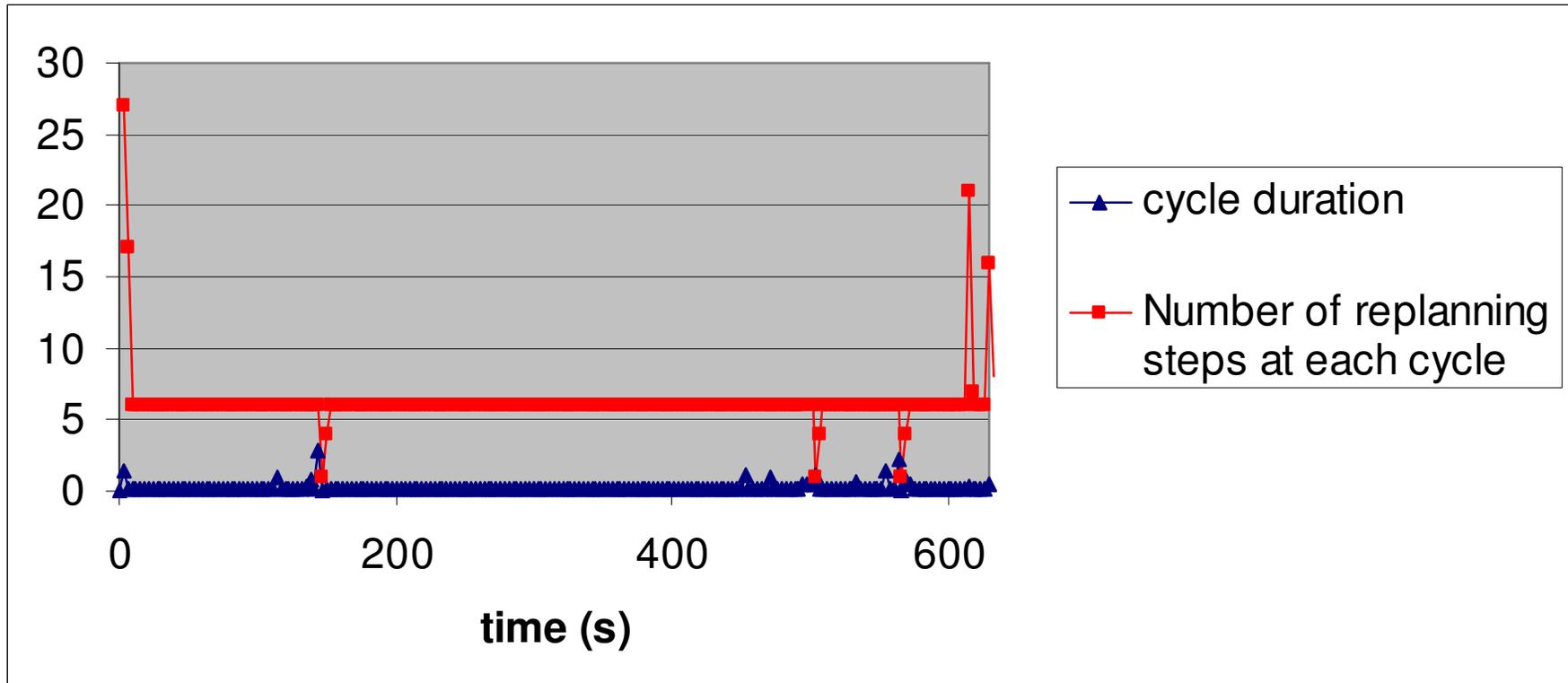


Measured latency at execution



- **Reactive planner is chronological backtracker without search control heuristics**
- **Plan database limited to “two tokens” for each timeline**
- **Dynamic “amnesia”**
- **Run on 300MHz Pentium in Linux**

Search cost at each cycle





Future Work



- **Produce theoretical paper on IDEA computational framework**
- **Produce a reference implementation available for distribution**
- **Extend to interaction between agents at planning time**
- **Use of different plan database technologies and planners**
- **Provide fault-protection for flight software application**
 - DS1 fault protection re-implementation
- **Mapping to procedural execution**
 - Formal mapping from/to BDI model
 - Augmentation/translation with PRS procedural system
- **Integration with diagnosis**
 - Diagnosis = “planning in the past”
- **Multi-agent fault-tolerance**
 - What happens if an agent in the control network must go off line (e.g., it does not respect a latency guarantee)?
- **Look at connection with other areas**
 - Real-Time Active Database transaction management