

CAMERA

Coordination and Management Environments for Responsive Agents

Robert Neches, P/I

Pedro Szekely, Project Leader

University of Southern California

Information Sciences Institute

Marina del Rey, California

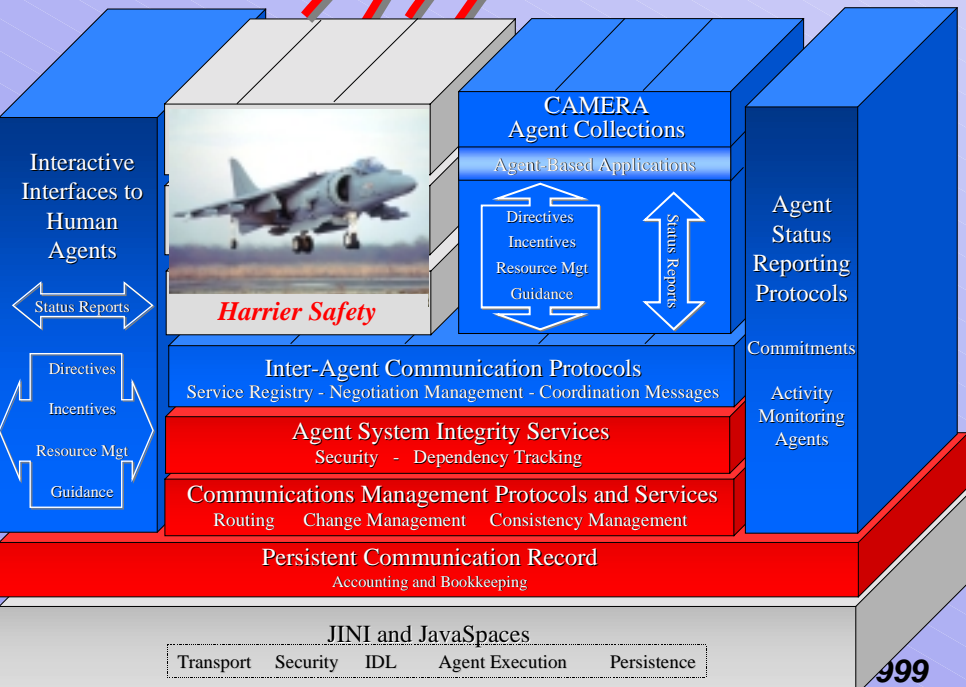
Other contractors: Crystaliz, USC ISI (Balzer), AverStar

CAMERA:

Software Framework and Substantive Applications



*Candidate Demonstrations:
Coordination of Actions to Promote Harrier Aircraft Safety*



Key Ideas:

Collection formation, Renegotiation

- **Negotiation management protocols**
 - Establish “rules of the game”
 - Permits alternative, *task-specific* variants
 - Ensure convergence on taking action
 - Closure through *commitments*
- **Commitment-based group control**
 - Resynchronization after separations
 - Problem / Opportunity Monitoring
- **Renegotiation revisits priorities**

Critical Implications of the CAMERA Approach

If we succeed, the collective system can:

- **Proactively adapt both to problems *and* opportunities...**
because potentially interested agents can monitor report processes, defined when commitments are negotiated
- **Systematically reevaluate priorities in face of conflicts...**
because agents responding to issues use records of negotiations to determine who to renegotiate with
- **Robustly handle system changes, communication breaks...**
because services are linked by request/offer structure and commitment records that facilitate resynchronization

What Is Our Approach?

- Agents communicate by *content-based addressing*
- *Negotiation management protocols* govern interaction
 - Soft policies, heterogeneous within system
- Formation of *agent collections* dependent on
 - Problem characteristics
 - Contextual constraints
- Agent collection: *active participants*, plus *hangers-on*
- *Commitment based negotiation* to enable localized and distributed decision making
- *Monitoring* of commitments, *renegotiation triggers*
 - Proactively address problems
 - Take advantage of opportunities for better solutions

Topics Covered in Negotiation Management Protocols

- *Negotiation response, timing and scheduling constraints*
 - Parameter bounds on counterproposals
 - Negotiation cut-off limits (time, first acceptable, nth, ...)
 - Scheduling and precedence
- *Capability/requirement descriptions and responses*
 - Domain-specific descriptions of service requests
- *Commitment, coordination, role and responsibility relations*
- *Reporting mechanisms and formats*
 - Form and content of reporting structures

Negotiation Management Protocols: Constraints on the Process

- RFPs are requests for services annotated by constraints
 - Differing, task-specific constraints allowed within same system
- Negotiation is based on submitting *counterproposals*
 - Constraints may affect whether to respond, form of response
 - Response offers commitments
 - Response need not be compliant
- Want to treat constraints as a preference hierarchy
 - Non-compliant responses are not rejected
 - Processed if compliant responses fail to satisfy a service request
- Motivation:
 - Favor efficiency whenever possible
 - Let the system degrade gracefully if constraints cannot be met

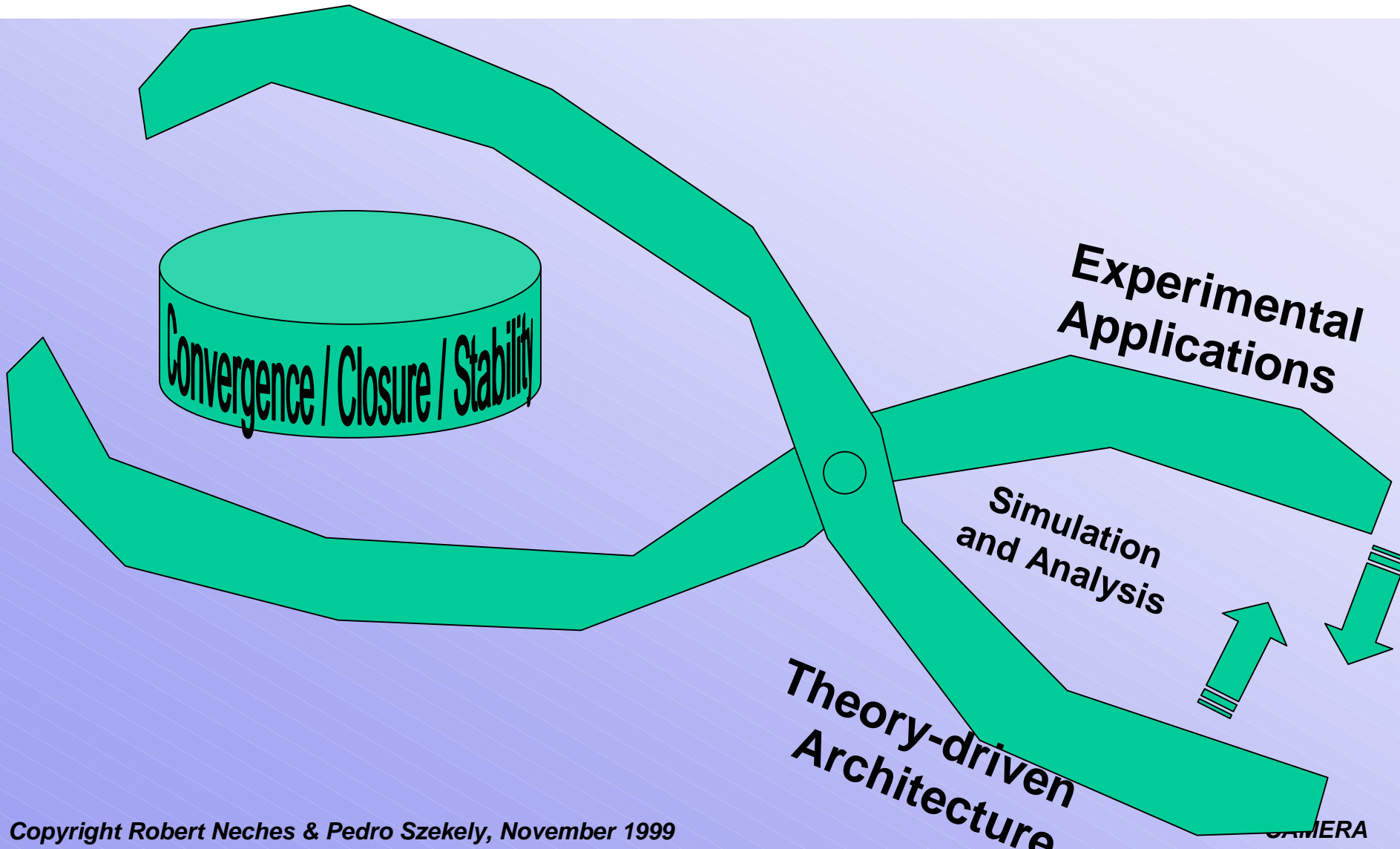
Key Ideas

- Soft negotiation protocols = meta-model
 - Heterogeneous system:
moves local knowledge into negotiation process itself
 - Allows experimentation with alternative regimes
- Notion of ranges around requirements in RFPs
 - Puts bounds on topics for renegotiation
 - Balances flexibility and efficiency
- Commitments, *with progress reports*
 - Support monitoring, triggering of renegotiations
 - Renegotiate *before* you're in trouble, not after

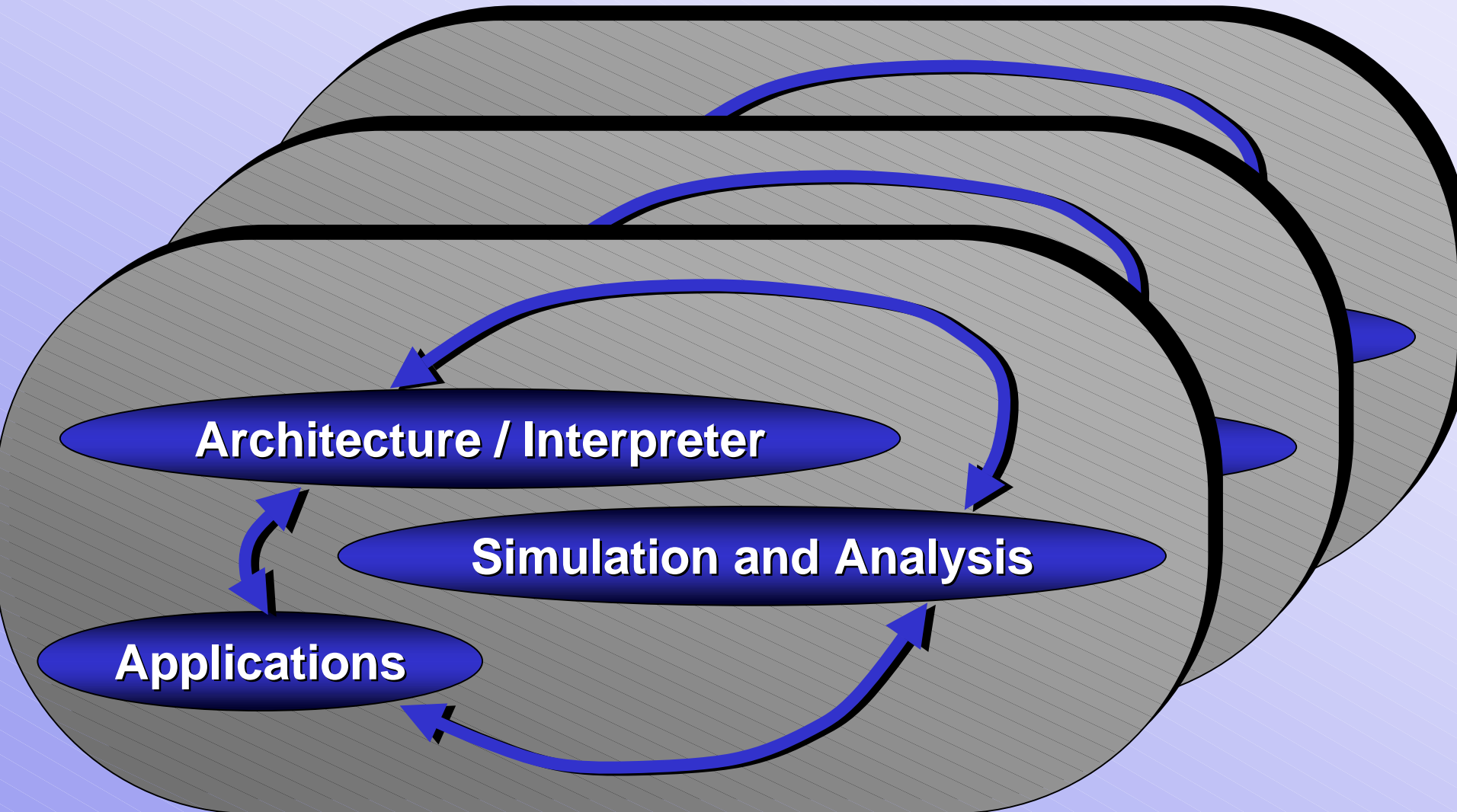
Convergence / Closure / Stability: Systems Approach but No Silver Bullets

- **Time limits** imposed by protocols
- **Ranges** around RFP parameters **focus search**
 - Discourage exploring unproductive alternatives
 - If out of time, select “Least Disrespectful Variants”
- **Heterogeneous protocols** let local knowledge tune each part of problem separately
- **“Global stress level indicators”** speed reaction:
Modify protocols with parameters, conditionals
- **Proactive renegotiation**: more time for repairs

Research and Development Strategy



Strategic Plan



Status

Developed
picture of
mature system



Current system
just taking its first steps

CAMERA

Software Development Progress Report

- Started in September
- Specified 3-layer API for basic services
- Specified API for a simple protocol
- Implemented agent simulator
- Implemented simple version of SNAP

CAMERA Layered API

Application Developers

■ Application developer layer

- Isolates developers from agent communication layers
- Defines methods agent objects must implement to support each protocol
- Strongly-typed Java API – protocol specific

Negotiation Researchers

■ Messaging layer

- Support for connecting to multiple infrastructures
 - Agent simulator
 - Analysis tools
 - Distributed agent communication

Deployment

■ Transport layer

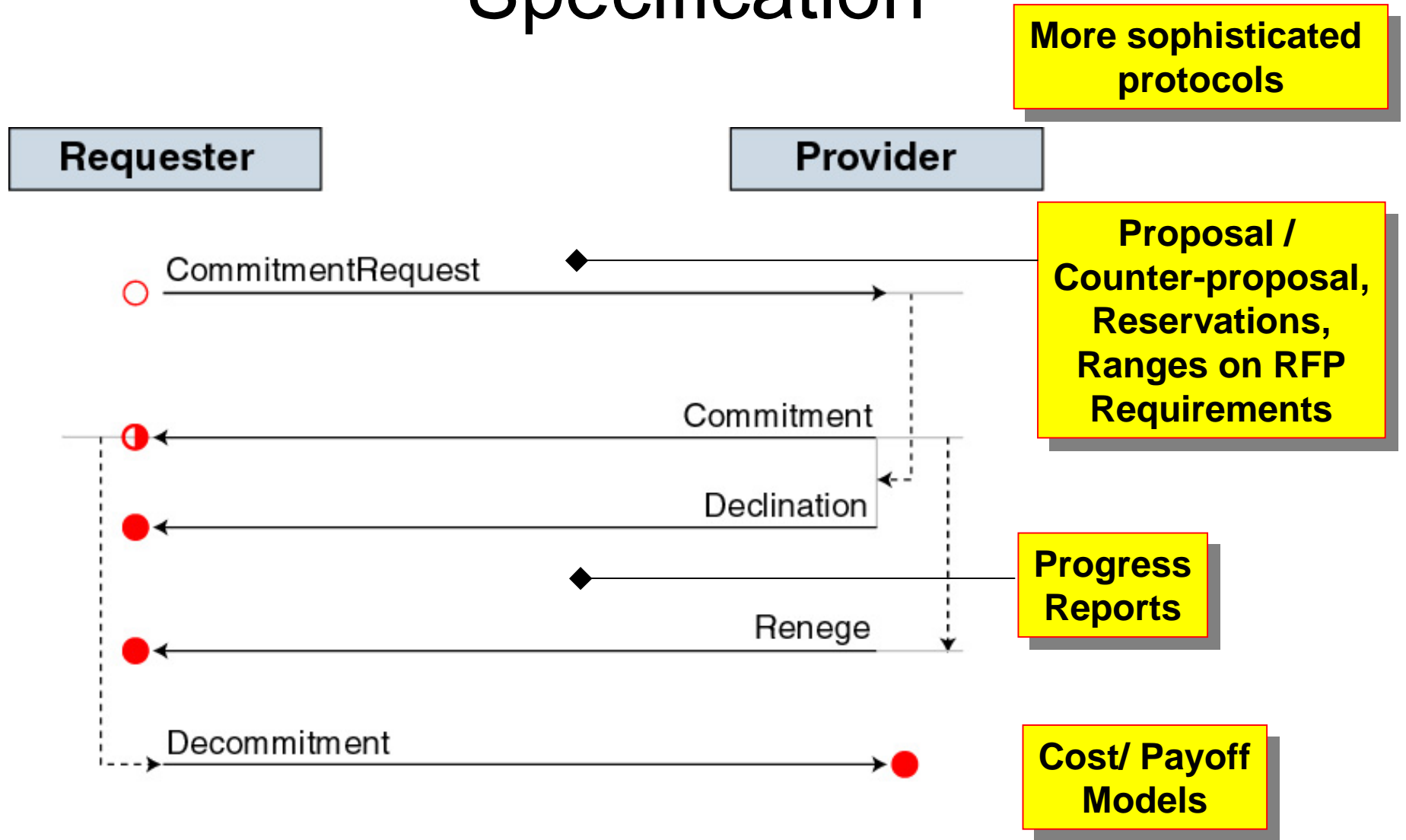
- Marshalling and unmarshalling of messages
- Defines transport mechanism for distributed support
- First implementation will use HP e-speak

Negotiation Protocol Specification

- Message types and parameters
- Valid responses

- Specification
 - Documentation
 - Java API
 - XML messages

Commit/Decommit Protocol Specification



Commit/Decommit Protocol

Java API

```
interface CdProvider extends Provider {  
    public void cdCommitmentRequest(CdCommitmentRequest);  
    public void cdDecommit(CdDecommitment); }
```

```
interface CdRequester extends Requester {  
    public void cdCommit(CdCommitment);  
    public void cdDecline(CdDeclination)  
    public void cdRenege(CdRenege); }
```

```
interface CdCommitmentRequest extends CdNegotiationMessage {  
    public DomainRequest getDomainRequest();  
    public CdCommitment composeCommitment(DomainResource);  
    public CdDeclination composeDeclination(); }
```

SNAP: Build Weekly Flight Schedules

Objective

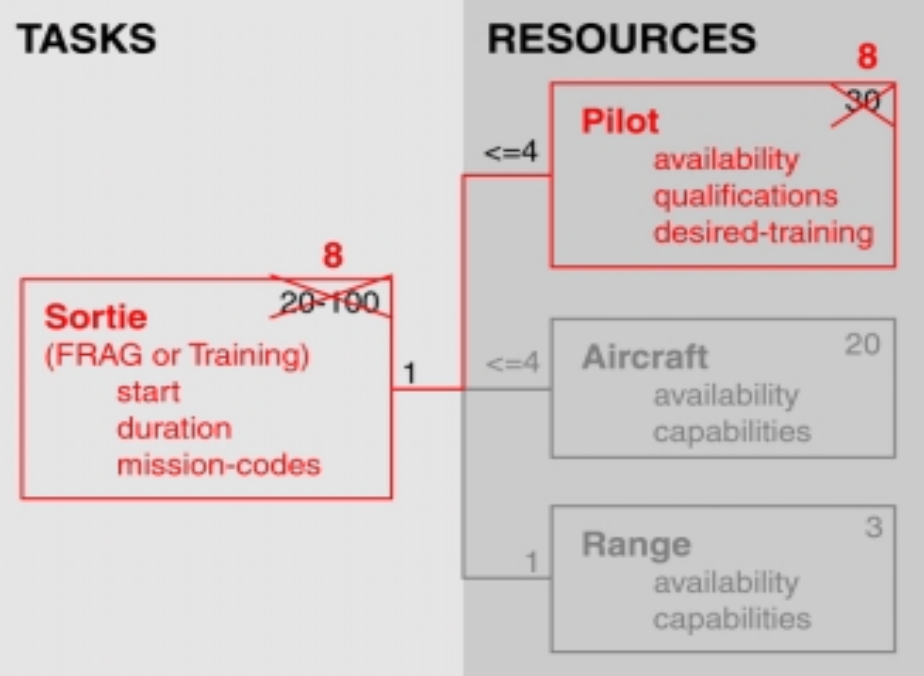
Build weekly flight schedule

Input

- FRAG sorties
 - Start, duration & mission codes
- Training goals for the week
 - Mission codes to emphasize

To do

- Define training sorties
 - How many, what mission codes
- Assign resources to all sorties
- Schedule training sorties

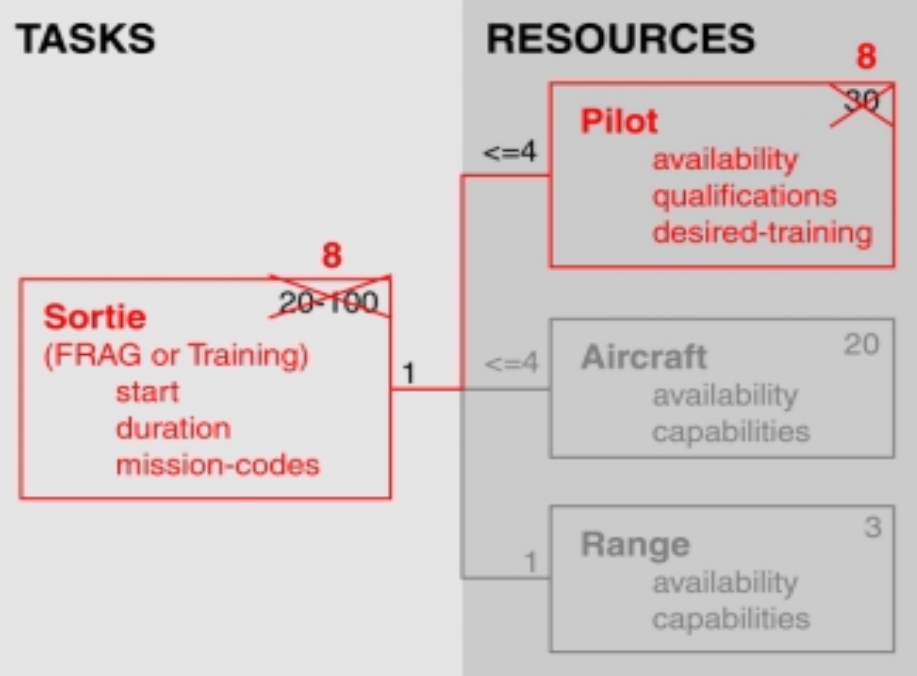


CONSTRAINTS

Pilot.qualifications *can-perform* **Sortie.mission-codes**
Pilot.availability *contains* **Sortie.start + duration**
Aircraft.capabilities *can-perform* **Sortie.mission-codes**
Aircraft.availability *contains* **Sortie.start + duration**
Range.capabilities *support* **Sortie.mission-codes**
Range.availability *contains* **Sortie.start + duration**
... lunar light, weather constraints

EVALUATION/UTILITIES

Schedule/ Sortie	<ol style="list-style-type: none">1) Minimize risk: depends on pilot qualifications, recent flight history and mission codes2) Perform all FRAG sorties3) Maximize readiness (train pilots)
Pilots	<ol style="list-style-type: none">1) Make fast progress in curriculum (fly desired mission codes)2) Don't let qualifications expire
Aircraft	<ol style="list-style-type: none">1) Maximize usage



CONSTRAINTS

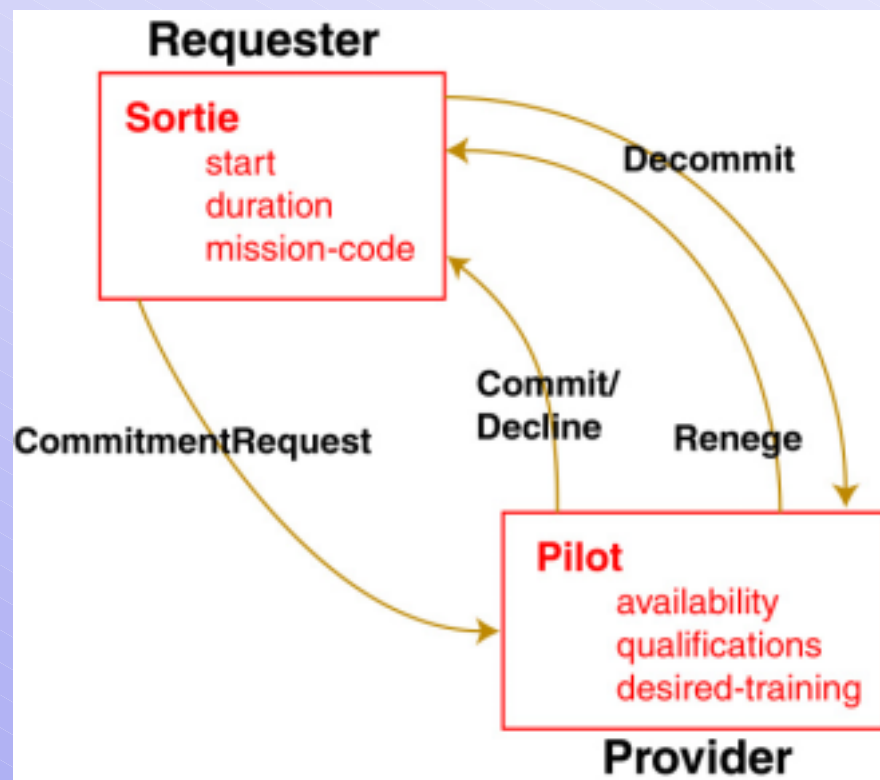
Pilot.qualifications *can-perform* **Sortie.mission-codes**
Pilot.availability *contains* **Sortie.start + duration**
Aircraft.capabilities *can-perform* **Sortie.mission-codes**
Aircraft.availability *contains* **Sortie.start + duration**
Range.capabilities *support* **Sortie.mission-codes**
Range.availability *contains* **Sortie.start + duration**
 ... lunar light, weather constraints

EVALUATION/UTILITIES

Schedule/Sortie	<ol style="list-style-type: none"> 1) Minimize risk: depends on pilot qualifications, recent flight history and mission codes 2) Perform all FRAG sorties 3) Maximize readiness (train pilots)
Pilots	<ol style="list-style-type: none"> 1) Make fast progress in curriculum (fly desired mission codes) 2) Don't let qualifications expire
Aircraft	<ol style="list-style-type: none"> 1) Maximize usage

Simple-SNAP

- Agent-Based solution using the Commit/Decommit protocol



Simple-SNAP Simulation

■ Purpose

- Illustrate CAMERA approach
- Empirically test convergence and stability

■ Experiment

- 5 Data sets
 - Many or few mission/qualification matching
 - Many or few time-slot/availability overlap
- Two pilot strategies (same negotiation protocol)
 - If available, sign up; if not available, decline
 - If a better offer comes, renege old and commit to new one
- 10 runs for each set reordering inputs

Example

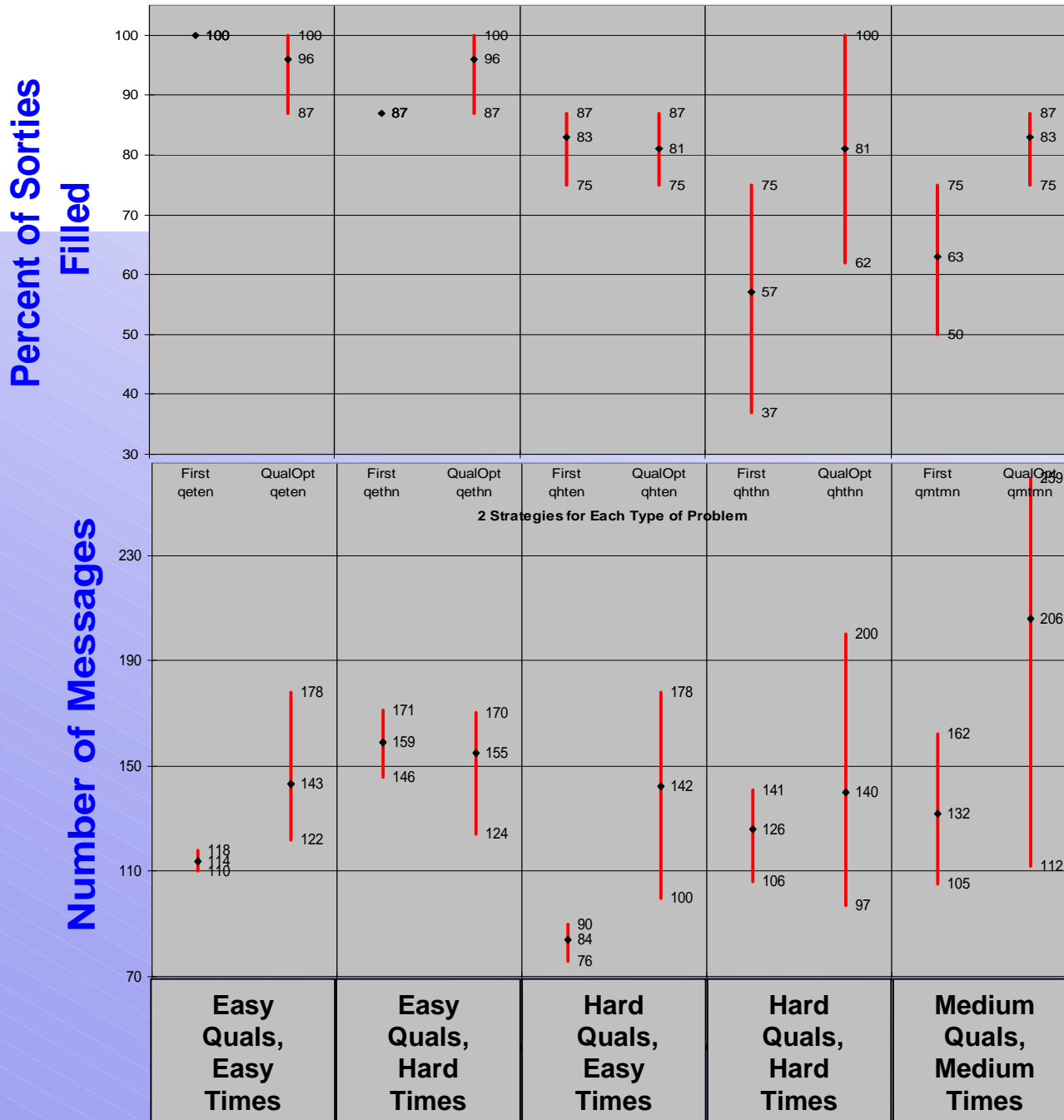
Convergence analysis

Observations

- Neither strategy solves the problem
- "Optimize quals" finds more solutions
- High sensitivity to input ordering

Commit/ Decommit protocol is too greedy:

- Locks critical resources for the wrong task



Results

■ Got started!

- Built first version of architecture, protocols and agents
 - Supports easy experimentation with protocols and strategies
 - Supports distributed implementation for deployment
- Built simulator and started experimenting
 - Agents are very simple, but the data is interesting

■ Paved the way for rapid progress

Next Steps

- More sophisticated protocols & strategies
 - Proposals/counter-proposals
 - Progress reports
 - Cost/payoff models
- Increased application realism
 - Aircraft, ranges, weather, ...
 - Connection to real data (Averstar data warehouse)
- Negotiation protocol handbook
 - Protocol/strategy vs. problem attributes
 - Based on experimentation and theory