Reflections about Mobile Agents and Software Engineering

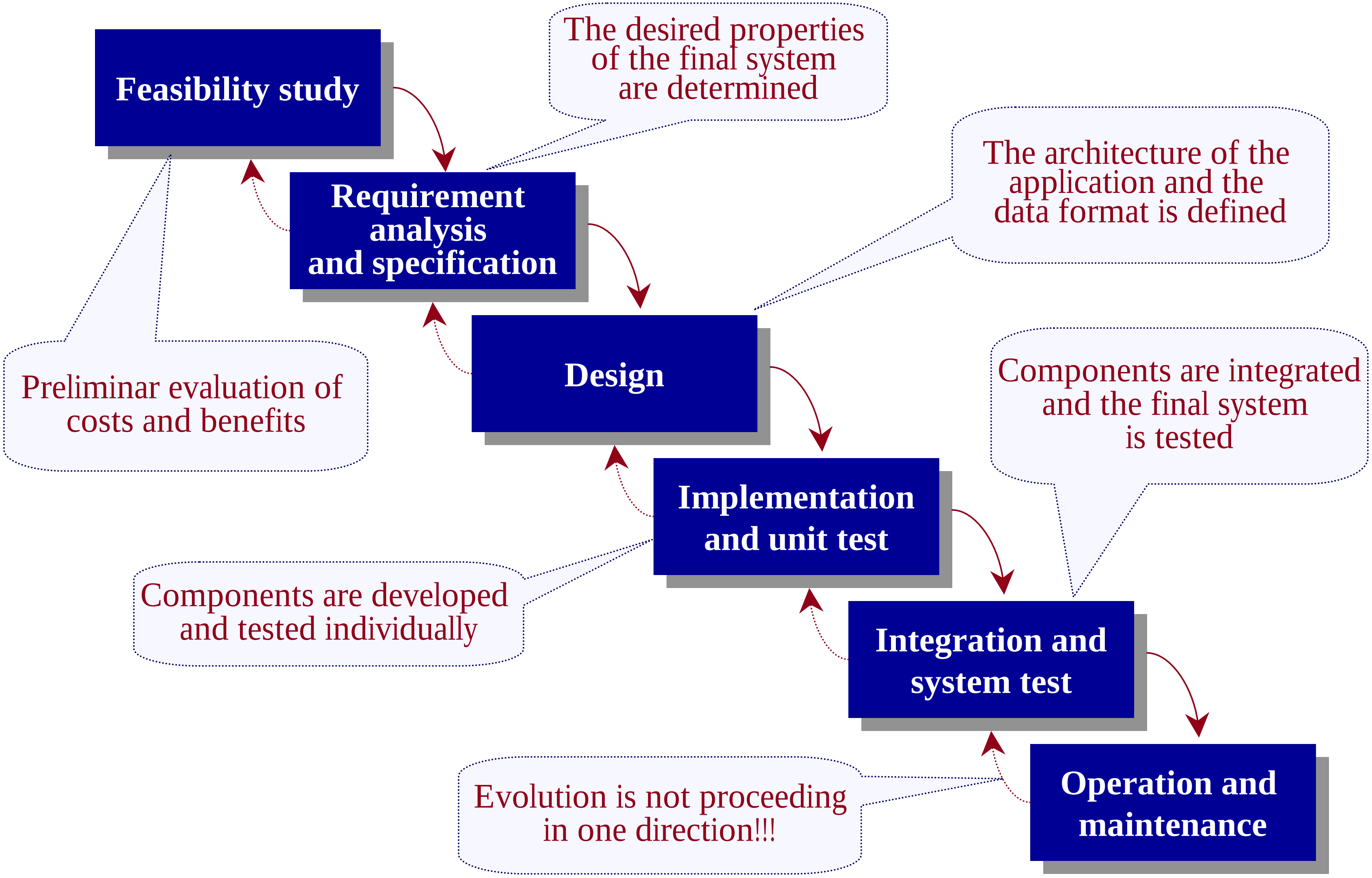
Gian Pietro Picco

Dipartimento di Elettronica e Informazione
Politecnico di Milano, Italy
picco@elet.polimi.it
http://www.elet.polimi.it/~picco

Motivation and Disclaimer

- Mobile agents: “nice solution in search of a problem”
  - “To talk about software *engineering*, you should find some application to engineer, first”
  - No “common sense” application, like Web applets
  - So far, we did not manage to convince people that mobile agents are a good idea; at most we convinced them it is not necessarily a bad idea
- Software engineering: still struggling with the “software crisis”
- Still, both fields have lessons to learn from each other

Classic Software Engineering



Feasibility study

Preliminar evaluation of costs and benefits

Requirement analysis and specification

The desired properties of the final system are determined

Design

The architecture of the application and the data format is defined

Implementation and unit test

Components are developed and tested individually

Integration and system test

Components are integrated and the final system is tested

Operation and maintenance

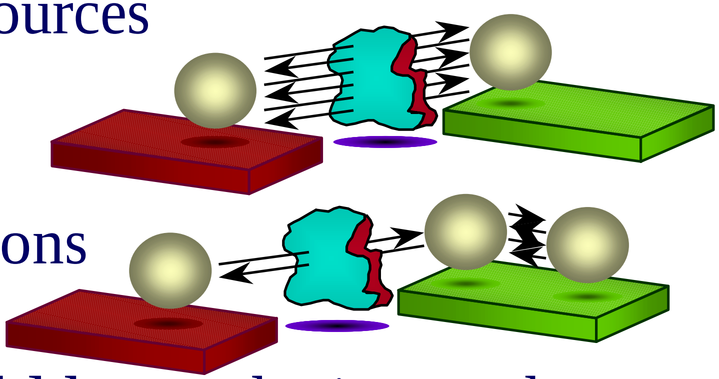
Evolution is not proceeding in one direction!!!

Feasibility Study

- Usually, a specific, non-conventional technical solution is motivated by a specific problem
- In the context of civil engineering:
  - *Problem*: a bridge is placed in an area where earthquakes occur frequently
  - *Solution*: use dynamic control techniques
- In the context of computer science:
  - *Problem*: search in a huge amount of unstructured data
  - *Solution*: data mining
- In the context of mobile agents:
  - *Problem*: somehow, the use of mobile agents is mentioned in the Technical Annex
  - *Solution*: use mobile agents, no matter what

Qualitative Arguments

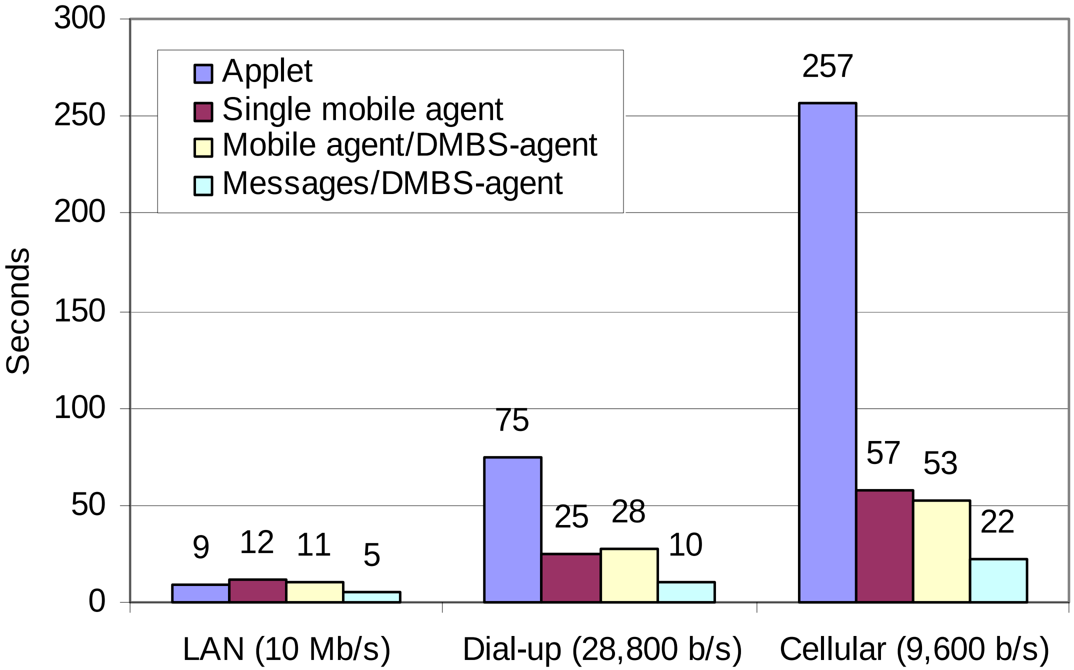
- A lot are documented in the literature
  - Mobile agents: Are They a Good Idea? (Harrison, Chess, Kershenbaum, 1995)
  - Understanding Code Mobility (Fuggetta, Picco, Vigna, 1998)
- Main advantages:
  - Better use of communication resources
  - Enhanced flexibility
  - Improved fault tolerance
  - Support for disconnected operations
  - Protocol encapsulation
- Very few are verified on the field by analyzing real applications



SW Engineering for Mobile Agent ***Applications***

Quantitative Arguments

- They constitute only a small fraction of the literature
  - and studies of good quality constitute an even smaller fraction



Mobile Agents for WWW Database Access
(Papastravrou, Samaras, Pitoura, ICDE 99)

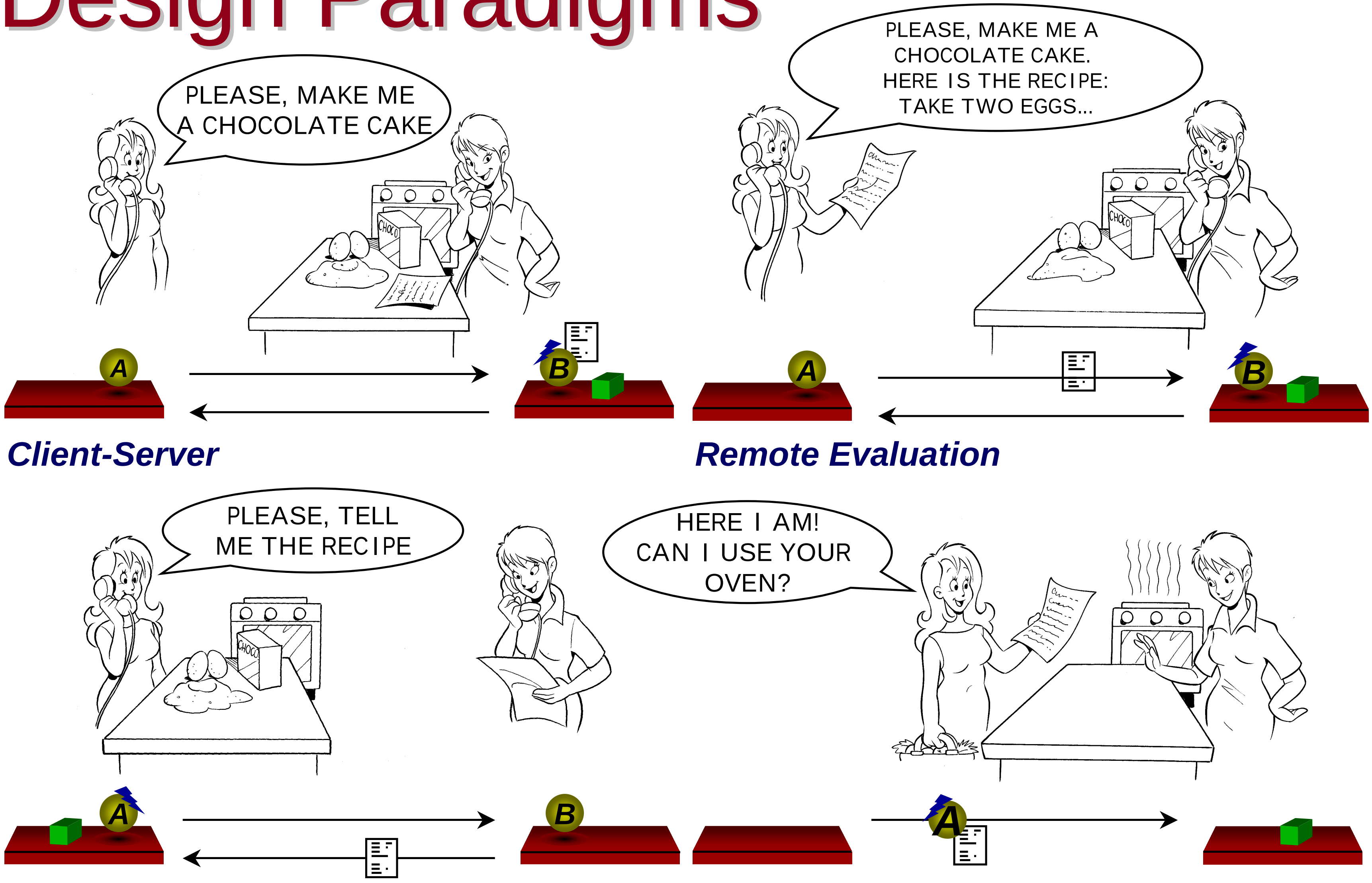
Requirement Specification

- The key question on the table is “how to model mobility”?
- Are traditional languages sufficient?
  - Is there anything really new that mobility brings in?
  - Location vs. context
- Are current proposals suited for the specification of real-world applications?
  - Do you really want to model your application using Ambients?
- Can we model logical and physical mobility using the same language?

SW Engineering for Mobile Agent *Applications*
Design

- This is where mobile code and mobile agents bring one (and possibly the only) fundamental revolution: location is no longer a deployment detail, rather it becomes a first-class design concept
- Logical mobility is essentially a new design tool (as opposed to physical mobility, which essentially defines new requirements)
- Some fundamental questions:
  - What are the fundamental architectural paradigms for mobile agents and, more in general, mobile code?
  - How to choose among design paradigms?
  - When would you use mobile code as opposed to mobile agents?

Design Paradigms


PLEASE, MAKE ME A CHOCOLATE CAKE
CHOCO
A
B
PLEASE, MAKE ME A CHOCOLATE CAKE. HERE IS THE RECIPE: TAKE TWO EGGS...
CHOCO
A
B
PLEASE, TELL ME THE RECIPE
CHOCO
A
B
HERE I AM! CAN I USE YOUR OVEN?
A

Client-Server

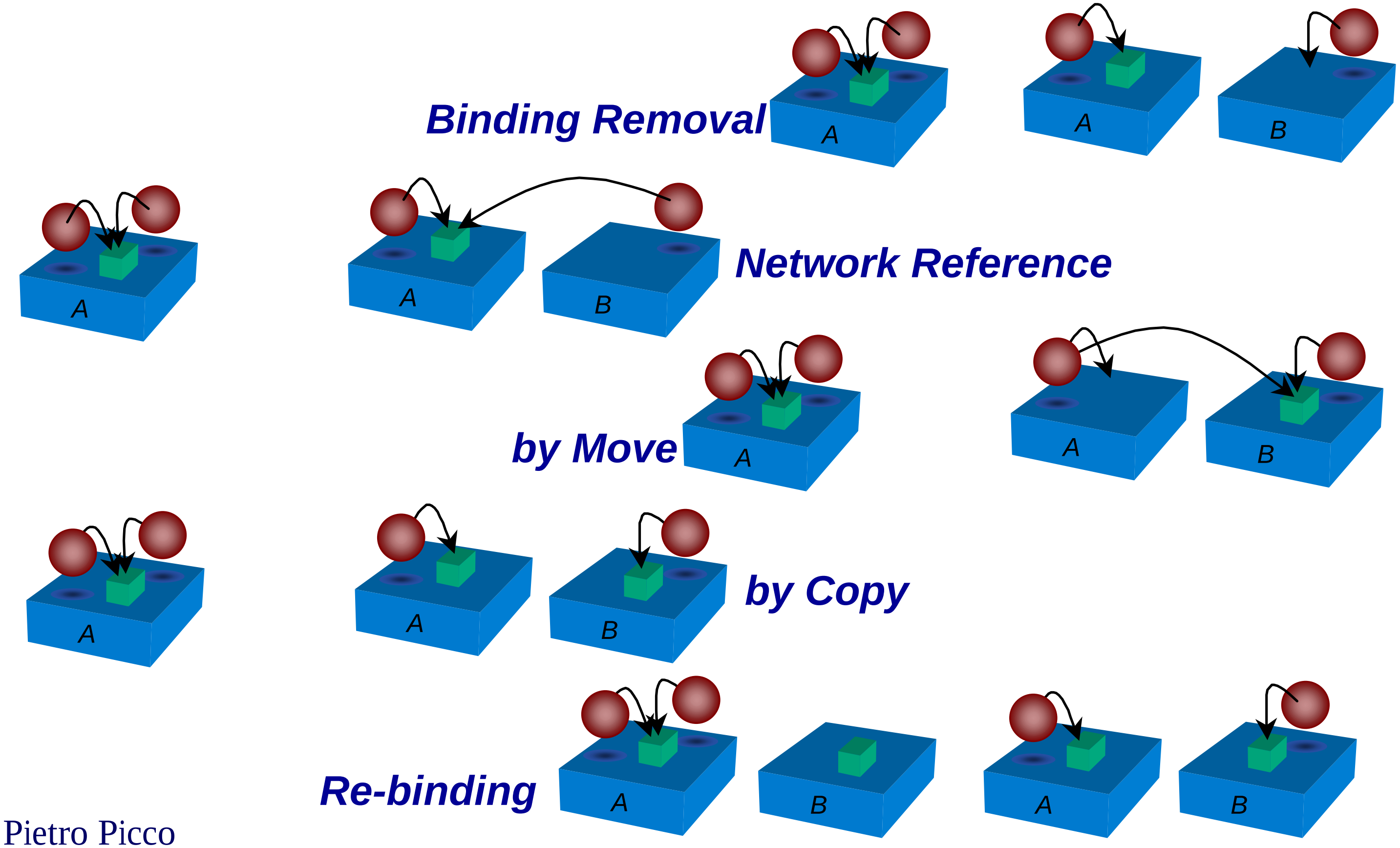
Remote Evaluation

Code on Demand

Mobile Agent

Design Paradigms

- Are these paradigms enough?
- What is the role of data relocation in combination to code relocation?
- Transient interactions are key in mobility



Binding Removal

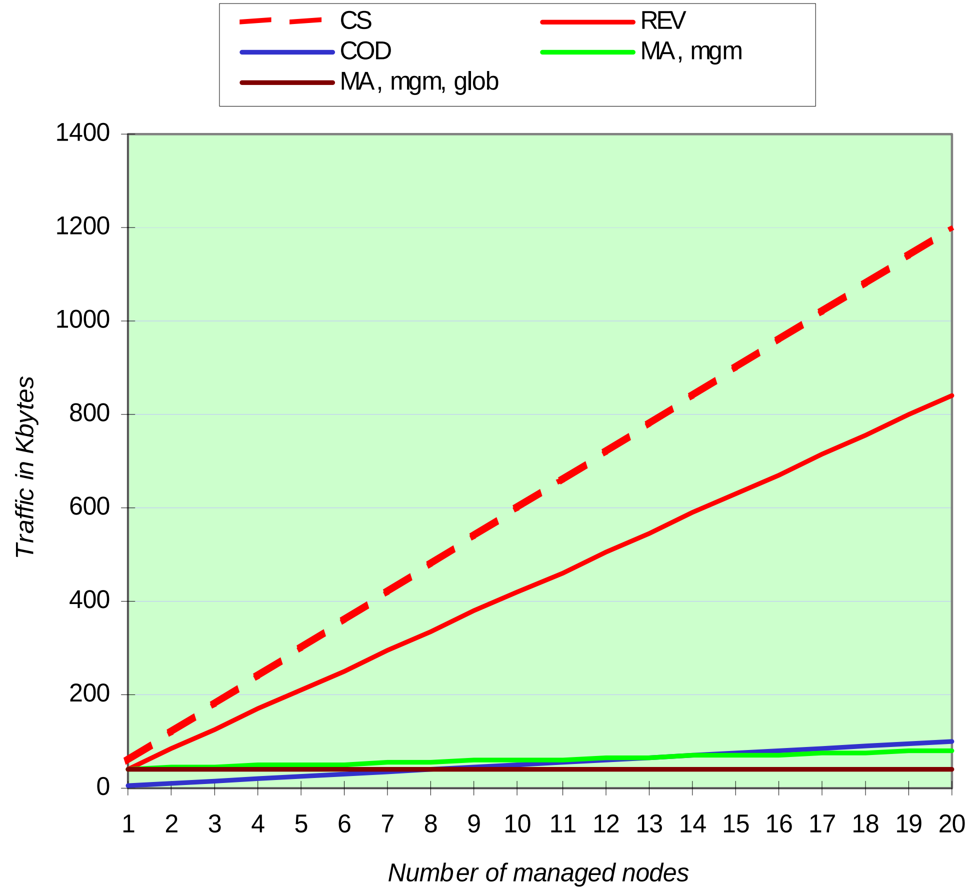
Network Reference

by Move

by Copy

Re-binding

More on Quantitative Arguments



Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications
(Baldi and Picco, ICSE 98)

- Design paradigms help in distinguishing between different forms of mobility, and identifying pros and cons
- They help in formulating general guidelines about when a given style of code relocation should be used
- After all, this is a big fraction of what ***engineering*** is all about

Mobile Code or Mobile Agents?

- Research currently focuses on mobile agent systems, theories, and applications
- Mobile agents are claimed to provide a powerful and unifying abstraction for the development of highly autonomous and decentralized software
  - A hypothesis that largely remains to be demonstrated
- Expressive power comes as a tradeoff for efficiency
  - In many circumstances mobile code appears to be expressive enough, often providing also a more efficient solution

Implementation

- How do we develop applications exploiting code mobility?
  - What are the common pitfalls? What are good technologies to exploit? What is a "good design"? What are the clever tricks you can play?
- For instance…
  - Do we really and deeply understand how class loading and serialization work in Java?
    - And … should we need to?
  - Do we really have a good understanding about the techniques that are necessary to handle a roaming mobile agent?
    - Do we have the right abstractions and mechanisms to do so?
- Is really the field mature for design patterns?

Validation and Testing

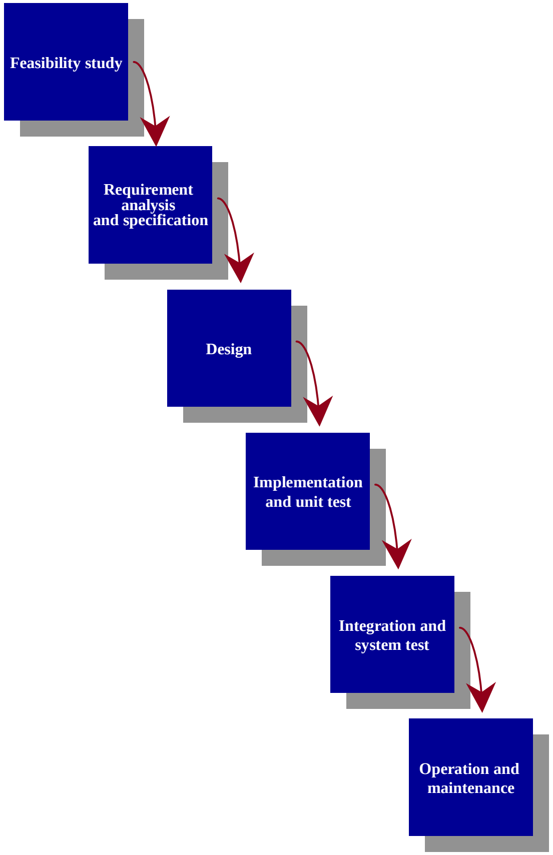
- We really do not know how to test an application that involves mobile code
  - What are the fundamental problems?
  - Is it any different from open systems?
- What new kinds of run-time errors should we look for?
  - Class not found
  - Resource not available
- Is there any way to check their absence statically?
- What techniques and methods can we reuse from software engineering?

Operation and Maintenance

- This is where non-functional requirements strike in
- Performance and scalability
  - Not really addressed by current systems
  - Scalability of mobile agent systems or mobile agent systems for scalability?
- Fault-tolerance
  - Some systems provide “local” fault-tolerance, e.g., through checkpointing
  - Very few provide address the problem of transferring code and state in a fault-tolerant way
  - Fault-tolerant mobile agent systems or mobile agent systems for fault-tolerance?

Software Engineering of Mobile Agent ***Systems***

- Thus far, we discussed the software engineering of applications based on mobile agents
- Nevertheless, mobile agent systems are software themselves: thus, can software engineering help construct better mobile agent systems?


Feasibility study
Requirement analysis and specification
Design
Implementation and unit test
Integration and system test
Operation and maintenance

Requirements

- What are the requirements for a new mobile agent system?
  - Requirements are rarely stated in the documentation
- A general requirement should be to maximize the benefits potentially achievable with mobile agents
  - Unfortunately many systems seem to be designed the other way round
  - Example: plenty of systems require permanent connectivity (e.g., by providing remote class loading as the only code relocation strategy), hence cutting off support for disconnected operations

Design

- Mobile agent systems are, by and large, naively designed
  - Little attention to software engineering principles like modularization, separation of concerns, incrementality, …
- Not only they do not match requirements, but they are often monolithic, proof of concept tools, and it is often not easy to use them in conjunction with other distributed middleware
  - It makes a difference when you get to build real systems for real platforms…
- They are often cluttered with functionality, and provide very little orthogonality
  - What is the core of a mobile agent system? It makes a difference when you get to use them in PDAs or want to maximize efficiency…
- Formal theories of mobility may help identify the essential constructs, and lead to novel language proposals
  - What is the role of coordination?

Implementation

- Software engineering teaches us that programming languages are just tools: different languages are appropriate for different tasks
- So, why everybody is developing mobile agents in Java?
- “Old” vs. “new” systems:

Telescript (1994)

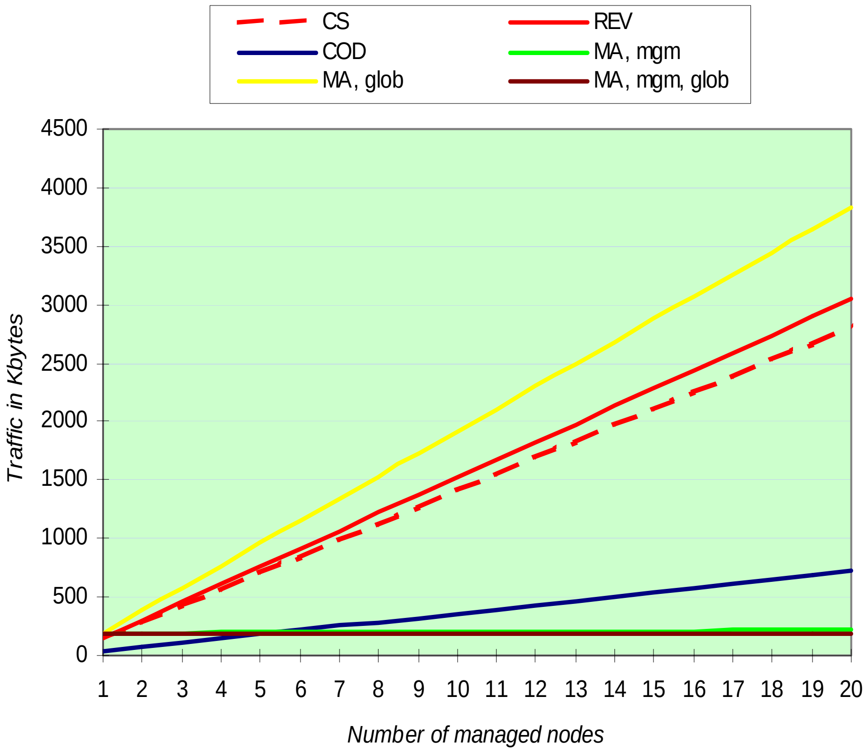
- Strong mobility
- Instance-level member protection
- Resource control
- Ownership ties together security and resources

Aglets—Java-based, so

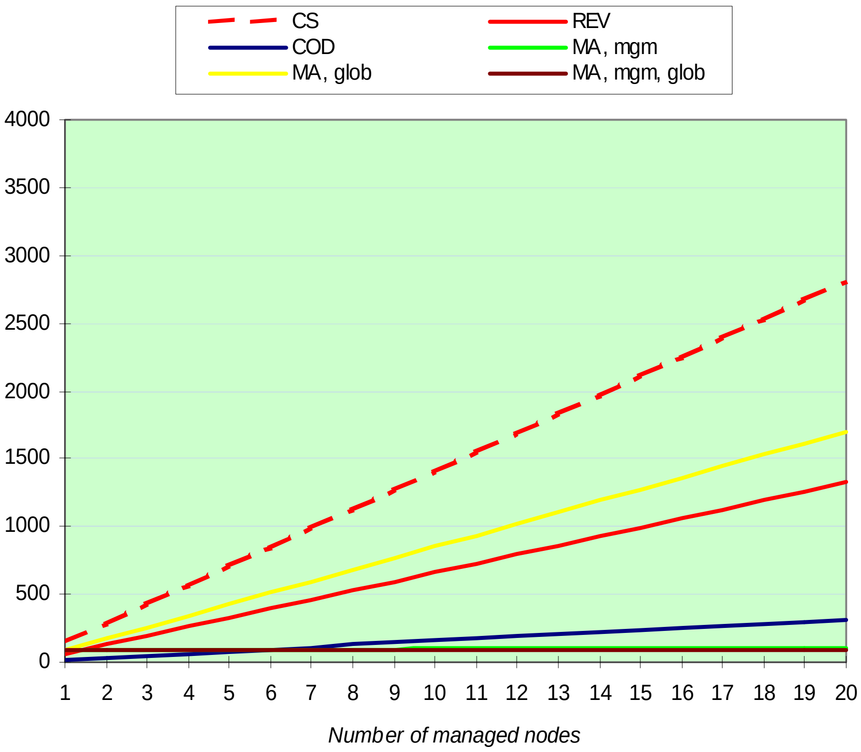
- Weak mobility
- No instance-level member protection
- No resource control
- No real novel construct besides mobile agents

The Language Matters

SNMP vs. ATP



SNMP vs. TCP



- What about scripting languages, or functional languages?
- The power of mobile code is in the architecture, ***not*** in the technology!

Some (Bitter) Considerations (from DWTA'99)

- Many systems embody design criteria that conflict with or hamper the exploitation of mobile code, e.g.:
  - Heavy application-level network protocols
  - Centralized code servers
  - Assumption of permanent connection
  - Migration policy hard-wired in the mechanism
- ***Key open issues:***
  - Lack of validation of abstractions and mechanisms against real world applications
  - Lack of quantitative characterization and comparison of mobile code technologies
- ***The above issues are hampering acceptance and credibility of mobile code research as a whole***

Summing up

- Research on mobile agents is characterized by a disproportionate emphasis on the implementation of mobile agent *systems*, and by a lack of mobile agent *applications*
- This has at least two negative effects:
  - The lack of applications (and of serious studies evaluating them) impairs the whole scientific process of advancing the field;
  - Disregarding the other phases of the software process leads to systems that are often <u>unreliable implementations of a poor design satisfying bad requirements</u>
- Software engineering does not mean just to use UML!
  It provides a general methodological framework, and a corpus of theories, techniques, and methods that can and should be:
  - Applied to the development of mobile agent systems
  - Adapted to encompass the specific requirements posed by mobility
- This holds for other disciplines as well, e.g., distributed systems