II.11

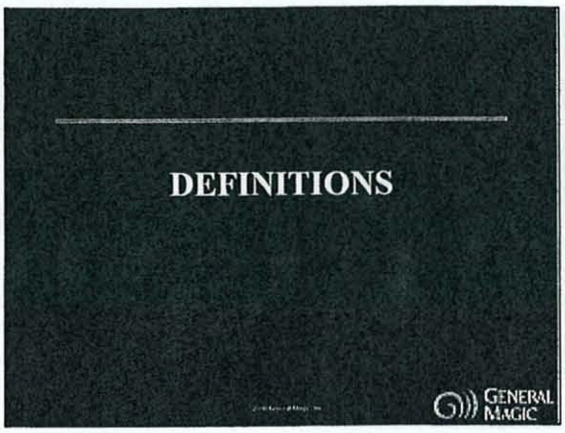
MOBILE AGENT ARCHITECTURES

D B Lange

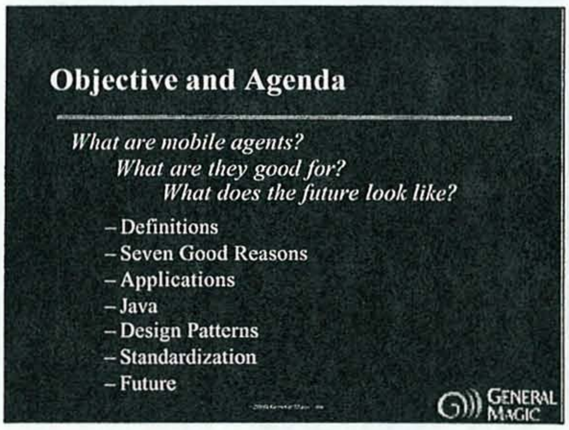
Rapporteur: Dr J A Smith

Mobile Agent Architectures

Danny B. Lange
Chief Technology Officer
General Magic, Inc.



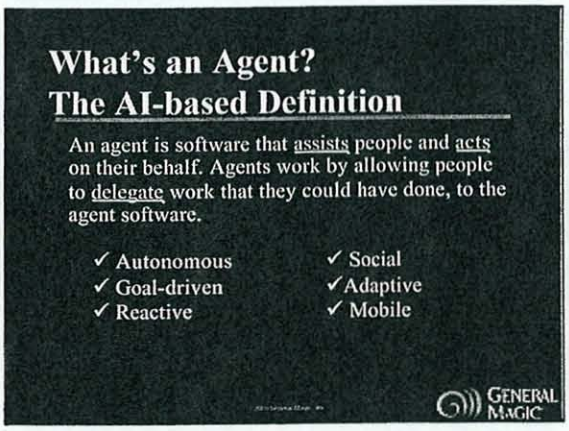
DEFINITIONS



Objective and Agenda

What are mobile agents?
What are they good for?
What does the future look like?

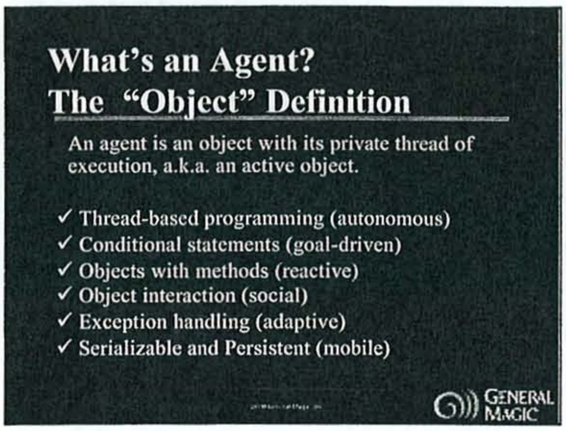
- Definitions
- Seven Good Reasons
- Applications
- Java
- Design Patterns
- Standardization
- Future



What's an Agent? The AI-based Definition

An agent is software that assists people and acts on their behalf. Agents work by allowing people to delegate work that they could have done, to the agent software.

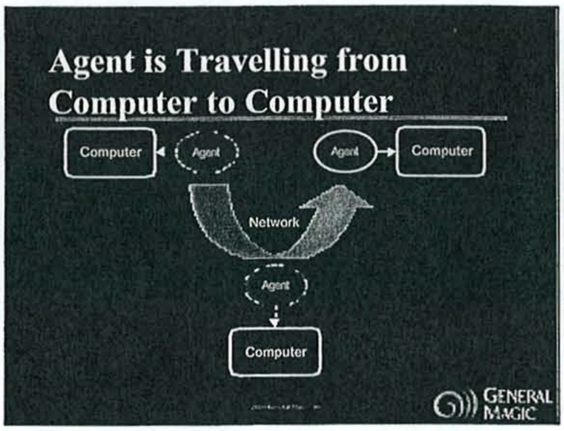
✓ Autonomous	✓ Social
✓ Goal-driven	✓Adaptive
✓ Reactive	✓ Mobile



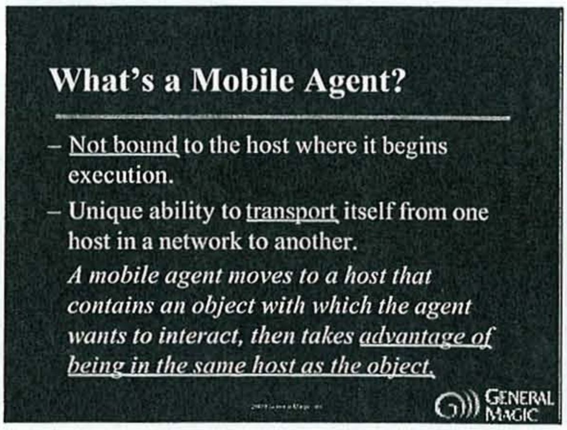
What's an Agent? The "Object" Definition

An agent is an object with its private thread of execution, a.k.a. an active object.

- ✓ Thread-based programming (autonomous)
- ✓ Conditional statements (goal-driven)
- ✓ Objects with methods (reactive)
- ✓ Object interaction (social)
- ✓ Exception handling (adaptive)
- ✓ Serializable and Persistent (mobile)



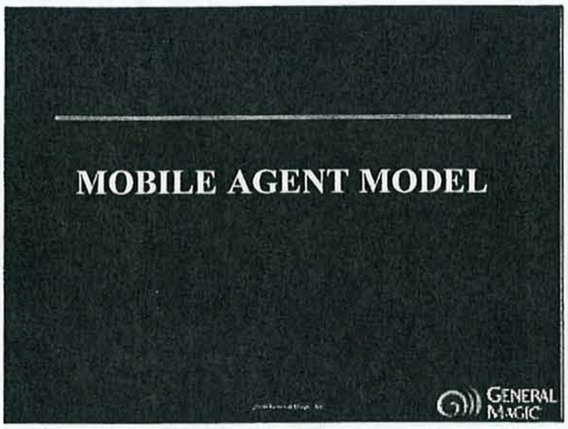
Agent is Travelling from Computer to Computer



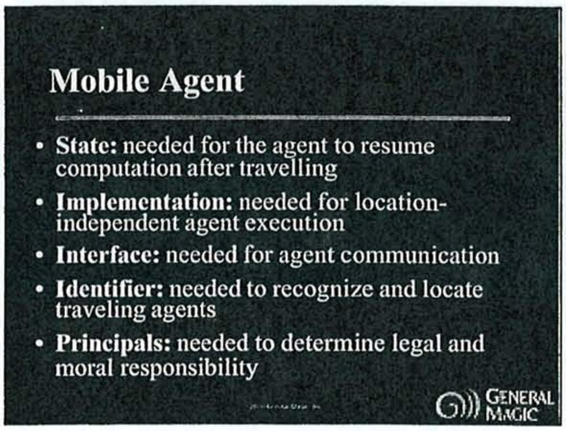
What's a Mobile Agent?

- Not bound to the host where it begins execution.
- Unique ability to transport itself from one host in a network to another.

A mobile agent moves to a host that contains an object with which the agent wants to interact, then takes advantage of being in the same host as the object.



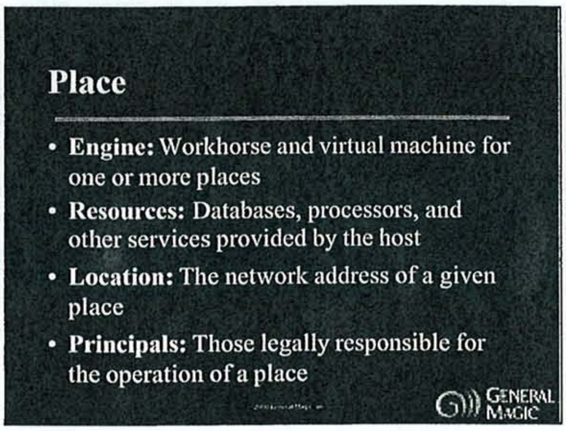
MOBILE AGENT MODEL



Mobile Agent

- **State:** needed for the agent to resume computation after travelling
- **Implementation:** needed for location-independent agent execution
- **Interface:** needed for agent communication
- **Identifier:** needed to recognize and locate traveling agents
- **Principals:** needed to determine legal and moral responsibility

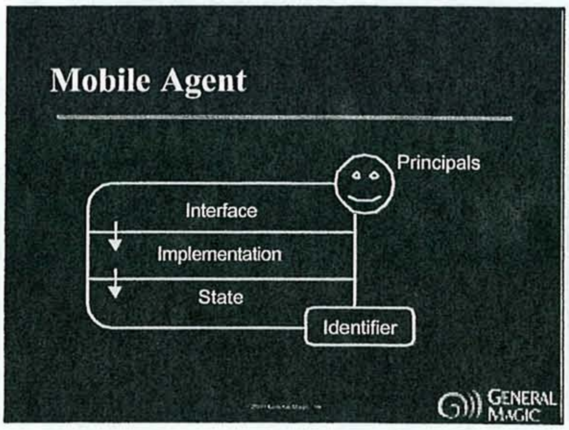
GENERAL MAGIC



Place

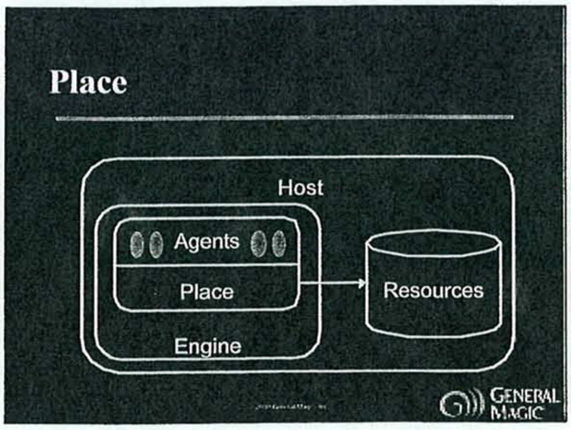
- **Engine:** Workhorse and virtual machine for one or more places
- **Resources:** Databases, processors, and other services provided by the host
- **Location:** The network address of a given place
- **Principals:** Those legally responsible for the operation of a place

GENERAL MAGIC



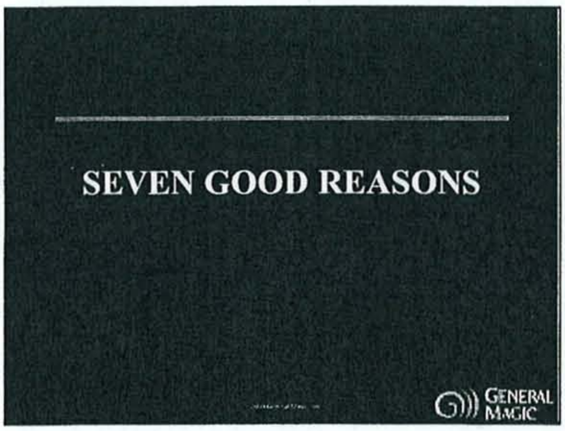
Mobile Agent

GENERAL MAGIC

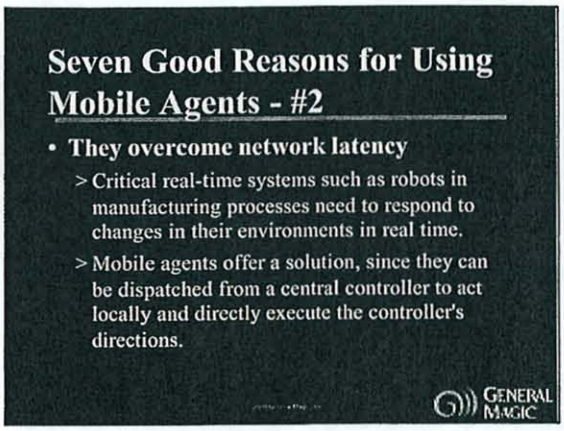


Place

GENERAL MAGIC

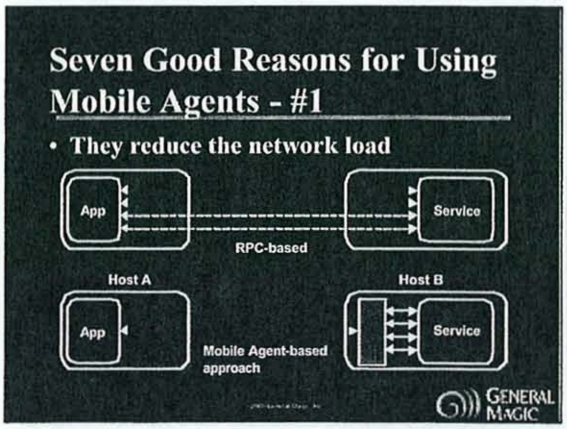


SEVEN GOOD REASONS



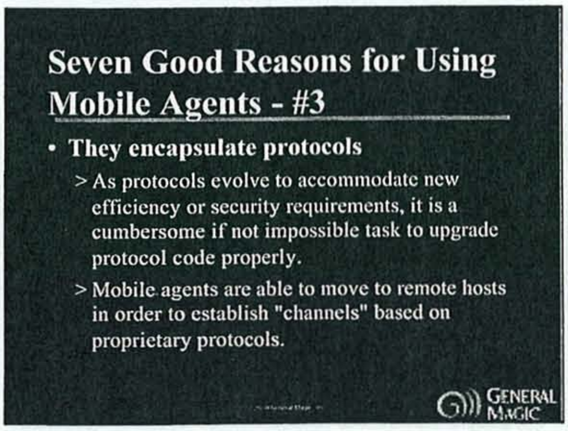
Seven Good Reasons for Using Mobile Agents - #2

- They overcome network latency
  - > Critical real-time systems such as robots in manufacturing processes need to respond to changes in their environments in real time.
  - > Mobile agents offer a solution, since they can be dispatched from a central controller to act locally and directly execute the controller's directions.



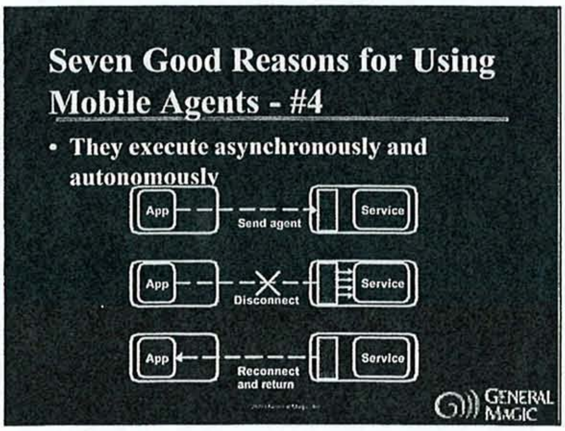
Seven Good Reasons for Using Mobile Agents - #1

- They reduce the network load



Seven Good Reasons for Using Mobile Agents - #3

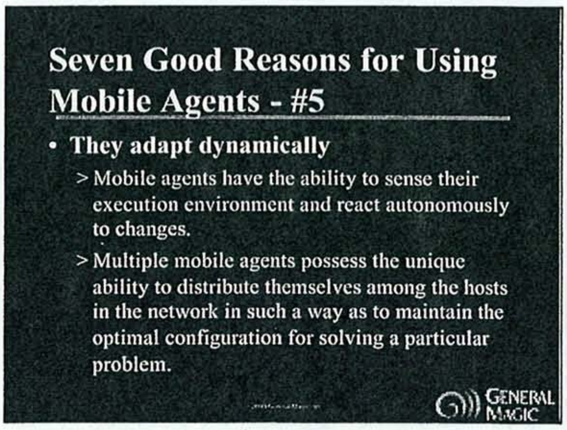
- They encapsulate protocols
  - > As protocols evolve to accommodate new efficiency or security requirements, it is a cumbersome if not impossible task to upgrade protocol code properly.
  - > Mobile agents are able to move to remote hosts in order to establish "channels" based on proprietary protocols.



Seven Good Reasons for Using Mobile Agents - #4

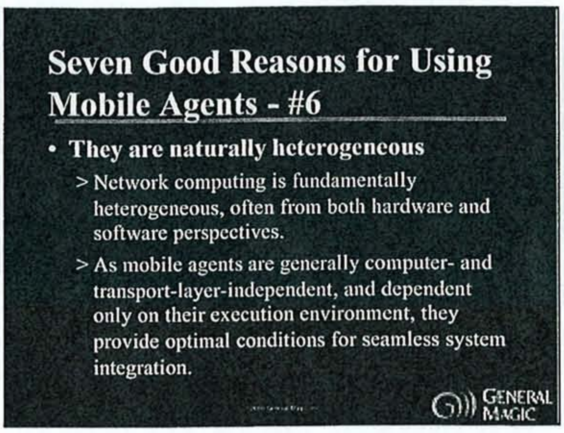
- They execute asynchronously and autonomously

App
Send agent
Service
App
Disconnect
Service
App
Reconnect and return
Service



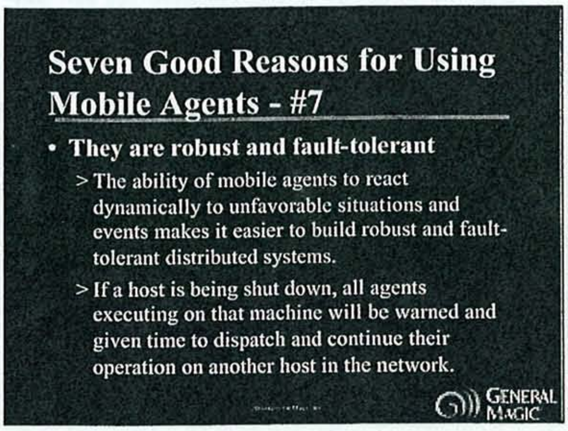
Seven Good Reasons for Using Mobile Agents - #5

- They adapt dynamically
  - > Mobile agents have the ability to sense their execution environment and react autonomously to changes.
  - > Multiple mobile agents possess the unique ability to distribute themselves among the hosts in the network in such a way as to maintain the optimal configuration for solving a particular problem.



Seven Good Reasons for Using Mobile Agents - #6

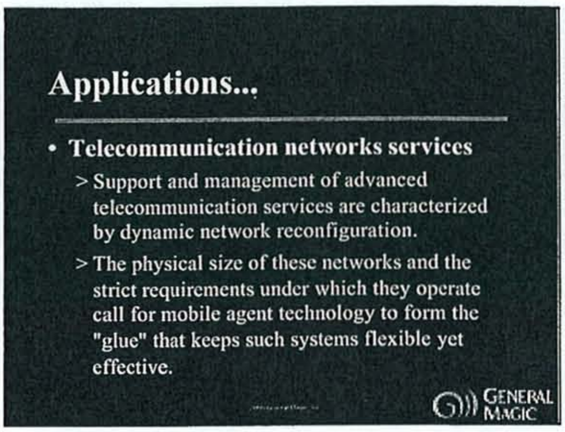
- They are naturally heterogeneous
  - > Network computing is fundamentally heterogeneous, often from both hardware and software perspectives.
  - > As mobile agents are generally computer- and transport-layer-independent, and dependent only on their execution environment, they provide optimal conditions for seamless system integration.



Seven Good Reasons for Using Mobile Agents - #7

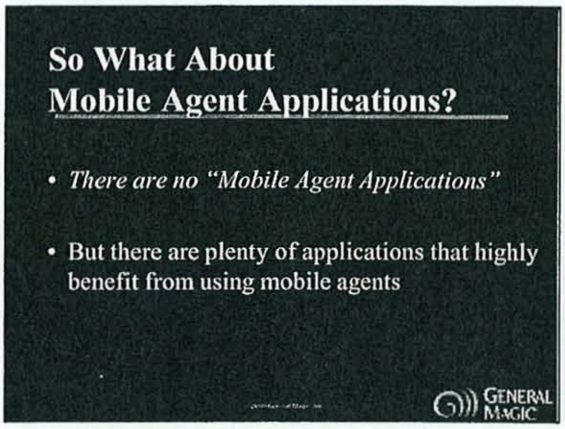
- They are robust and fault-tolerant
  - > The ability of mobile agents to react dynamically to unfavorable situations and events makes it easier to build robust and fault-tolerant distributed systems.
  - > If a host is being shut down, all agents executing on that machine will be warned and given time to dispatch and continue their operation on another host in the network.

APPLICATIONS



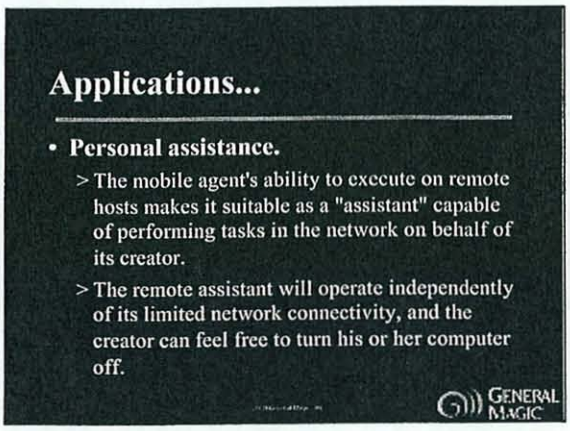
Applications...

- **Telecommunication networks services**
  - > Support and management of advanced telecommunication services are characterized by dynamic network reconfiguration.
  - > The physical size of these networks and the strict requirements under which they operate call for mobile agent technology to form the "glue" that keeps such systems flexible yet effective.



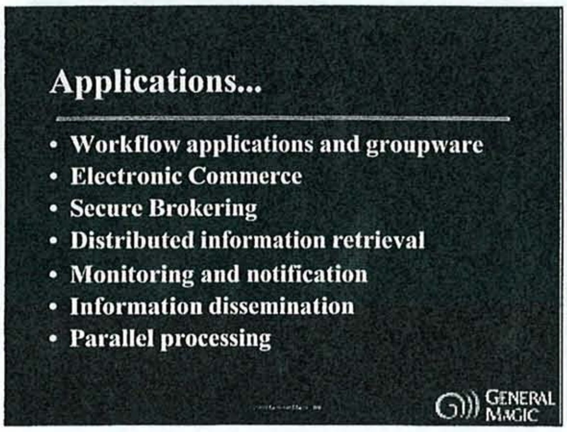
So What About Mobile Agent Applications?

- *There are no "Mobile Agent Applications"*
- But there are plenty of applications that highly benefit from using mobile agents



Applications...

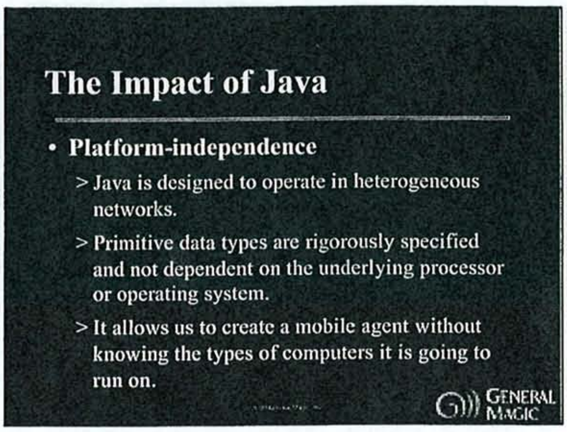
- **Personal assistance.**
  - > The mobile agent's ability to execute on remote hosts makes it suitable as a "assistant" capable of performing tasks in the network on behalf of its creator.
  - > The remote assistant will operate independently of its limited network connectivity, and the creator can feel free to turn his or her computer off.



Applications...

- Workflow applications and groupware
- Electronic Commerce
- Secure Brokering
- Distributed information retrieval
- Monitoring and notification
- Information dissemination
- Parallel processing

GENERAL MAGIC



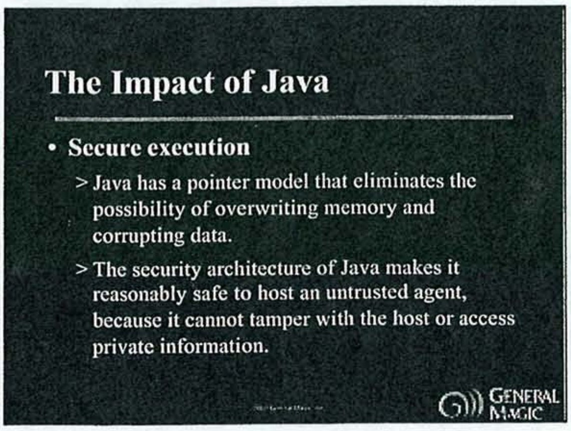
The Impact of Java

- Platform-independence
  - > Java is designed to operate in heterogeneous networks.
  - > Primitive data types are rigorously specified and not dependent on the underlying processor or operating system.
  - > It allows us to create a mobile agent without knowing the types of computers it is going to run on.

GENERAL MAGIC

JAVA

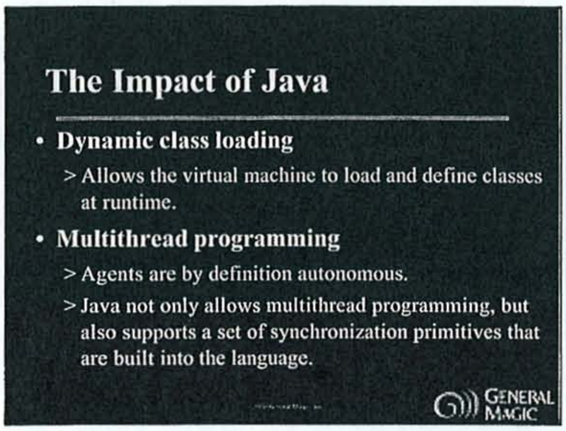
GENERAL MAGIC



The Impact of Java

- Secure execution
  - > Java has a pointer model that eliminates the possibility of overwriting memory and corrupting data.
  - > The security architecture of Java makes it reasonably safe to host an untrusted agent, because it cannot tamper with the host or access private information.

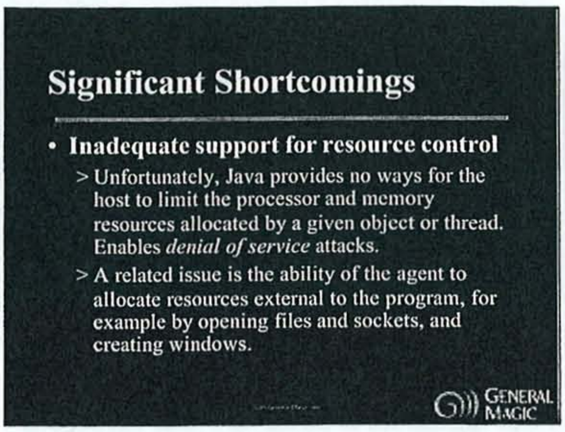
GENERAL MAGIC



The Impact of Java

- **Dynamic class loading**
  - > Allows the virtual machine to load and define classes at runtime.
- **Multithread programming**
  - > Agents are by definition autonomous.
  - > Java not only allows multithread programming, but also supports a set of synchronization primitives that are built into the language.

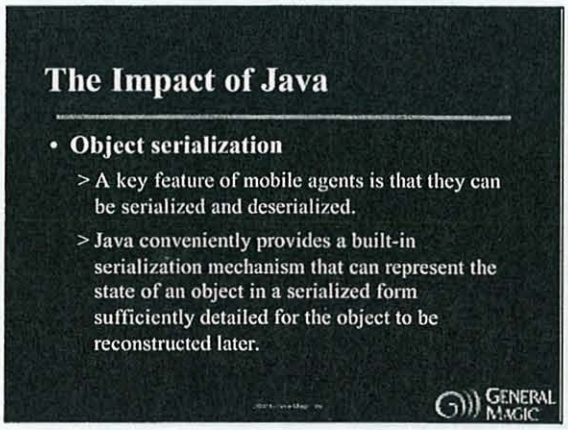
GENERAL MAGIC



Significant Shortcomings

- **Inadequate support for resource control**
  - > Unfortunately, Java provides no ways for the host to limit the processor and memory resources allocated by a given object or thread. Enables *denial of service* attacks.
  - > A related issue is the ability of the agent to allocate resources external to the program, for example by opening files and sockets, and creating windows.

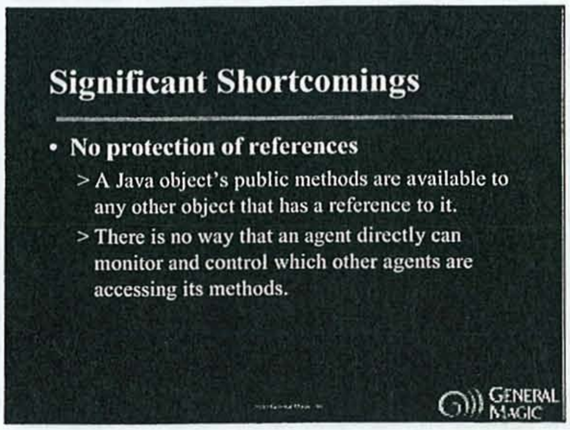
GENERAL MAGIC



The Impact of Java

- **Object serialization**
  - > A key feature of mobile agents is that they can be serialized and deserialized.
  - > Java conveniently provides a built-in serialization mechanism that can represent the state of an object in a serialized form sufficiently detailed for the object to be reconstructed later.

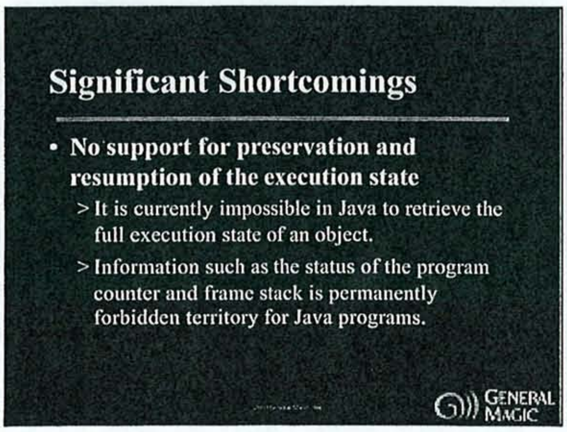
GENERAL MAGIC



Significant Shortcomings

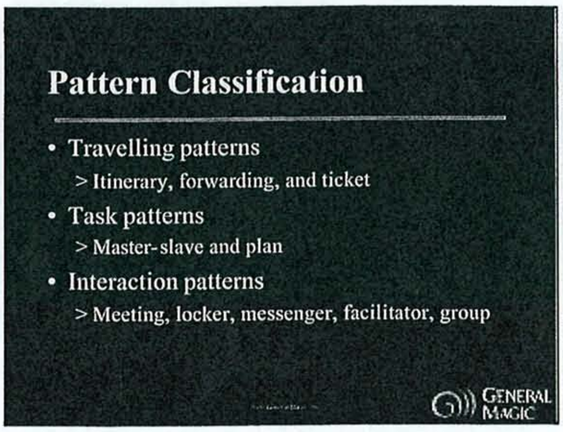
- **No protection of references**
  - > A Java object's public methods are available to any other object that has a reference to it.
  - > There is no way that an agent directly can monitor and control which other agents are accessing its methods.

GENERAL MAGIC



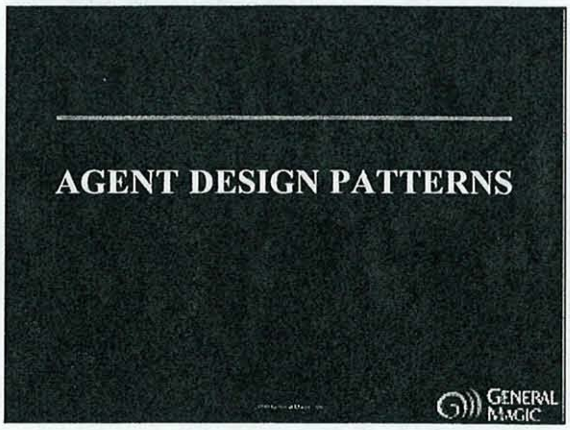
Significant Shortcomings

- **No support for preservation and resumption of the execution state**
  - > It is currently impossible in Java to retrieve the full execution state of an object.
  - > Information such as the status of the program counter and frame stack is permanently forbidden territory for Java programs.

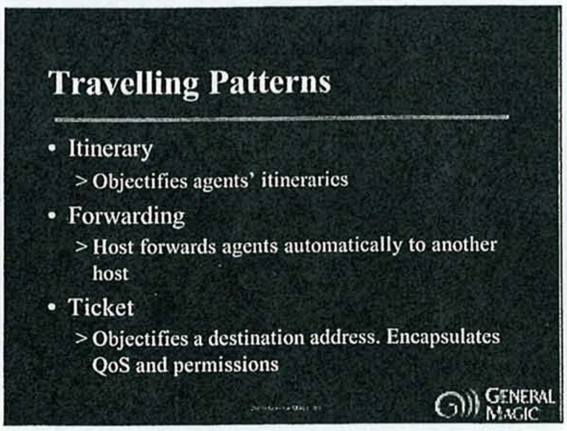


Pattern Classification

- Travelling patterns
  - > Itinerary, forwarding, and ticket
- Task patterns
  - > Master-slave and plan
- Interaction patterns
  - > Meeting, locker, messenger, facilitator, group

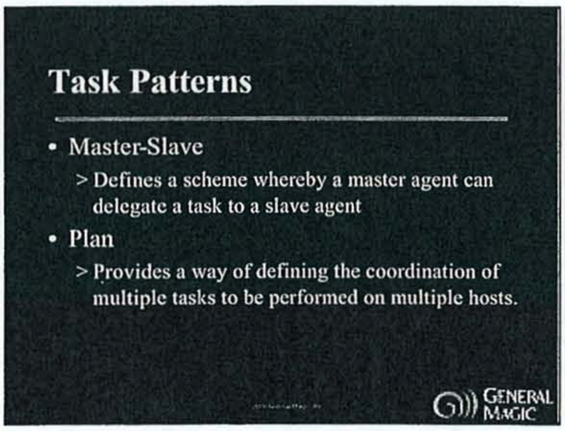


AGENT DESIGN PATTERNS



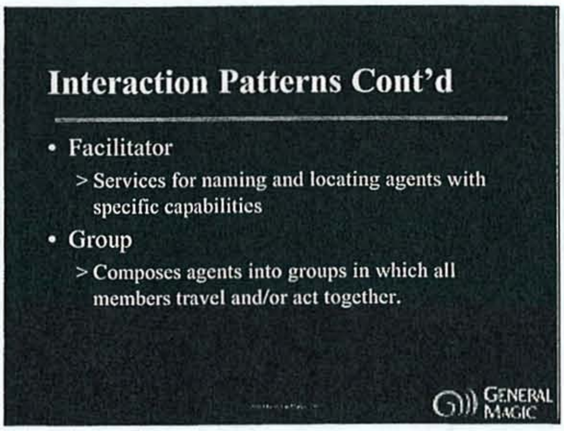
Travelling Patterns

- Itinerary
  - > Objectifies agents' itineraries
- Forwarding
  - > Host forwards agents automatically to another host
- Ticket
  - > Objectifies a destination address. Encapsulates QoS and permissions



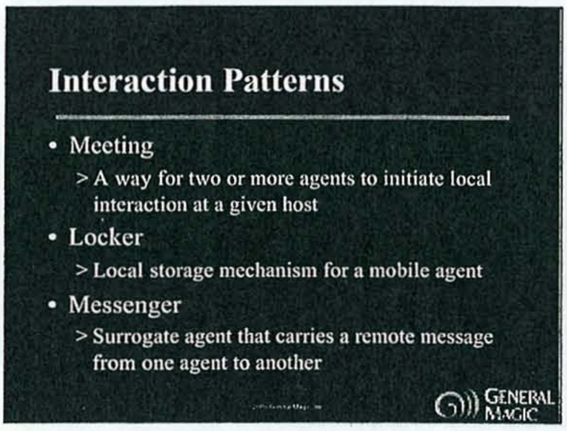
Task Patterns

- Master-Slave
  > Defines a scheme whereby a master agent can delegate a task to a slave agent
- Plan
  > Provides a way of defining the coordination of multiple tasks to be performed on multiple hosts.



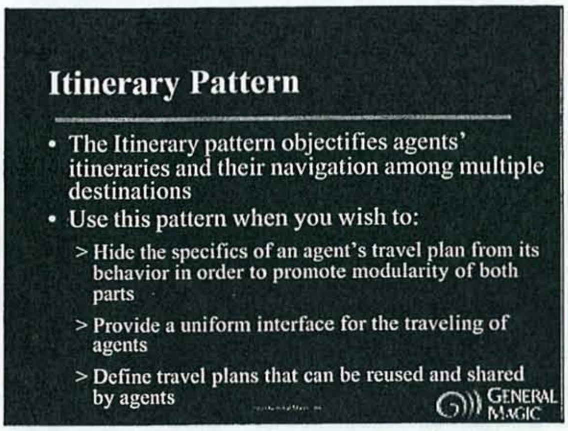
Interaction Patterns Cont'd

- Facilitator
  > Services for naming and locating agents with specific capabilities
- Group
  > Composes agents into groups in which all members travel and/or act together.



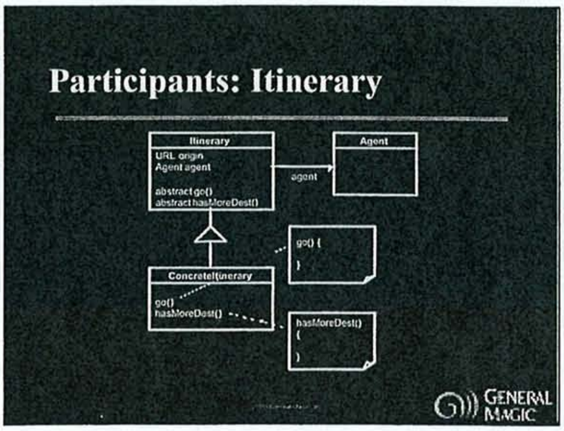
Interaction Patterns

- Meeting
  > A way for two or more agents to initiate local interaction at a given host
- Locker
  > Local storage mechanism for a mobile agent
- Messenger
  > Surrogate agent that carries a remote message from one agent to another



Itinerary Pattern

- The Itinerary pattern objectifies agents' itineraries and their navigation among multiple destinations
- Use this pattern when you wish to:
  > Hide the specifics of an agent's travel plan from its behavior in order to promote modularity of both parts
  > Provide a uniform interface for the traveling of agents
  > Define travel plans that can be reused and shared by agents



Participants: Itinerary

Itinerary

URL origin
Agent agent

abstract go()
abstract hasMoreDest()

agent

Agent

go() {

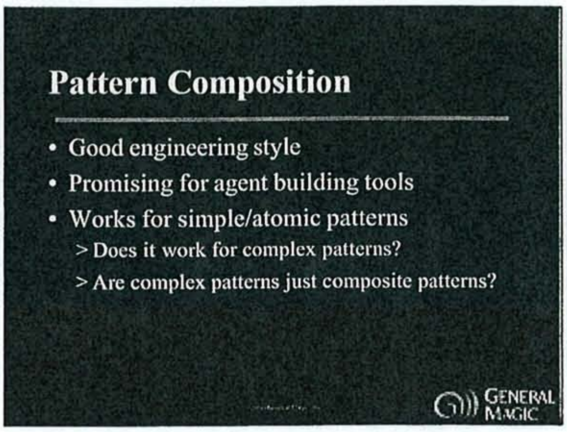
}

ConcreteItinerary

go()
hasMoreDest()

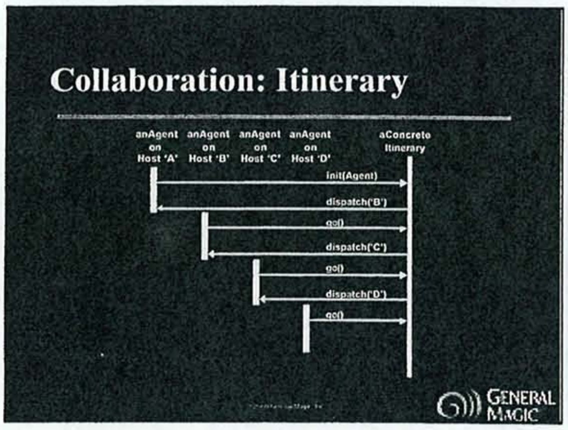
hasMoreDest()
{

}



Pattern Composition

- Good engineering style
- Promising for agent building tools
- Works for simple/atomic patterns
  - > Does it work for complex patterns?
  - > Are complex patterns just composite patterns?



Collaboration: Itinerary

anAgent on Host 'A'

anAgent on Host 'B'

anAgent on Host 'C'

anAgent on Host 'D'

aConcrete Itinerary

init(Agent)

dispatch('B')

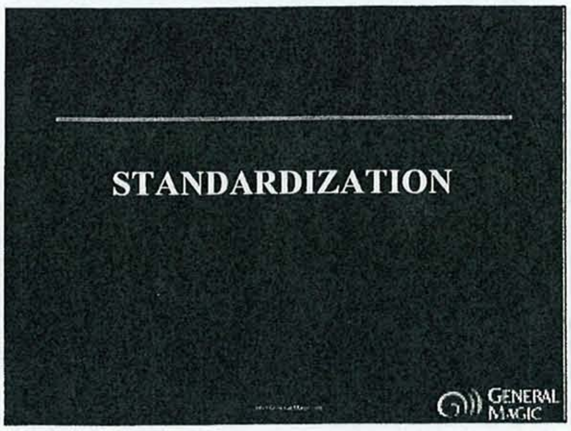
go()

dispatch('C')

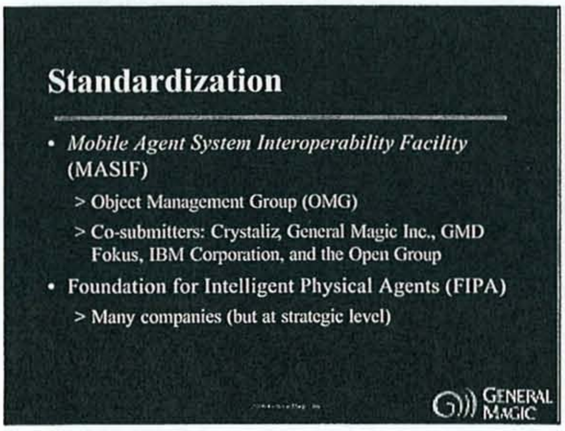
go()

dispatch('D')

go()

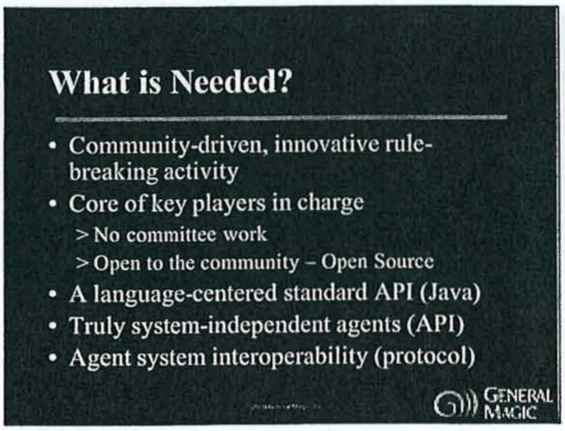


STANDARDIZATION



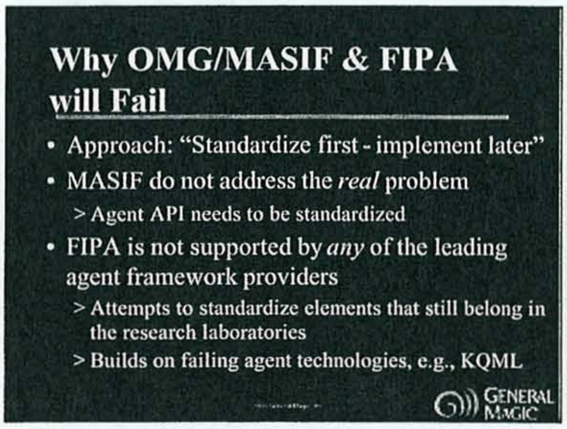
Standardization

- *Mobile Agent System Interoperability Facility* (MASIF)
  - > Object Management Group (OMG)
  - > Co-submitters: Crystaliz, General Magic Inc., GMD Fokus, IBM Corporation, and the Open Group
- Foundation for Intelligent Physical Agents (FIPA)
  - > Many companies (but at strategic level)



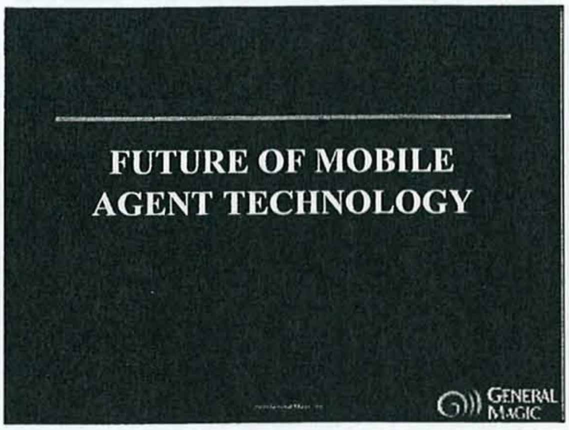
What is Needed?

- Community-driven, innovative rule-breaking activity
- Core of key players in charge
  - > No committee work
  - > Open to the community – Open Source
- A language-centered standard API (Java)
- Truly system-independent agents (API)
- Agent system interoperability (protocol)

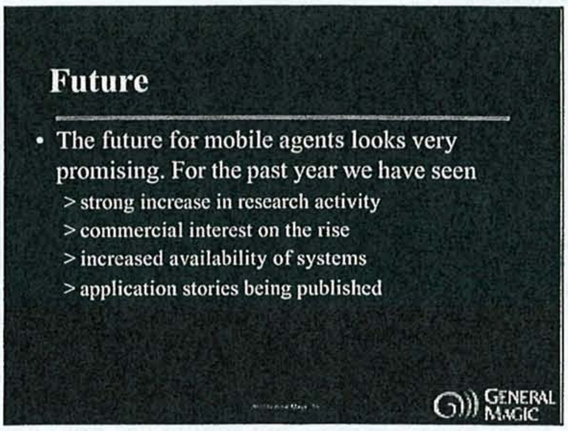


Why OMG/MASIF & FIPA will Fail

- Approach: "Standardize first - implement later"
- MASIF do not address the *real* problem
  - > Agent API needs to be standardized
- FIPA is not supported by *any* of the leading agent framework providers
  - > Attempts to standardize elements that still belong in the research laboratories
  - > Builds on failing agent technologies, e.g., KQML



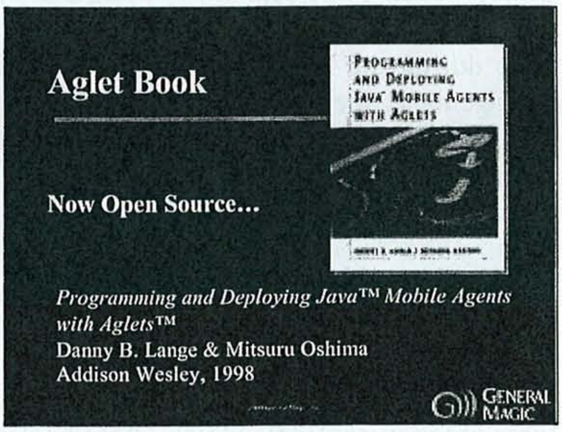
FUTURE OF MOBILE AGENT TECHNOLOGY



Future

- The future for mobile agents looks very promising. For the past year we have seen
  - > strong increase in research activity
  - > commercial interest on the rise
  - > increased availability of systems
  - > application stories being published

GENERAL MAGIC



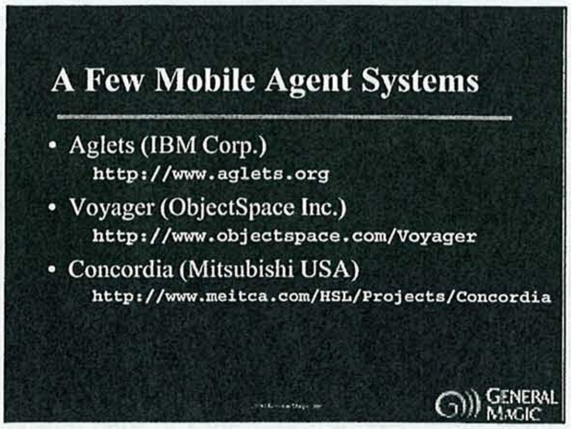
Aglet Book

PROGRAMMING AND DEPLOYING JAVA MOBILE AGENTS WITH AGLETS

Now Open Source...

Programming and Deploying Java™ Mobile Agents with Aglets™
Danny B. Lange & Mitsuru Oshima
Addison Wesley, 1998

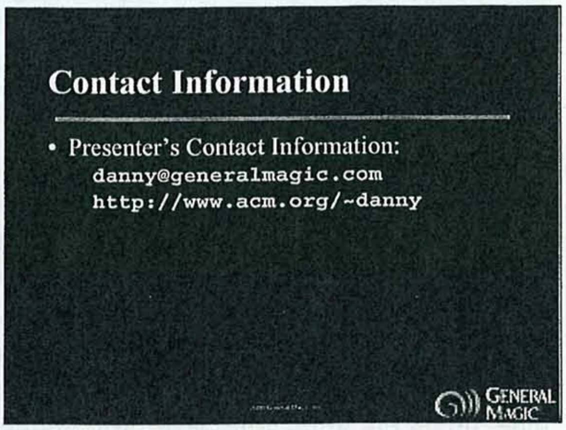
GENERAL MAGIC



A Few Mobile Agent Systems

- Aglets (IBM Corp.)
  `http://www.aglets.org`
- Voyager (ObjectSpace Inc.)
  `http://www.objectspace.com/Voyager`
- Concordia (Mitsubishi USA)
  `http://www.meitca.com/HSL/Projects/Concordia`

GENERAL MAGIC



Contact Information

- Presenter's Contact Information:
  `danny@generalmagic.com`
  `http://www.acm.org/~danny`

GENERAL MAGIC

DISCUSSION

Rapporteur: Dr J A Smith

During initial presentation of Aglets, Professor Sloman asked if there was a notation of contract. Dr Lange replied that there wasn't; that the intention was to achieve a quick prototype, all-be-it with a list of items to be fixed later.

Also during the main presentation Dr Lange suggested that an as yet unfulfilled purpose for agents is the establishment of a special purpose protocol above the basic transport layer. Mr Peine suggested that Jini operates in this way, and Dr Waldo suggested that RMI is another example.

Starting the discussion period after the talk, Professor Randall asked what is the future of agent technology. Dr Lange suggested it didn't look good; specifically that five years (since the arrival of agent technology) is too long for there to have been no breakthrough, i.e. significant application. Dr Thomsen suggested that Erikson are using agent technology in telecommunications switches, but perhaps without great publicity. Dr Lange replied that there are many systems which might be described as agent systems but which are not agent systems in reality. He described how typically such applications could make use of an agent framework, and cited as examples the management of hubs by Cisco and remote software updating by Marimba. He stressed however that there is nothing so far which is really driven by agent technology.

Professor Sloman suggested that perhaps the idea of "mobile" agents is misplaced; more useful perhaps is that of "remote" agents. Dr Lange concurred basically though thought that one possible application for a genuine mobile agent might the encapsulation of the state belonging to a person who is moving around, and needs to take his state with him. He added that a Microsoft application which monitors for possible suitable updates is really an example of a remote agent; likewise the Merimba application referred to earlier. Voice Agents

Following the first audio example during the main presentation, Professor Sloman asked what happens in the event of a failure of understanding. Dr Lange replied that the system speaker would be apologetic in tone. Dr Thomsen asked if it would be possible to choose the personality of the speaker and Dr Lange replied that it would soon be possible.

Following presentation of the On-Star service, Professor Jones remarked that of the component services described, share update etc, a particularly interesting one is email since this contains quite arbitrary dialogue. Dr Lange explained that the mechanism employed is voice messages, so that the arbitrary dialogue is contained in packets which are handled within scripted dialogues.

Professor Sloman suggested that if the system failed to understand and went ahead with some action it would be necessary to interrupt. Dr Lange replied that the policy is always to support a user controlled roll-back, for instance, rather than actually deleting a message on request, moving the message into a delete folder.

Dr Holt asked how easy it is to use voice to create a script. Dr Lange replied that this wouldn't be the approach used; describing a demonstration by Microsoft of a possible interaction with a word processor, he said he could see no broad use for such techniques. Rather, while testing scripts in audio interaction, the linguists at General Magic would enter them using conventional keyboard input.

Following on from his earlier comments, Professor Sloman pointed out that it would sometimes be desirable to interrupt, e.g. if the user realises that a misunderstanding has occurred. Dr Lange replied that the audio examples presented earlier did include such interruptions.

Professor Vogt commented on the instances where the system performs transactions, asking how the interface, e.g. to a trader, is actually handled. Dr Lange replied that the trader has a web site and that the interaction works in the same way as a conventional web site interaction.

Dr Watson asked what is the situation with regard to liability in interactions with stock trading. Dr Lange replied that currently NASDAQ don't allow this sort of voice based trading; continuing to say that rules need to be defined.

Professor Jones asked about personalization of the system, suggesting for instance that he could define his maximum available amount and then the system could recognise a misunderstanding if the amount to be spent exceeded this amount. Dr Lange acknowledged a research interest in home scripting and agreed that scripts developed by General Magic would be developed with a personal computer, being able to tailor the system is a reasonable wish, but suggested that the auto companies would be interested at present in a basic pre-defined system.

Dr Wood commented that the choice of an ultra-thin client is ideal for the voice exchanges with the driver, but suggested that a passenger in the back seat might appreciate a screen based interaction. Dr Lange replied that voice is just one channel being developed, and that other channels share a common server-based architecture.

Dr Elliott remarked on the difference between the voice agents scripted in XML and the earlier perception of agents. Dr Lange replied that the idea was to borrow the most suitable notions used in agent work and exploit them in a new context.