Programming Languages for Mobile Code

Anna Philippou

University of Pennsylvania

Bibliography

- *Programming Languages for Mobile Code*, T. Thorn.
- *Understanding Code Mobility*, Fuggetta, Picco, and Vigna.
- *The Java Language Environment, a White Paper*, Gosling and McGilton.
- *A language with Distributed Scope*, L. Cardelli.

Synopsis

- Mobile code, aims and concerns
- Safety and Security Issues
- Mobility Issues
- Java, Limbo, O'Caml, Safe-Tcl
- Obliq, Telescript
- Comparison
- Review and perspectives

Code Mobility

- The capability to change dynamically the bindings between code fragments and the location where they are executed.

Motivations

- Efficiency
- Simplicity and Flexibility
- Storage

Examples

- Postscript
- Database technology
- Documents with embedded executable contents (e.g. email, www)
- The UNIX `rsh' command

Programming Language Concerns

- Portability
- Safety
- Efficiency
- Security
  - Confidentiality
  - Integrity
  - Availability
  - Authenticity

Dealing with Safety and Security

- The Communication Level
  - robust protocol implementation to withstand a faulty or malicious communication partner
  - cryptographic techniques to guarantee confidentiality, security and authentication
- The Operating-System Level
  - Hardware memory protection
  - Control of access to information and communication channels (e.g. limits on resources)

Safety and Security

- The Abstract-Machine Level
  - memory protection
- The Programming Language Level
  - typing, restricted pointers, automatic memory management
  - scope and access rules
  - compilation and type-checking before shipping
    - trust of author, or compiler
    - use of intermediate language

Mobility in Programming Languages

- **Weak Mobility**: the ability of a PL to allow an execution unit in a site to be bound dynamically to code coming from a different site. (Java, Tcl, Facile)
- **Strong Mobility**: the ability of a PL to allow execution units to move code and execution state between different sites. (Telescript, Agent Tcl)

Programming Languages

Language	OO	Concurrency	Mobility	Safety	Security model
Java	Yes	Yes	Weak	Yes	PL
O'Caml	Yes	Yes	Weak	Yes	PL
Limbo		Yes	Weak	Yes	OS
Obliq	Yes	Yes	Strong	Yes	PL
Telescript	Yes	Yes	Strong	Yes	PL
Safe-Tcl			Weak	Yes	OS

Java

- A class-based object-oriented language based on the **applet model**.
- Applets are small programs that are downloaded and executed when encountered by an application.

The Java Language

- Simplified version of C++ augmented with
  - automatic memory management
  - built-in array and string types
  - exception handling
  - threads for concurrency and serialized methods using mutual exclusion mechanisms

Java Entities

- Interface Types: collections of abstract methods and constants with their associated types
- Packages: extensible collections of classes and interface definitions

Scoping and Visibility Rules

- A class can be defined as
  - final
  - abstract
  - private
- Attributes can be defined as
  - private
  - default
  - protected
  - public

Java Security

- Local resources are protected by the scope and visibility rules.
- The Security Manager
  - a centralized security monitor
  - defined as an *abstract* type
  - with all its methods declared as *final*
- The Abstract Machine checks and verifies code received through the network

Java security

- “Safety” bugs have been discovered
- No way to protect user-defined libraries

Limbo

- Inferno (Lucent Technologies, 1996) is a network operating system aimed mostly at media providers and telecommunication companies, designed to achieve
  - portability, versatility, “economical” implementation
- Applications for Inferno are written in Limbo.

Limbo...

- Is a "safe" language.
- Is inspired by C. Additionally, it includes
  - declarations
  - abstract data types
  - first-class modules
  - first-class channels
  - automatic memory management
  - threads

Safety and Security

- Limbo programs are compiled to a RISC-like abstract machine, Dis.
- Rich library of standard modules (e.g. for network communication, secure and encrypted communication, and graphics).
- Safety is achieved by restrictions in the language (guaranteed by trusted authorities)
- Security is achieved by treating all resources as files

Objective Caml

- Developed and used as a language for mobile code for the MMM web browser (INRIA)
- A functional language in the ML tradition
- It includes imperative features including references and assignment and a class-based object system
- It offers the possibility of dynamically linking applets.

O'Caml

- Polymorphic, static typing
- Higher-order functions
- Powerful module system
- Concurrency through threads and mutexes.
- Class-based object orientation
  - A class can be declared as **virtual,** and **closed**
  - An attribute can be declared as **private**

O'Caml Security

- MMM applets may only use *safe variants of standard libraries*
- Safety is ensured by type checking
- Object-code is associated with a cryptographic signature
- Need for trusted compilation sites

Safe-Tcl

- Proposed to support active email
- Based on Tcl, it is a procedural script language.
- No mobility or communication mechanisms
- Simple scoping rules (local and global)

Safety and Security

- Dangerous/general Tcl primitives were replaced by more specific ones.
- Aim: to protect the recipient's computational environment.
- Twin interpreter scheme, consisting of a trusted (Tcl) interpreter and an untrusted (Safe-Tcl) interpreter.

Obliq

- Lexically-scoped, dynamically-typed
- Object-based
- Computations are network transparent (distribution is managed explicitly at the language level)
- Free variables of transmitted computation can refer to values at the origin site.

Obliq objects

- An object is a collection of attributes: e.g

  $\{x_1 \Rightarrow 3, x_2 \Rightarrow \text{meth}(y, y_1, \ldots, y_n)\ b\ \text{end}\}$

- Objects are obtained by cloning
- They are local to their site of creation but object migration can be programmed

Object operations

- Invocation

  $a.x_1$

- Updating

  $a.x_1 = -2$

- Cloning

  $\text{clone}(a, b)$

- Aliasing

  $\{x => \text{alias } y \text{ of } b\}$

Concurrency and Security

- Objection protection is achieved using the keyword **protected**
- lexical scoping can be used to hide values
- processes execute concurrently on different servers and objects are serialized via mutex mechanisms

Communication

- Shared global name servers allow to import and export local values

net_export("display", display)

let mydisp = net_import("display", Namer);

mydisp.plot(p)

Telescript

- A special-purpose, class-based object-oriented language
- Telescript *agents* autonomously travel on the Telesphere (a Telescript network of engines), doing work for their owner
- A Telescript engine is an interpreter with a collection of built-in classes and a place
- A place is a stationary process that can accept travelling agents

The language

- Classes can inherit from a single superclass and a collection of mix-ins.
- A class can be defined as
  - sealed, or abstract
- Attributes can be defined as
  - private, or public
- The operator **protect**, can be applied to object references. A protected reference can not be used to modify the referred object

Telescript agents

Agents are processes with the following attributes:

- the telename = (authority, pid)
- the owner
- the sponsor
- the client
- the permit=(age, extent, priority, canCreate, canGo, canGrant, canDeny…)

Agent mobility

- Agents are invoked via the command **go**
- The capabilities of an agent are computed as the minimum of various permits that can be imposed by local and destination places
- Mix-ins are available for further protection of classes
  - unmoved, uncopied, copyrighted, protected

Discussion

- Telescript deals with denial of access attacks
- Telescript agents have their own initiative to travel and are independent of the user's connections
- Security has been a central consideration in the development of the language
- However, there has been no justification for consistency claims

Programming Languages

Language	OO	Concurrency	Mobility	Safety	Security model
Java	Yes	Yes	Weak	Yes	PL
Limbo		Yes	Weak	Yes	OS
O'Caml	Yes	Yes	Weak	Yes	PL
Safe-Tcl			Weak	Yes	OS
Obliq	Yes	Yes	Strong	Yes	PL
Telescript	Yes	Yes	Strong	Yes	PL

Comparison

- Mobility
  - Java, O'Caml and Limbo, exhibit weak mobility (code is downloaded and executed).
  - Obliq and Telescript are strongly mobile (processes can be programmed to migrate).

Comparison

- Security model
  - Java: trusted libraries (security monitor) check that scope rules are not violated
  - Limbo: resources are available as files that can be accessed via “secure” modules
  - O’Caml: safe libraries
  - Obliq: language constructs
  - Safe-Tcl: restricted functionality

Comparison

- Trust in the object code:
  - Limbo and O'Caml: cryptographic signature of trusted authority
  - Java: verification of object code

Perspectives

- Need for formal treatment of language and security aspects
  - programming language semantics
  - formal statements and verification of properties
- Integrated development/analysis of mobile code languages and operating systems
  - formal models
  - automatic and interactive proof techniques