Software Engineering

Prof. Dr. Bertrand Meyer
Dr. Manuel Oriol
Dr. Bernd Schoeller

Lectures 20: Engineering Complex Systems
Today

- **Complex Systems**
  - What is it?
  - Examples
  - Technologies involved
- **Dynamically Evolvable Systems**
  - What is it?
  - Examples
  - Technologies involved
- **Challenging Systems Engineering**
  - How to engineer dynamic systems on dynamic?
  - How to specify or model them?
In this lecture, a complex system is a system that can only be understood through its code, its platform and its dynamics.

This covers multithreaded applications, applications relying on complex runtime, dynamically evolvable programs...
Kinds of Systems

- Embedded Systems
- Concurrent Systems
- Distributed Systems
- Modifiable Systems
Embedded Systems

- Typically:
  - Limited amount of memory
  - Limited amount of processor time
  - Limited interfaces and limited GUI
  - Real-time constraints
Concurrent Systems

- Typically:
  - Multithreaded or Multiprocess applications
  - Concurrent accesses to data
  - Trade-off performance/safety
Distributed Systems

- Typically:
  - Several applications running concurrently
  - Communications through the network
  - Service-oriented
  - Several administrative domains
Modifiable Systems

- Typically:
  - Applications or groups of applications that can have their code evolve at runtime
  - Service-oriented
Challenges: Designing the apps

- How to represent applications that are not simple to represent?
- How to write specifications?
Challenges: Feasibility

- Feasibility is always an issue
- Try and find projects with similar features to assess it
- Specialists are the key
Challenges: Understanding the apps

- Difficult to understand because the code relies heavily on the runtime (e.g. Aspects in Spring)

- Boilerplate code is everywhere (e.g. JDBC code, Spring Code)

- Become a specialist to code them: tricks can make a big difference (e.g. iPhone, Palm Pilots)
Challenges: Testing the apps

- Do you test them?
- How to test them?
- How to make tests reproducible?
- Doug Lea: “I wrote thousands of unit tests”
Challenges: Prototyping

- These applications tend to be of the “All or nothing” type: once the difficult part is coded it is quite easy and fast to code the rest... Before it is, it is difficult to show prototypes that make sense to non-specialists

- Application Deployment is a challenge
Challenges: Debugging the apps

- The runtime structures are difficult to debug (e.g. there is no good debugger for concurrent apps)

- Printed values do not always reflect actual structures (e.g. in Java System.out.println(o))

- Once you get down to the runtime, how do you know what actually happens?
DYNAMICALLY EVOLVABLE SYSTEMS
Example of Challenging Systems

- Dynamically evolvable applications are applications that can change their code and data types at runtime.

- What applications do that?
  - Apps with plug-ins
  - Components-Based applications
  - Applications made on top of dynamic programming languages (interpreted, compiled)
Applications with Plug-ins

- Eclipse
- Firefox
- Photoshop
- iTunes
Applications with made from Components

- Eclipse
- TomCat
Dynamically Evolvable Applications

- None?

- Well... some very specialized cases are running (ex. DynInst) and in the general case, we have done it!!! 😊
Technologies involved

- Dynamic Loading + Dynamic Linking
- Instrumentations
- On more level of indirection
Dynamic Loading and Dynamic Linking

- **Dynamic Loading**: the capacity to load code at runtime

- In C/C++ there are various loaders and linkers
  - the most used one on Linux/Unix is dlopen

- In Java the JVM loads classes lazily and the ClassLoader abstraction can be used for specific loading
Instrumentation

- Instrumentation means that you replace some code by some other code...

- Aspects can help you out with that (e.g. Aspects used in Spring for logging and transactions)
One more level of indirection

- Simply keep a table that returns a pointer on the most recent structure on demand.

- Works for methods...
TWO SYSTEMS FOR DYNAMIC UPDATES
Why Dynamic Updates?

- 24/7 service providers
- Embedded systems
- End-user updates
Existing Solutions?

- Stop and launch
- Redundant hardware
- Unsafe (e.g. Smalltalk 80)
- Safe but limited (e.g. Malabarba 2001, K42, dynamic libraries...)
Issues: D’ replaces D

Object Graph

Call Graph

a:A → b:B → d’:D’ → e:E → f:F

a.m1 → d’.m4 → e.m5 → f.m6

b.m2 → c.m3
Requirements for Dynamic Updates in Java

- Safe (type-safe)
- As flexible as possible
- Usable by programmers with minimal effort
- Use a standard JVM
- Change code on-the-fly with no constraint on time of change or type of change
Solution: Components

Object Graph

Call Graph

Solved with Associative Naming

Solved with Asynchronous Invocations
d offers services that are requested, if d changes, they are called on its next version
Asynchronous Invocations

arguments passed by value
Software Produced: LuckyJ

- Component platform
- Components in isolation (using class loaders)
- Associative naming
- Asynchronous invocations
- State transfer from one component to another when updated
- Local, distributed centralized, and P2P implementation
Results: WeeselJ Web Server

- 160 versions of some parts of the code
- 18 months
- 4 reboots due to external causes


Requirements for Dynamic Updates in C

- Safe (type-safe)
- As flexible as possible
- Usable by programmers with minimal effort
- Make real programs dynamically updatable
- Dynamic patches easy to make
Issues: D' replaces D
f4' replaces f4

Data Graph

Call Graph

Solved with Type Transformers
Solved with Indirect Calls
Type Transformers

Based on versions and a level of indirection
Indirect Calls

It always calls the latest version of f4
Software Produced: Ginseng

- A compiler based on cil that performs static analyses and links to runtime libraries
- Instruments concrete accesses of named types and function calls
- Type transformers generated mostly automatically with heuristics
- Loop extraction for long-running loops
- The stack is untouched (implies delayed updates to keep type-safety)
Using Ginseng
Results

- Updated real programs
  - 3 years/12 versions of vsftpd
  - 1 years/9 versions of openssh
  - used the tool on apache, bind, zebra, linux kernel...

ENGINEERING COMPLEX SYSTEMS
Complex Systems?

- What is different with complex systems is that intuition is not enough

- Studies to check feasibility are time-consuming

- Theory/Specifications to take care of nasty details.
Usual Techniques still work…

- They just don’t reflect very well what is happening in the system…

- How do you model dynamic updates?
Figure 4: Updating Pattern.
Techniques especially adapted to complex systems

- Example of Petri Nets for concurrency... (as well as formal calculi such as Pi-Calculus or Ambients)

- Example of updating components (which is a UML extension)

- Do not hesitate to make your own extension of the norm or to reuse an existing one (from a research paper for example)
Reusing Formalisms...
Petri Nets

- Places
- Token
- Inputs
- Transition
- Outputs
Petri Net

- Firing a transition means removing all tokens from the input state and putting one on each of the output state.

- Inputs are all needed to fire a transition.

- The whole thing is asynchronous and non-deterministic.
Petri Nets Example: simple concurrency

Petri Net Diagram:
- Initial place
- Fork transition
- Join transition
- Multiple places connected to fork and join transitions
Petri Nets Example: simple concurrency
Petri Nets Example: simple concurrency
Example

http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/java/Guth/
Conclusions on Complex Systems

- Engineering systems is not easy: Simply coding them will not work.

- Need to have more rigor than usual in the process

- Modeling behavior and the way it works is a necessary step

- The good software engineer is able to work these things out and reuse modeling techniques found on the web, or in articles when possible, make his own when not possible.
Today

- **Complex Systems**
  - What is it?
  - Examples
  - Technologies involved
- **Dynamically Evolvable Systems**
  - What is it?
  - Examples
  - Technologies involved
- **Complex Systems Engineering**
  - How to engineer dynamic systems on dynamic?
  - How to specify or model them?