

Introduction to Model-based Design of Distributed and Mobile Systems

Thomas Hildebrandt
Dept. of Theoretical Computer Science, ITU

February 4, 2005

Today's lecture

- Practical issues (9.15-9.30)
- Introduction to process models
 - Historical overview (9.30-10.00)
 - Models formally 10.15-12)

Practical issues

- Course home-page at www.itu.dk/IMDD/F2005/
 - News
 - Lecture plan
 - Teachers
 - Lecture notes, Tool(s), etc.

Why this course?

- Distributed and mobile communicating systems abound us,
- are deceptively easy to write as programs in e.g. Java or .NET, and
- technological advances (Bluetooth, WLAN, IP ...) make it cheap and easy to enable communication between all sorts of devices

However, race conditions, the state-explosion and their dynamic nature make them almost impossible to test and understand!

Formal models

- Abstraction
- General understanding of concepts
- Foundation for safe programming languages
- Formal and/or automatic analysis

Aiming at provable safe and correct systems, still a long way to go...

Quotes from the Economist

- "software bugs are so common that their cost to the American economy alone is \$60 billion a year" (estimate by NIST)
- "writing models and then writing the code never works, the code and the model are two ways of looking at the same thing" (Grady Booch, co-inventer of UML)

A brief history of formal models

During the last 70 years focus has been on models of

- computation (30-40'ties)
- recognition (50'ties)
- communication (60'ties-80'ties)
- mobile communication (90'ties)
- spatial computation 2000-

Models of computation (30-40'ties)

The big question: "Which functions can be computed?"

- A. Turing: the Turing machine
foundation for computability and complexity
- A. Church and S. Kleene: the λ -calculus
foundation for computability and functional programming languages (ML)
inspiration for later calculi for concurrent, distributed and mobile systems!

State of IT systems: Automatic calculators with punched cards, developing into ENIAC (Electrical Numerical Integrator And Calculator) in '42 and the von Neuman architecture in '45.

Models of recognition (50'ties)

The big question: Understand human intelligence

- Kleene's theorem: Finite state machines recognizes exactly regular languages
- Rabin and Scott: Non-determinism does not add recognizing power
- Chomsky: Strict hierarchy of grammars, from finite automata to Turing machines

State of IT systems: Transistors and magnetic core memory. Operating systems, batch processing and time-sharing.

Models of communication (60-80'ties)

- Carl A. Petri: Kommunikation mit Automaten (Petri Net)
- David Harel: Statecharts
- Robin Milner: Calculus for Communicating Systems (CCS)
- Tony Hoare: Communicating Sequential Processes (CSP)

State of IT systems: cathode-ray-tube displays, multi-purpose personal mini-computers (PCs), VLSI chips, Cray super computers, DOS operating system.

Models of mobile communication (80'ties)

- Milner: The π -calculus
- Bengt Thomsen: Calculus of Higher Order Communicating Systems (CHOCS)

State of IT systems: Birth of the Internet and Java. Massively concurrent, distributed systems. Mobile code (as applets) and concurrent threads deceptively easy to program.

Models of spatial computations (90'ties to now)

- Cardelli: The Ambient calculus
- Calculi for biological systems
- Milner, Jensen: Bigraphical reactive systems
- UK Grand Challenge on Science for Global Ubiquitous Computing

Research in Models for Concurrency and Mobility at ITU

- The Concurrency and Mobility group: www.itu.dk/research/theory/CoMo
- Formal models for testing
- Calculus of Higher-order Mobile Resources
- Bigraphical Programming Languages (and models)