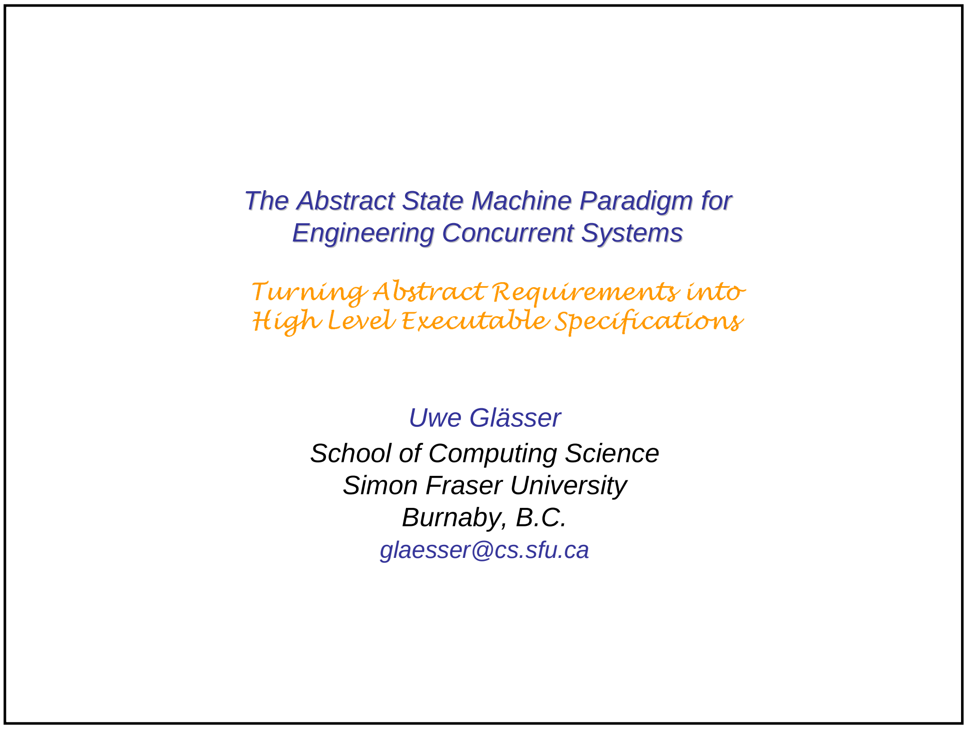
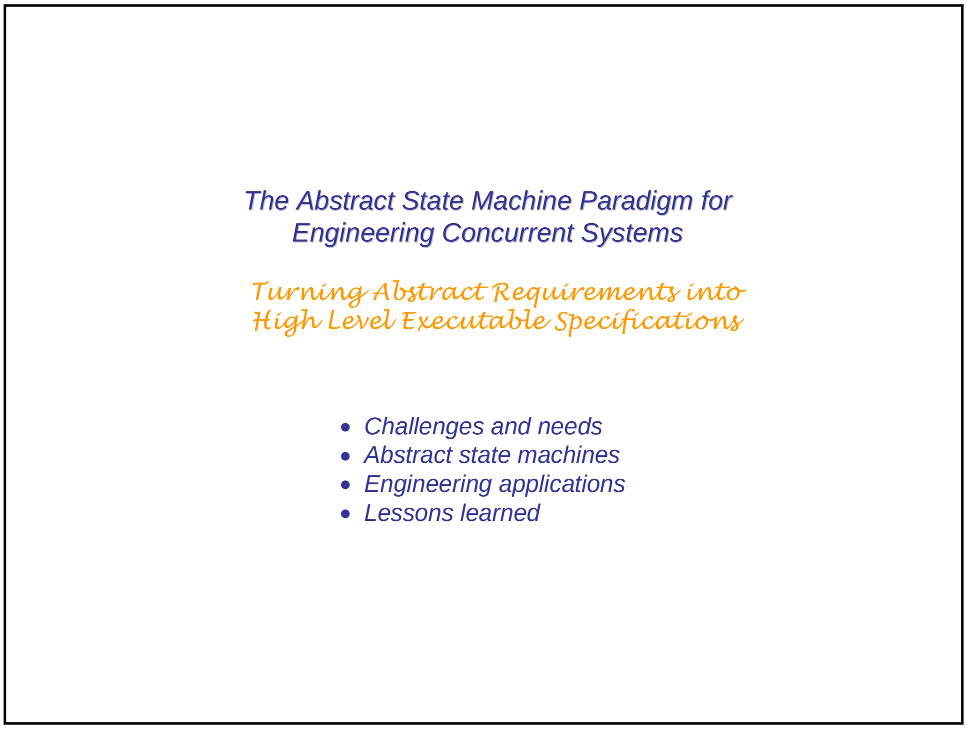


The Abstract State Machine Paradigm for Engineering Concurrent Systems

Turning Abstract Requirements into High Level Executable Specifications

Uwe Glässer
School of Computing Science
Simon Fraser University
Burnaby, B.C.
glaesser@cs.sfu.ca



The Abstract State Machine Paradigm for Engineering Concurrent Systems

Turning Abstract Requirements into High Level Executable Specifications

- *Challenges and needs*
- *Abstract state machines*
- *Engineering applications*
- *Lessons learned*

Challenges and Needs

Establishing requirements is the first and most important step on the way from a fuzzy concept to a concrete implementation.

INFORMAL REQUIREMENTS

Ambiguo
us

Incomplet
e

Inconsistent

Formalization helps eliminating deficiencies at high levels of abstraction

Challenges and Needs

Establishing requirements is the first and most important step on the way from a fuzzy concept to a concrete implementation.

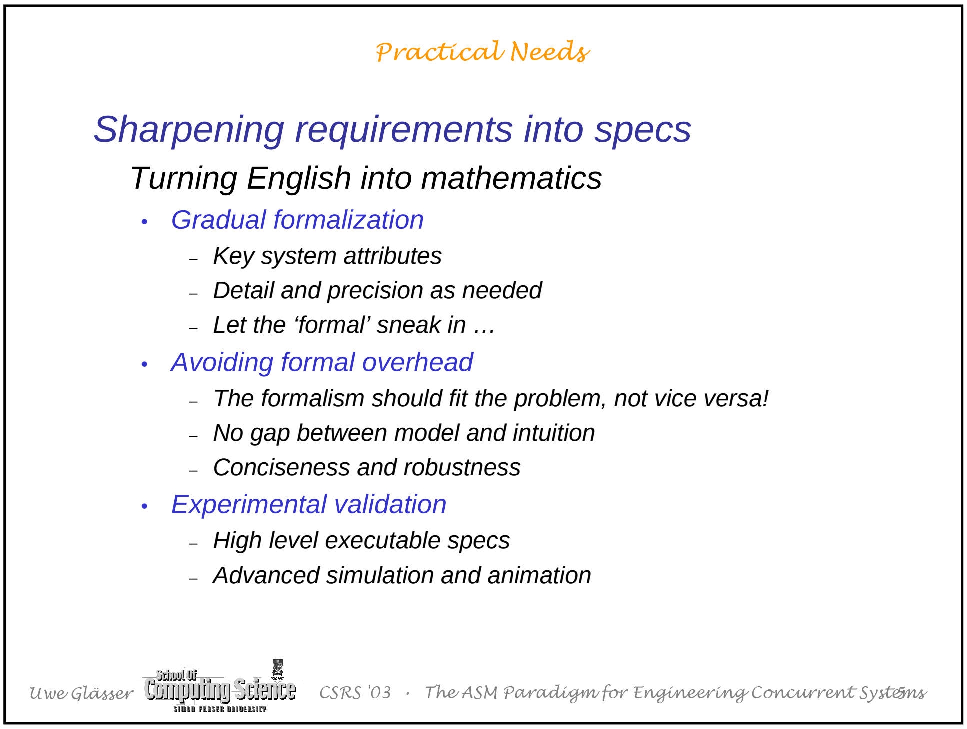
Control intensive software

Communications Software

Protocol Architectures

Distributed Embedded Systems

Real Time Protocols

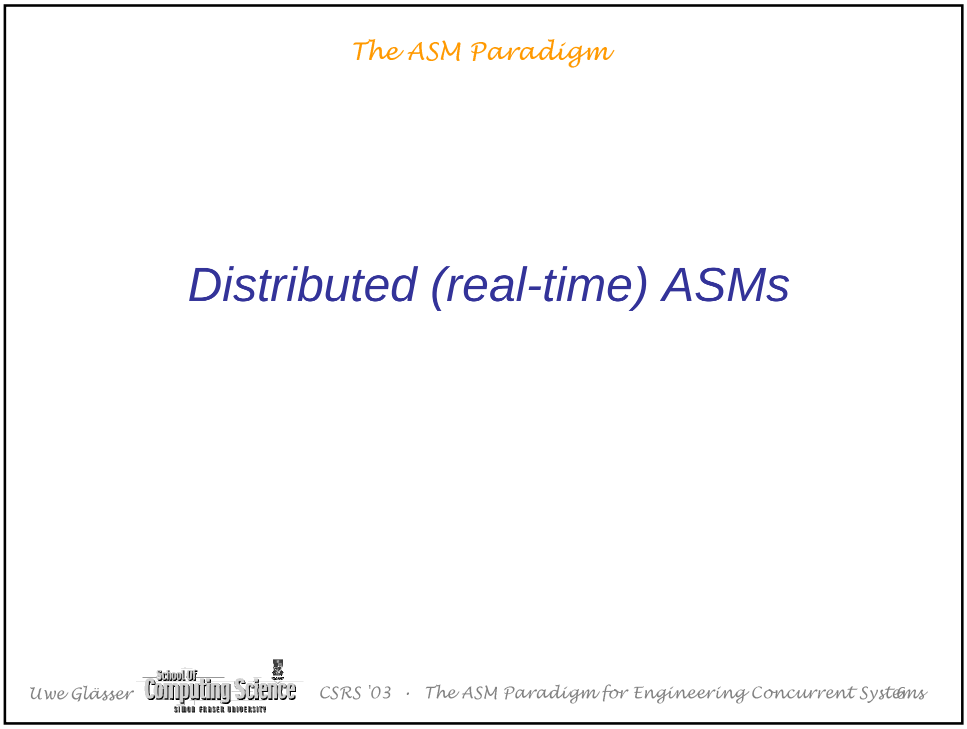


Practical Needs

Sharpening requirements into specs

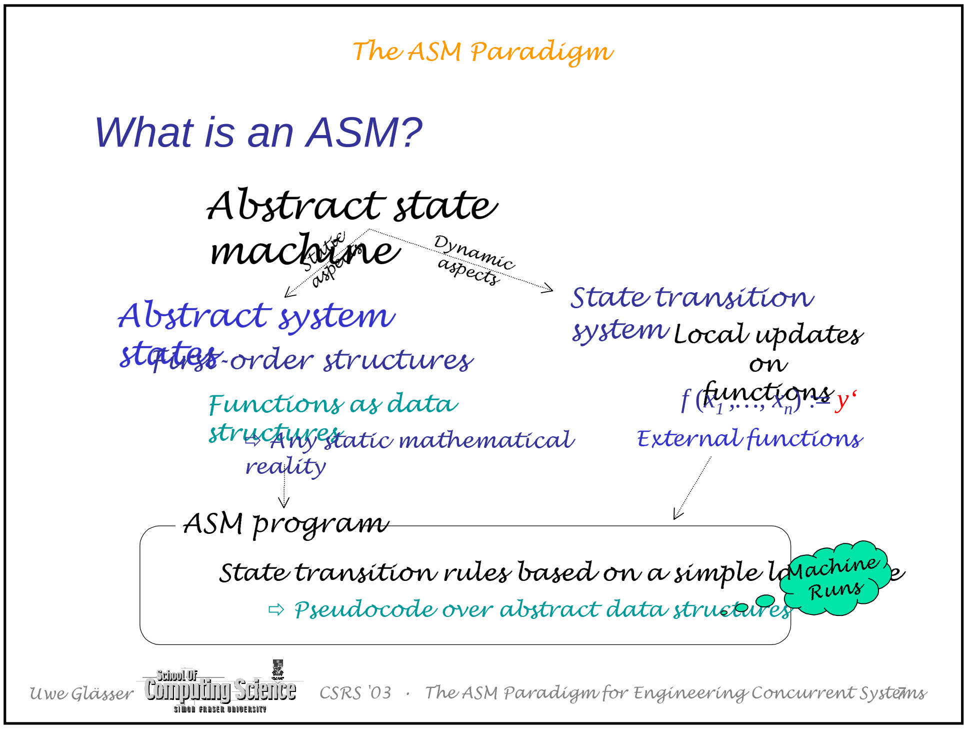
Turning English into mathematics

- *Gradual formalization*
  - *Key system attributes*
  - *Detail and precision as needed*
  - *Let the ‘formal’ sneak in …*
- *Avoiding formal overhead*
  - *The formalism should fit the problem, not vice versa!*
  - *No gap between model and intuition*
  - *Conciseness and robustness*
- *Experimental validation*
  - *High level executable specs*
  - *Advanced simulation and animation*



The ASM Paradigm

Distributed (real-time) ASMs



The ASM Paradigm

What is an ASM?

Abstract state machine

Static aspects

Dynamic aspects

Abstract system states

First-order structures

Functions as data structures

⇨ Any static mathematical reality

State transition system

Local updates on functions

$f(x_1,\dots,x_n) := y$‘

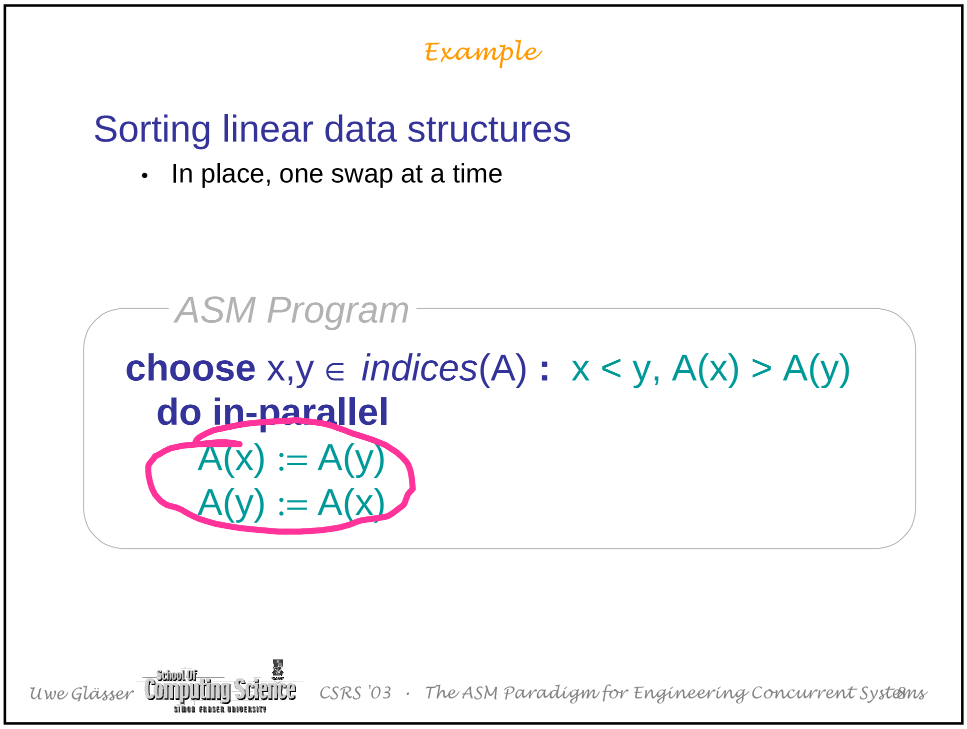
External functions

ASM program

State transition rules based on a simple language

⇨ Pseudocode over abstract data structures

Machine Runs



Example

Sorting linear data structures

- In place, one swap at a time

ASM Program

```
choose x,y ∈ indices(A) :  x < y, A(x) > A(y)
  do in-parallel
    A(x) := A(y)
    A(y) := A(x)
```

Concurrency, Reactivity and Time

Distributed ASM model

- Autonomous agents

  Reading/writing shared locations of global machine states

  - *Potential conflicts are resolved according to partially ordered runs*
  - *Dynamic creation of agents (on demand)*

- Reactive behaviour

  Open System View

  - *Interaction with operational environment*

- Real time behaviour

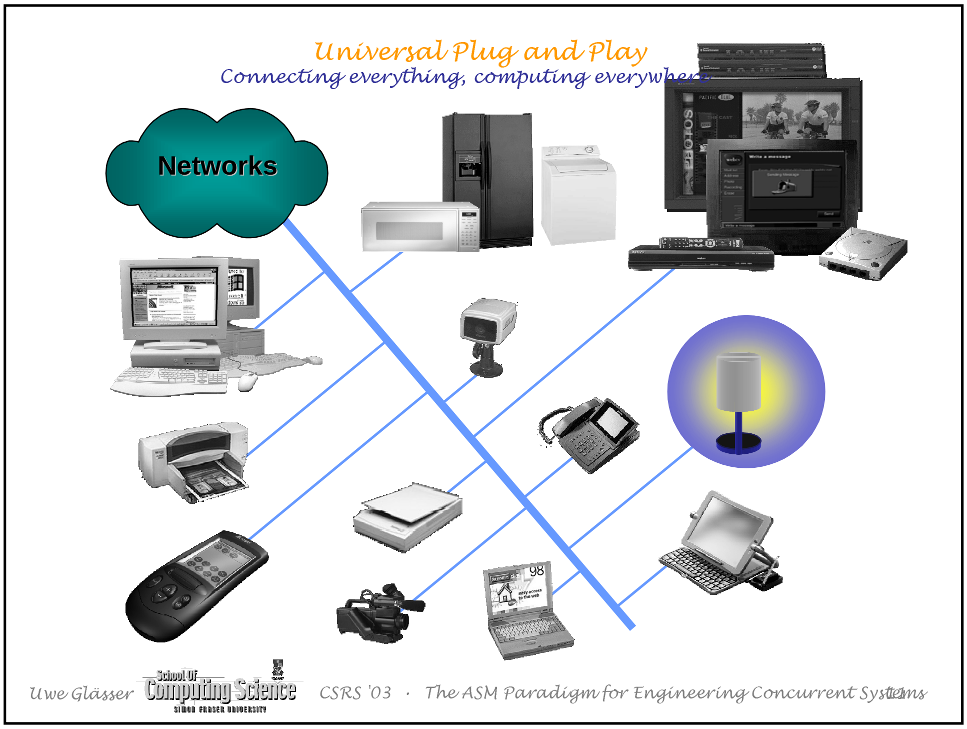
  Continous (dense) time / discrete time

  - *E.g., global system time as real numbers*
  - *Agents react instantaneously*

Engineering Applications

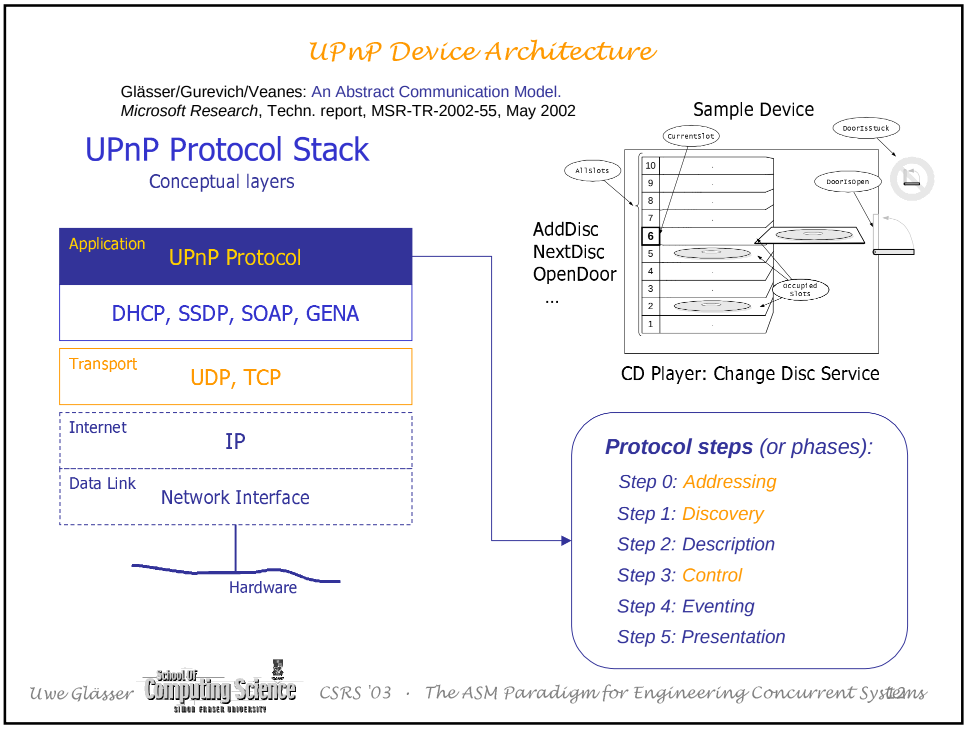
Typical applications of distributed real-time ASMs

Architectures, Languages, Protocols



Universal Plug and Play

Connecting everything, computing everywhere



UPnP Device Architecture

Glässer/Gurevich/Veanes: An Abstract Communication Model.
Microsoft Research, Techn. report, MSR-TR-2002-55, May 2002

UPnP Protocol Stack

Conceptual layers

Sample Device

AddDisc
NextDisc
OpenDoor
...

CD Player: Change Disc Service

***Protocol steps** (or phases):*

Step 0: Addressing

Step 1: Discovery

Step 2: Description

Step 3: Control

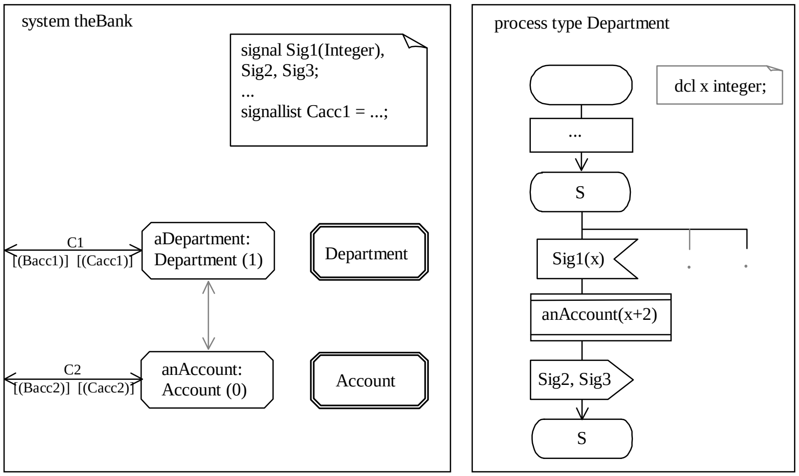
Step 4: Eventing

Step 5: Presentation

ITU-T Recommendation Z.100

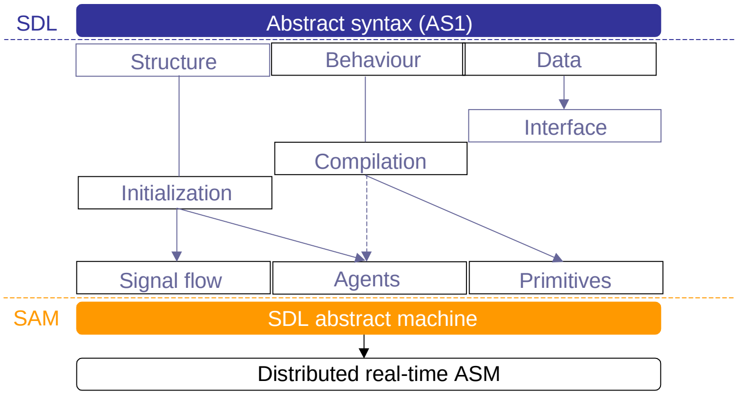
Specification and Description Language - SDL

Glässer/Gotzhein/Prinz: The formal semantics of SDL-2000: Status and perspectives. *Computer Networks*, vol. 42, no. 3, pages 343-358, June 2003



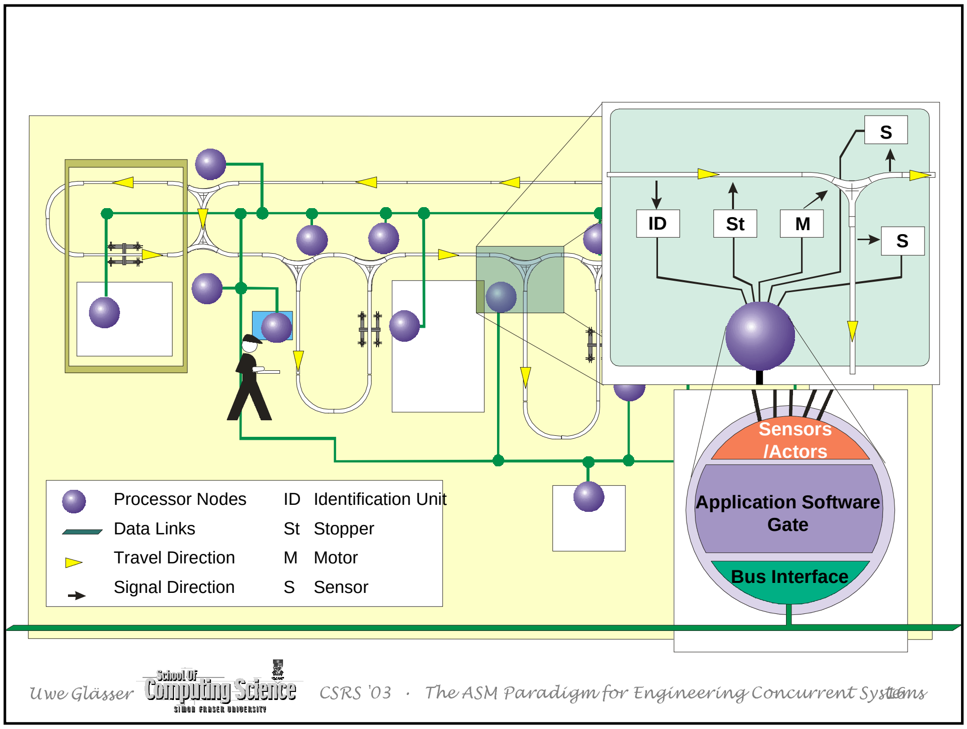
SDL Abstract Machine

SDL-to-ASM Compiler

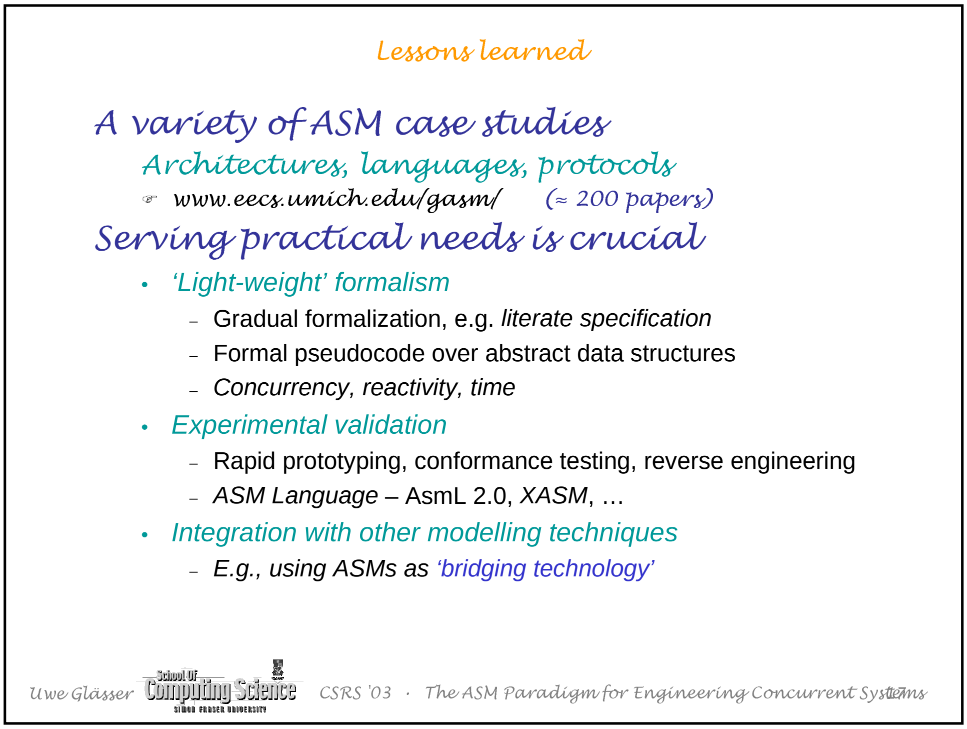


Computer Integrated Manufacturing (CIM)

Distributed Material Flow System



●	Processor Nodes	ID	Identification Unit
▬	Data Links	St	Stopper
▷	Travel Direction	M	Motor
→	Signal Direction	S	Sensor



Lessons learned

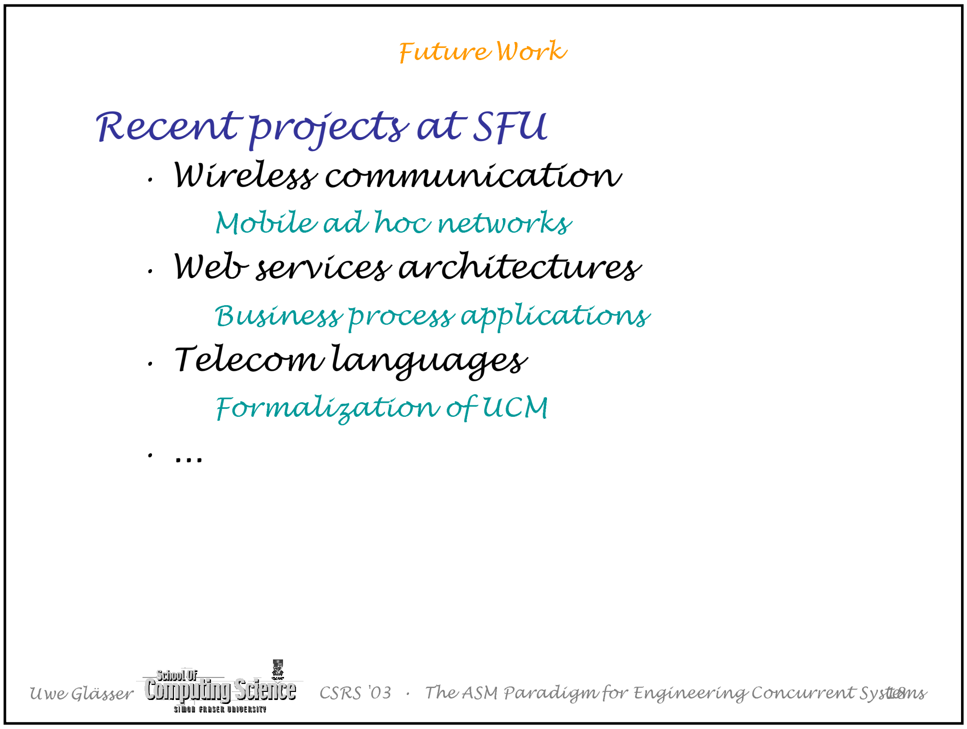
A variety of ASM case studies

Architectures, languages, protocols

☞ www.eecs.umich.edu/gasm/ (≈ 200 papers)

Serving practical needs is crucial

- ‘Light-weight’ formalism
  - Gradual formalization, e.g. *literate specification*
  - Formal pseudocode over abstract data structures
  - *Concurrency, reactivity, time*
- *Experimental validation*
  - Rapid prototyping, conformance testing, reverse engineering
  - *ASM Language* – AsmL 2.0, *XASM*, …
- *Integration with other modelling techniques*
  - *E.g., using ASMs as ‘bridging technology’*



Future Work

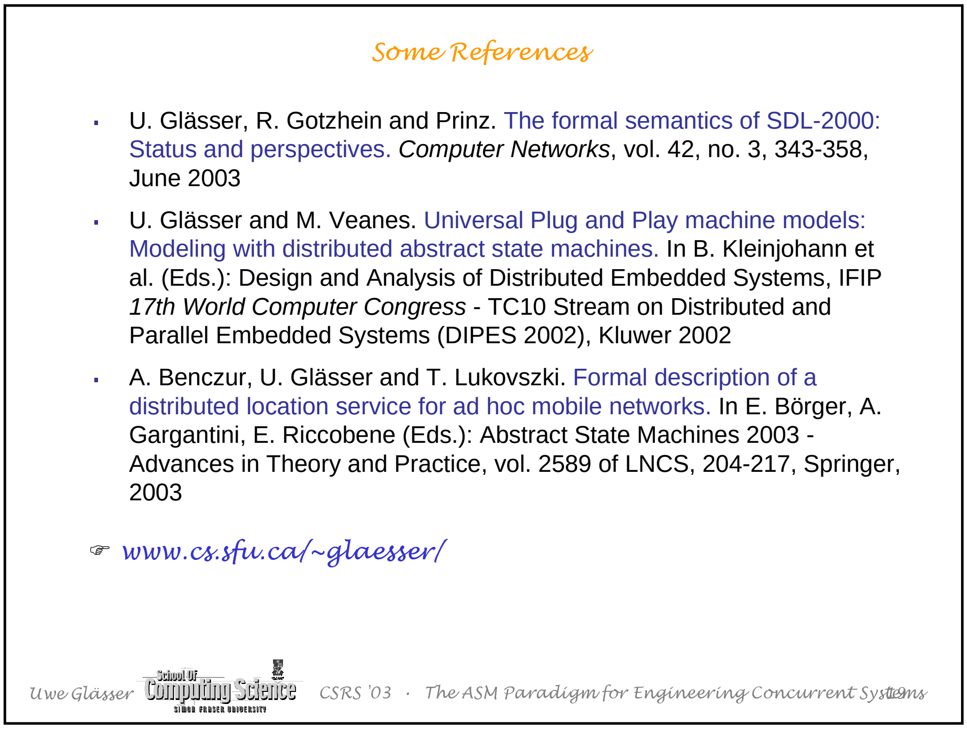
Recent projects at SFU

- Wireless communication

  Mobile ad hoc networks
- Web services architectures

  Business process applications
- Telecom languages

  Formalization of UCM
- …



Some References

- U. Glässer, R. Gotzhein and Prinz. The formal semantics of SDL-2000: Status and perspectives. *Computer Networks*, vol. 42, no. 3, 343-358, June 2003
- U. Glässer and M. Veanes. Universal Plug and Play machine models: Modeling with distributed abstract state machines. In B. Kleinjohann et al. (Eds.): Design and Analysis of Distributed Embedded Systems, IFIP *17th World Computer Congress* - TC10 Stream on Distributed and Parallel Embedded Systems (DIPES 2002), Kluwer 2002
- A. Benczur, U. Glässer and T. Lukovszki. Formal description of a distributed location service for ad hoc mobile networks. In E. Börger, A. Gargantini, E. Riccobene (Eds.): Abstract State Machines 2003 - Advances in Theory and Practice, vol. 2589 of LNCS, 204-217, Springer, 2003

☞ *www.cs.sfu.ca/~glaesser/*