Systems and Services

Jan Bessai

2017-06-28

technische universität dortmund
fakultät für informatik

KSOS

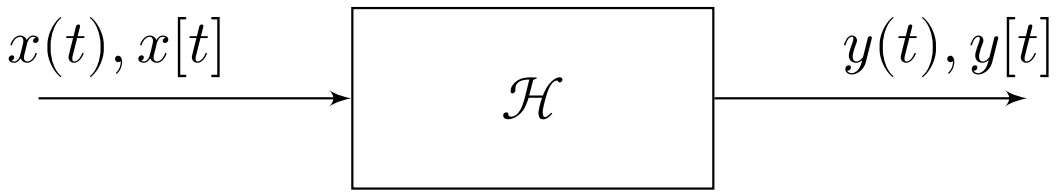
Today:

Komponenten- und Service-orientierte Softwareentwicklung

- Services / Systems
- Concurrency Concepts
- Actors
- Live Example Coding (Akka) + Demo (type inhabitation in simple types!)

Systems (Frey and Bossert [4])

input signal system output signal



Definition (System)

A (time-) continuous system is a map $\mathcal{H}$ from (time-) continuous input signals to (time-) continuous output signals:

$$y(t) = \mathcal{H}\{x(t)\}$$

A (time-) discrete system is a map $\mathcal{H}$ from (time-) discrete input signals to (time-) discrete output signals:

$$y[t] = \mathcal{H}\{x[t]\}$$

System Classification (Frey and Bossert [4])

Domain	(time-) continuous	(time-) discrete / discontinuous
Range	value-continuous value-bounded	value- / amplitude-discrete unbounded
Other properties	linear $\mathcal{H}\{a_1x_1(t) + a_2x_2(t)\} = a_1\mathcal{H}\{x_1(t)\} + a_2\mathcal{H}\{x_2(t)\}$ timeinvariant $y(t - t_0) = \mathcal{H}\{x(t - t_0)\}$ causal $y(t_0) = \mathcal{H}\{x(t \leq t_0)\}$ stable $\lvert x(t)\rvert \leq \infty \Rightarrow \mathcal{H}\{x(t)\} \leq \infty$ dynamic (memory) $y(t) = \mathcal{H}\{x(t), x(t - t_1), ...\}$ deterministic	non-linear timevariant acausal unstable static (no memory) non-deterministic, indeterministic stochastic

Services and Systems

Definition (Service)

According to the definition above, services are just time- and value-discrete systems.

- Service composition is function composition
- Algebraic abstractions can be used
- We model communicating services with connected systems $y[t] = \mathcal{H}_1\{\mathcal{H}_0\{x[t]\}\}$
- Concurrency via time-dependence $x(t), y(t)$
- Channels are also systems $y[t] = \mathcal{H}_1\{\mathcal{H}_{\text{channel}}\{\mathcal{H}_0\{x[t]\}\}\}$

Research chance: apply classifications and lessons from system theory to computer science!

Concurrent Systems in Computer Science (Practice)

- POSIX threads (`man 7 pthreads`)
- Corba
  - interface stubs
  - transparent method calls via network
- Apache Thrift [3]
  - C-header like interface description language
  - compiled to client and server implementations in different languages
  - network and serialization layer abstracted away
  - invented by Facebook, used e.g. by Uber and Siemens
- The Internet
  - REST APIs (HTTP + no server side state)
- many more...

Concurrent Systems in Computer Science (Models)

Many formalisms for concurrent systems:

- Petri Nets [14]
  - EGP, ES
- Communicating Sequential Processes (CSP) [7]
  - used in Go [11]
- CCS [12], π-Calculus [13]
  - EGP
- Threads, Mutexes, Semaphores, Monitors [2]
  - BS
  - all modern operating systems
- Kahn Process Networks [10]
  - ES
- Arrow-Calculus [9]
  - Haskell [8]
- **Today:** Actors [6]

Actors – Why?

- Simple (old) concept
- Theoretically interesting:
  - nondeterminism vs. indeterminism
- Direct connection to services
- Practically interesting
  - Foundation of Erlang
  - Libraries for almost every language
  - Scala/Java: Akka (users include: Intel, Amazon, Paypal, Zalando, Agido)

Actors - Basics [1]

Definition (Actor)

An Actor is an object capable of receiving a message and then performing three operations:

1. create a finite number of new actors
2. send a finite number of messages
3. designate the behavior to be applied to the next message

Definition (Actor System)

An Actor System

1. manages names by which actors address each other
2. provides message delivery guarantees:
   - arrival
   - duplicate freedom
   - **no order/time guarantees**

For a mathematical formalization see [5]

- e.g. to prevent Zenon machines [15]

References

[1] Clemens Szyperski Carl Hewitt Erik Meijer. *The Actor Model*. 2012. URL: https://www.youtube.com/watch?v=1zVdhDx7Tbs.

[2] E. W. Dijkstra. "Solution of a Problem in Concurrent Programming Control". In: *Commun. ACM* 8.9 (Sept. 1965), pp. 569–. ISSN: 0001-0782. DOI: 10.1145/365559.365617. URL: http://doi.acm.org/10.1145/365559.365617.

[3] Apache Foundation. *Thrift*. 2017. URL: https://thrift.apache.org/.

[4] Thomas Frey and Martin Bossert. *Signal-und Systemtheorie*. Springer-Verlag, 2009.

[5] Carl Hewitt and Henry G. Baker. "Laws for Communicating Parallel Processes". In: *IFIP Congress*. 1977, pp. 987–992.

[6] Carl Hewitt, Peter Boehler Bishop, Irene Greif, Brian Cantwell Smith, Todd Matson, and Richard Steiger. "Actor Induction and Meta-Evaluation". In: *Conference Record of the ACM Symposium on Principles of Programming Languages, Boston, Massachusetts, USA, October 1973*. Ed. by Patrick C. Fischer and Jeffrey D. Ullman. ACM Press, 1973, pp. 153–168. DOI: 10.1145/512927.512942.

[7] C. A. R. Hoare. "Communicating Sequential Processes". In: *Commun. ACM* 21.8 (Aug. 1978), pp. 666–677. ISSN: 0001-0782. DOI: 10.1145/359576.359585. URL: http://doi.acm.org/10.1145/359576.359585.

[8] Paul Hudak, Antony Courtney, Henrik Nilsson, and John Peterson. "Arrows, Robots, and Functional Reactive Programming". In: *Advanced Functional Programming, 4th International School, AFP 2002, Oxford, UK, August 19-24, 2002, Revised Lectures*. Ed. by Johan Jeuring and Simon L. Peyton Jones. Vol. 2638. Lecture Notes in Computer Science. Springer, 2002, pp. 159–187. DOI: 10.1007/978-3-540-44833-4_6.

[9] John Hughes. "Generalising monads to arrows". In: *Sci. Comput. Program.* 37.1-3 (2000), pp. 67–111. DOI: 10.1016/S0167-6423(99)00023-4.

[10] Gilles Kahn. "The Semantics of Simple Language for Parallel Programming". In: *IFIP Congress*. 1974, pp. 471–475.

[11] Thomas Kappler. *package csp*. 2017. URL: http://godoc.org/github.com/thomas11/csp.

[12] Robin Milner. *A Calculus of Communicating Systems*. Vol. 92. Lecture Notes in Computer Science. Springer, 1980. ISBN: 3-540-10235-3. DOI: 10.1007/3-540-10235-3.

[13] Robin Milner, Joachim Parrow, and David Walker. "A Calculus of Mobile Processes, I". In: *Inf. Comput.* 100.1 (1992), pp. 1–40. DOI: 10.1016/0890-5401(92)90008-4.

[14] Carl Adam Petri. "Communication with automata". eng. PhD thesis. Universität Hamburg, 1966.

[15] Petrus H. Potgieter. "Zeno machines and hypercomputation". In: *Theor. Comput. Sci.* 358.1 (2006), pp. 23–33. DOI: 10.1016/j.tcs.2005.11.040.