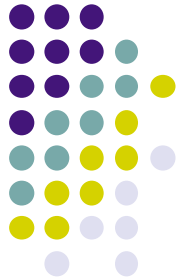
Conclusion of the course



- We hope you have enjoyed this course
  - If you are a student, we hope it gives you an incentive to continue studying programming
  - If you are a professional, we hope that we were able to hold your interest and give you some new insights along the way
- We are using and developing Oz and Mozart2 for education and research, in several ongoing projects
  - We are always looking for help and you are welcome to join our community!
  - Just make yourself known on the mozart-users mailing list (see www.mozart-oz.org) or send me mail
- We will continue to improve this course
  - All feedback and constructive comments are welcome

A final reflection on programming

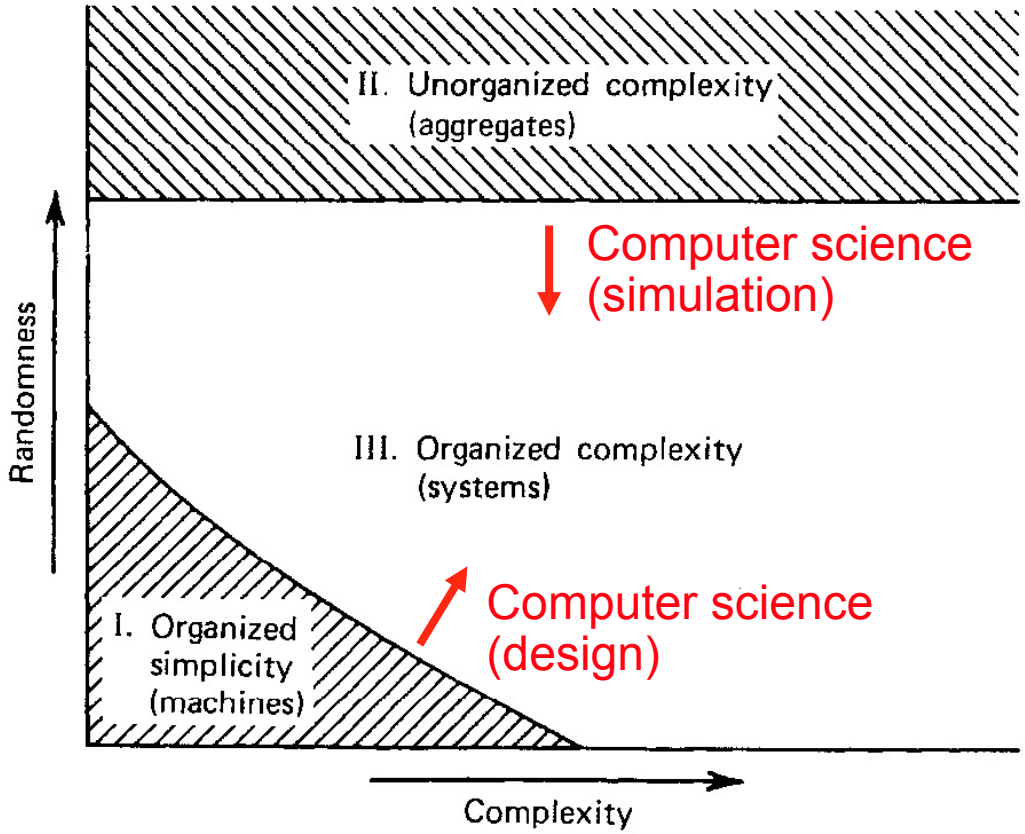
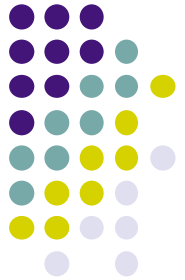


Diagram from [Weinberg 1977]
An Introduction to General Systems Thinking

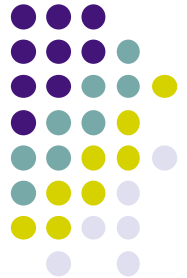
- Programming languages are not arbitrary inventions of the human mind
  - Programming language design is a science of nature, like physics
  - Successful programming languages model some essential aspects of how to construct complex systems
  - Systems are composed of parts that interact in a well-defined way to provide a new behavior
- Computer science is the most advanced discipline for building complex systems
  - Programming languages are the vanguard of complex systems
- Each newly discovered programming concept advances the science of building complex systems

Further study



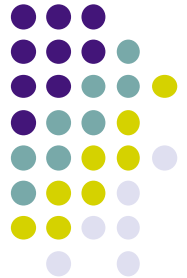
- We have only scratched the surface of programming languages and paradigms
  - Paradigms poster: www.info.ucl.ac.be/~pvr/paradigms.html
- To help you further your study, I have compiled a short bibliography of reference works
  - It is rather idiosyncratic and definitely not complete
  - Not all works focus on programming languages, but all have helped me understand programming

Short bibliography



- Harold Abelson, Gerald Jay Sussman, and Julie Sussman, *Structure and Interpretation of Computer Programs*, 2nd edition, MIT Press, 1996.
- Genrich Altshuller, *The Innovation Algorithm: TRIZ, Systematic Innovation, and Technical Creativity*, Technical Innovation Center, 2005.
- Joe Armstrong, Mike Williams, Claes Wikström, and Robert Virding, *Concurrent Programming in Erlang*, Prentice Hall, 1996.
- Frederick P. Brooks, Jr., *The Mythical Man-Month*, Anniversary edition, Addison-Wesley, 1995.
- Sébastien Doeraene, *Ozma: Extending Scala with Oz Concurrency*, Master's thesis, Université catholique de Louvain, June 2011.
- Sébastien Doeraene and Peter Van Roy, A New Concurrency Model for Scala Based on a Declarative Dataflow Core, *Fourth Annual Scala Workshop*, July 1-2, 2013.
- Doug Lea, *Concurrent Programming in Java*, 2nd edition, Addison-Wesley, 2000.
- Bertrand Meyer, *Object-Oriented Software Construction*, 2nd edition, Prentice Hall, 2000.
- Martin Odersky, *Programming in Scala*, 2nd edition, Artima, 2011.

Short bibliography



- Chris Okasaki, *Purely Functional Data Structures*, Cambridge University Press, 1999.
- Benjamin Pierce, *Types and Programming Languages*, MIT Press, 2002.
- Leon Sterling and Ehud Shapiro, *The Art of Prolog*, MIT Press, 1986.
- Peter Van Roy and Seif Haridi, *Concepts, Techniques, and Models of Computer Programming*, MIT Press, 2004.
- Peter Van Roy, Programming Paradigms for Dummies, *New Computational Paradigms for Computer Music*, IRCAM/Delatour France, 2009.
- Peter Van Roy, The CTM Approach for Teaching and Learning Programming, *Horizons in Computer Science Research*, Chapter 5, Vol. 2, Nova Science Publishers, Jan. 2011, pp. 101-126.
- Gerald M. Weinberg, *An Introduction to General Systems Thinking*, Silver Anniversary Edition, Dorset House, 2001.