CASE

CASE SCHOOL OF ENGINEERING

Introduction to Software Engineering

Andy Podgurski

Electrical Engineering & Computer Science Dept.

Software Engineering

The term ***software engineering*** has multiple meanings:

- The application of “engineering principles” to software development
- The software development profession
- A field of research that aims to improve methods of software development

Software engineering methodology is the body of techniques used to develop software.

Goals of Software Engineering

To produce, as quickly and inexpensively as possible, software that is:

- Easy to use
- Reliable
- Efficient
- Straightforward to maintain, adapt, and enhance
- Secure

Aspects of Software Engineering

Technical aspects:

- Specification
- Design
- Programming
- Inspection and Testing
- Static & dynamic analysis
- Debugging
- Maintenance
- Configuration management

Aspects of Software Engineering cont.

Non-technical aspects:

- Project management
- Psychology:
  - cognitive, behavioral, organizational
- Law:
  - contracts, liability, intellectual property

Software Complexity

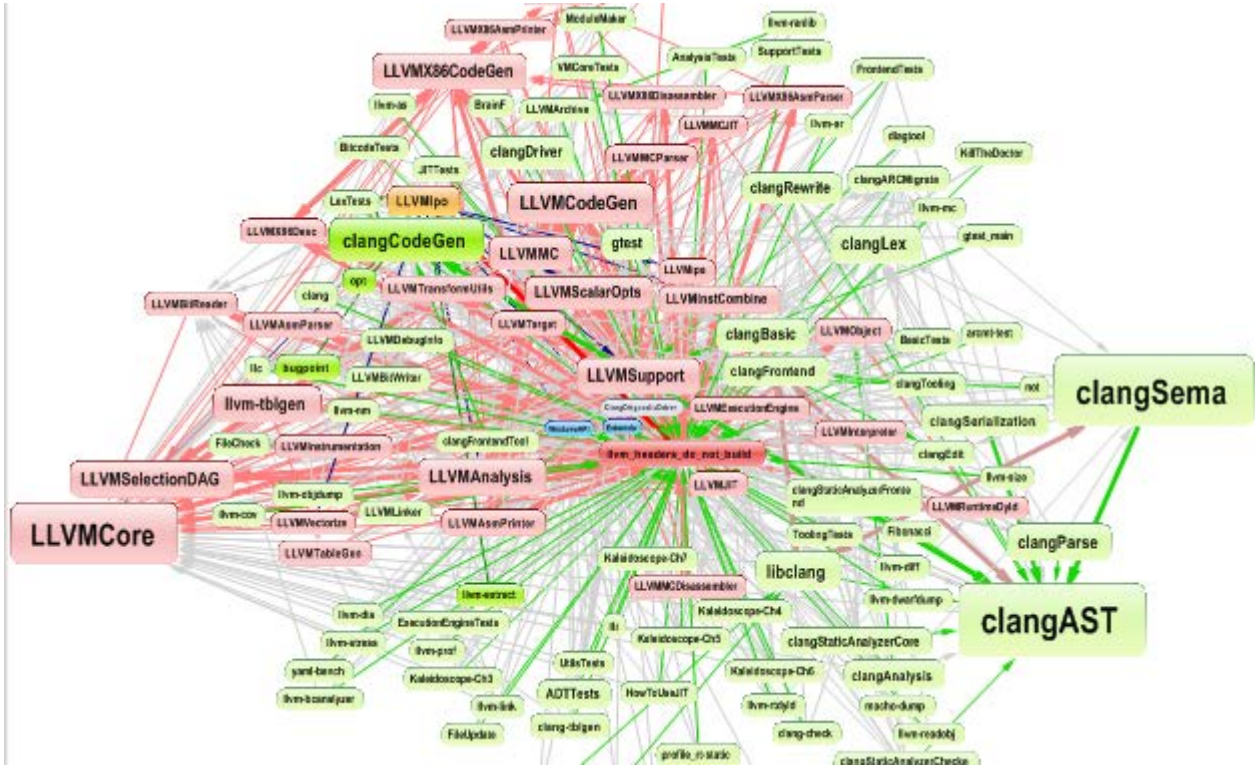
A primary issue confronting software engineers is ***complexity***:

- Problem complexity
- Design/implementation complexity
- Platform complexity

Large software systems are among the most complex artifacts ever produced by man.

Examples: Windows 8.1 consists of ***~80 million lines*** of source code. Google (all services) consists of ***~ 2 billion SLOC***.

Example: *clang* C-Language Family LLVM Frontend Architecture



www.cppdepend.com/Doc_VS_Arch.aspx

Software Complexity cont.

- Complex systems are difficult and time-consuming to produce and to maintain.
- They cannot be fully understood by any one person.
- They seldom satisfy all user needs and desires, which are ***highly changeable***.
- They invariably contain ***residual defects***.
- Many software development projects are completed ***late*** and ***exceed their budget***.
  - Some are never completed.

Software Engineering Methodology

- Broad collection of techniques and tools addressing each phase of software development
- Continually evolving
- Specialized for particular subfields, e.g.,
  - Web applications
  - Real-time systems
  - Health informatics
- Influential methods:
  - Object-oriented and aspect-oriented programming
  - “Agile” methods
  - Design patterns
  - Test-driven design

Coarse Goal

To help you to understand the challenges, key ideas, and methods of large-scale software development