Compositional programming

Early history
- paradigm: procedural composition
  - motivation: code reuse, managing complexity
- subroutines (David Wheeler, 1949)
  - repeatedly invoke common code sequences
- libraries (FORTRAN, 1955)
  - huge libraries led to long-live success of FORTRAN
- recursion (ALGOL, 1958)
  - initially a hack; turned out to be essential for abstraction
  - reentrant procedures
- modules (COBOL, 1959)
  - enabled stepwise decomposition of large systems

OOP
- paradigm: object composition + incremental refinement
  - motivation: domain modeling
- ADTs
  - encapsulation
  - P2 slides
  - abstraction
  - information hiding
- Simula vs Smalltalk
  - OOP as an add-on
    - C++
    - Objective C
  - motivated by simulation
- objects
  - classes
  - inheritance
- Smalltalk
  - motivated by modeling and engineering concerns
  - "pure" OO
- principles
  - open/closed principle
    - modules are closed for modification
    - open for extensiom
  - substitutability principle
    - Wegner & Zdonik
      - incremental reuse
    - Liskov & Wing
  - design by contract
    - pre- and post-conditions
    - class invariants
  - other SOLID principles?
- white box frameworks
  - hollywood principle
    - reverse usual control
  - subclassing contract often implicit
    - hard to learn and adapt
- patterns

Components
- paradigm: configuration of interacting components
  - manage variation
- software product lines
  - domain specific framework for producing a range of related applications
  - key concern: managing variation
  - can use a range of techniques
- components
  - independent unit of deployment
    - Szyperski definition
  - black box frameworks
    - emphasize composition, not refinement
- scripting
  - applications = components + scripts (+ glue)
  - DSLs
    - external
      - special language
    - internal
      - API + idiomatic use of host language
  - Piccola
  - 1 slide summary?
- plugin architectures
  - cf wikipedia article
- SOA
  - web-based composition of services
    - stateless services
  - supports distribution and integration
  - throwback to library-based approaches?

Features
- paradigm: model and compose elementary features
  - features represent domain concepts
- Mixins
  - Flavors
  - "abstract subclass"
  - encapsulate reusable methods and state
  - sensitive to composition order
    - determines conflict resolution
    - source of fragility
- Traits
  - insensitive to composition order
    - explicit aliasing and exclusion
  - purely static
    - can be flattened away
  - Talents
    - dynamic traits for individual objects
- FOP/FOSD
  - transformations add features to base programs
- Generative programming
  - template metaprogramming
- Subject Oriented Programming
  - multi-dimensional separation of concerns
  - like COP, but context is the subject
- COP

Fine-grained adaptation
- paradigm: composition is metaprogrammng
  - separate base and meta-levels
- reflection
  - reflection vs reification
  - introspection vs intercession
  - structural vs behavioural
  - MOPs vs metaobjects vs tower of interpreters
- AOP
  - factor out cross-cutting concerns
  - pointcuts apply aspects to joinpoints in code
  - joinpoints may be static or dynamic
  - AspectJ
- Dependency injection
  - externalize dependencies
  - mechanisms
    - parameterization
    - code generation
    - reflection
      - bytecode adaptation
  - example: injecting mocks for testing
- MDD
  - Models are instances of meta-models
  - Stop with M3 (meta-meta-models)
  - practical application in UML
  - Generate platform-specific applications from platform-independent models
  - use model transformations between levels

Conclusions
- Core paradigm is component composition
  - various levels of granularity
  - static or dynamic
  - base or meta level
- Mechanisms
  - invocation
  - binding
  - code generation
  - message sending
- Programming is modeling

/ Compositional programming

  Dagstuhl 12511 tutorial, Dec 2012

/ Compositional programming / Components / plugin architectures / cf wikipedia article

  http://en.wikipedia.org/wiki/Plug-in_(computing)

/ Compositional programming / Components / SOA

  http://en.wikipedia.org/wiki/Service-oriented_architecture

/ Compositional programming / Features / FOP/FOSD

  http://en.wikipedia.org/wiki/Feature-oriented_programming

/ Compositional programming / Features / Generative programming

  http://en.wikipedia.org/wiki/Automatic_programming#Generative_programming

/ Compositional programming / Features / Generative programming / template metaprogramming

  http://en.wikipedia.org/wiki/Template_meta-programming

/ Compositional programming / Features / Subject Oriented Programming

  http://en.wikipedia.org/wiki/Subject-oriented_programming

/ Compositional programming / Conclusions

  Tutorial for Dagstuhl 12511