Systematic Software Reuse

*It Isn’t What It Used to Be*

Martin Griss, PhD
Principal Research Scientist
Carnegie Mellon University, Silicon Valley Campus
&
Principal, Martin Griss Associates
Agenda

• Background
• From Libraries to Factories
• Generative Reuse
• Agile Reuse
• Conclusions
40 Years Evolving Reuse Practice

- Software portability, LISP compilers, languages - U of Utah
- HP Reuse libraries, corporate reuse program, process
- Software Reuse: From Library to Factory
- (Hybrid) Domain Specific Kits
- UML 1.0 standards committee
- Reuse advice to HP divisions & customers
- RSEB: Software Reuse: Architecture, Process, & Organization for Business Success
- FeatureRSEB, Product Lines
- LogicLibrary, Flashline and TopCoder consulting
- Reuse Comes in Several Flavors
- Study of TopCoder crowdsourcing
- Agile Reuse
Systematic Software Reuse

*Component-oriented software engineering*

A simple idea

Use previously developed components, frameworks, other artifacts

... with complex execution ...

New component & framework & generator technology & methods
Architecture, process, organization, economics, cultural changes

... but with major benefits!

AT&T, GTE, Ericsson, HP, IBM, NEC, Rolls-Royce, Toshiba, Volvo,…
Significant cost and time reductions
Improved agility
Reuse Body of Knowledge

Many books & conferences on reuse & related topics

- Architecture, aspects, patterns, frameworks, components, product lines, generators, domain engineering, management, organization

(C) - MARTIN GRISS - ICSR 2015
Many Reuse Technologies

- Aspects
- Patterns
- Templates
- Parameters
- Components
- Frameworks
- Domain-specific languages
- Generators
- Services/SOA
- Agents

- Library system(s)
- Horizontal vs Vertical reuse
- Domain Engineering
- Variability Analysis
- Reuse-oriented Architecture
- Model-Driven Development
- Product Line Engineering
- Open Source/Corporate Source
- Crowd Source
Many Reuse Questions

• What kind of reuse should we do?
• What strategy of marketing, incentives for reuse?
• What is an appropriate organization model?
• Should we do full scale product line reuse?
• Should we do model-driven development
• Should we use generators and domain-specific languages
• What technologies and tools to focus on?
• How are assets and support funded?
• What kind of reuse pilots to do?
• How and when to scale up?
• How is reuse connected to other software initiatives: architecture, SOA, process improvement, quality, metrics, open source, crowd source, ...
(Staged) Adoption of Reuse

Improved time to market, costs, quality

Reuse Benefits

Investment, experience, time

- Reduced development time
- Informal code salvaging
- Planned black-box code reuse
- Broader coverage
- Interoperability high reuse levels
- Significant management support. Code, other workproducts
- Architected reuse, process metrics
- Rapid custom product development business
- Pervasive domain-specific reuse

No reuse

Reduced maintenance costs

(C) - MARTIN GRISS - ICSR 2015
Reuse May Vary Across Organization

- Ad hoc, random reuse
- Powerful enablers and process enhancements
- Strategic to company success

Components, Libraries

Architecture, Frameworks

Platform, Services
Reuse “Flavors”

1. Facilitated
- Encourage, support, enable individual or team choice

2. Managed Reuse
- Require, enforce, control participation, use of assets

3. Architected Reuse
- Architect, domain engineer assets for reuse, domain

4. Reuse-Driven Business
- Reuse central to all decisions

ad hoc reuse - NONE
Mixing Reuse Flavors

- Facilitated Reuse
- Managed Reuse
- Architected Reuse
- Reuse Driven Business

Governance/Process/ Roles/ Tools
Agenda

• Background
• From Libraries to Factories
• Generative reuse
• Agile reuse
• Conclusions
From LEGO “components” to “kits”
(Hybrid) Domain-Specific “Kit”

Combine compatible asset types

Framework  Components  Glue  Tools  Procedures

(C) - MARTIN GRISS – ICSR 2015
• A set of products sharing common set of requirements (or features), with significant variability
• *Feature* = product characteristic users, customers & developers use in describing/distinguishing members of product-line.
Expressing Variability

Components have *Variation Points* where they can be customized with *variants* using various mechanisms.
RSEB

Product Line Engineering

- Business Model
  - Business processes
- Business priorities
- Application roadmap

- Applications
- Application Family Engineering

- Layered Architecture
  - Component System Engineering
  - Component Systems

- Ranking/Prioritizing
  - Business use cases
  - Application use cases
  - Component systems

- Existing Applications
  - Standards
  - Technology trends
  - Application priorities
  - Customer trends

- Domain Experts
  - Domain Analysis
  - Domain Model
    - Feature Model
    - Domain Architecture
  - Exemplers

- Reengineering
  - Candidate Components
Developing for Application Family

Domain-specific, architected, product-line

Provide: Develop For Reuse
• scope domain
• variability
• architecture
• components & frameworks
• DSL & generators

Utilize: Develop With Reuse
• match to domain
• delta analysis
• select, adapt, integrate
FeatuRSEB

Combine RSEB, FODA, UML

Legend

- Composed of
- Optional feature
- Vp-feature (XOR)
- Vp-feature, use time bound

(C) - MARTIN GRISS - ICSR 2015
FeatuRSEB

```
<feature>
  tone
</feature>

requires

<feature>
P.I.N
</feature>

description: Use P.I.N to id user
source: Exemplar 3
nature: functional
existence: optional
alternative: fixed
category: context
bindingTime: reuse
issuesAndDecisions: Must validate length of P.I.N
notes:

<feature>
Variable
</feature>

description: Implement new input mode
source: Exemplar 2
nature: functional
existence: optional
alternative: fixed
category: context
bindingTime: reuse
issuesAndDecisions: Need guidelines
notes: For example, "dial 1 for ..."
```
Agenda

• Background
• From Libraries to Factories
• Generative reuse
• Agile reuse
• Conclusions
Generative Approaches

• Built-in to Language
  • C/C++ macros, LISP macros, C++ templates, Java Generics, ...

• General Purpose Macro Preprocessor
  • GPM, STAGE2, M4, Basset Frames (NETRON), XVCL, VCL, ..

• Extensible Languages
  • LISP, BALM/MBALM, Algol-68, EL1, ...

• Domain Specific Languages/Kits
  • Via YACC, MetaLISP, BALM,... (e.g., PictureBALM), Visual Programming kit, OO Instrument Kits)

• Model-driven Generators
  • GenVOCA; MetaCASE; OMG MDA (UML for PSM/PIM), ...
  • Aspects, ...
XVCL/Bassett Frame Generator

- XML-based generator
- Template-based DSL
- Easy to layer onto existing software
- Manage commonality and variability
- Weaves code fragments (“aspects”)
- Used for code reuse and product lines

**Legend**
- `adapt`-
- `X`-
- `frame`
- `traversal path`

**Order of Assembly:** (ABDBEBACECFCA)
Aspect-Oriented MDD

AOP, SOP, FOP, XVCL

Withdraw Money
Deposit Money
Cashier Interface
Dispenser
Withdraw
Account
Deposit
Slot
Component
Component
Component
Component

(C) - MARTIN GRISS - ICSR 2015
OMG/UML Model Driven Architecture

- Use UML + <<stereotypes>> + OCL
- Create
  - Problem Independent Model (PIM)
- Generate
  - Problem Specific Model (PSM)
- Transformation rules
Agenda

• Background
• From Libraries to Factories
• Generative reuse
• Agile reuse
• Conclusions
Agile in the Enterprise

*Plan-driven vs Agile vs Hybrid*

- **Conventional plan-driven process**
  - Large teams
  - Standardized models, architecture, documents and process

- **Feature-oriented agile process**
  - Small teams
  - Rapid development
  - Customer-oriented release and evolution
  - Expertise and tacit knowledge
  - Emergent architecture

- **Hybrid approaches**
  - Address scale, reuse, architecture
Approaches to “Agile” Reuse

**Oxymoron? - YAGNI**

Incremental Feature-Oriented Reuse

- Leverage agile feature/story cards, SCRUM backlog
- Feed incremental Feature-Oriented Domain Engineering (FODA, FeatuRSEB)

Leverage Management of Technical Debt

- Technical Debt accumulates with deferred decisions and work, coding shortcuts
- Incrementally pay off debt by re-factoring, re-engineering, re-architecting
- Economic/metrics models to help make decisions

Domain Specific Languages

- Various levels of model-driven development
(New) Sourcing Models

- **Open Source**
  - Varying community and process management

- **Corporate Source**
  - Foster open source “style” in companies

- **Crowd Source**
  - Deliberate engagement of community
Agenda

• Background
• From Libraries to Factories
• Generative reuse
• Agile reuse
• Conclusions
Assess Reuse Readiness

1. Business goals that motivate reuse
   - Time, cost, quality, integration, agility, standards, ...
   - Urgency, importance, champion ...

2. Domain(s) readiness for reuse
   - Stability and variability, standards
   - Obvious, pervasive product line

3. Organizational readiness
   - Culture, process maturity, autonomy, standards
   - Conflicting initiatives, prior history, technology shifts,

4. Reuse experience
   - Current stage or flavor of (systematic) reuse
   - Reuse level, technology use, library use
Conclusions

• Software reuse approaches keep evolving
• Assess reuse readiness before selecting reuse goal and flavors
• Identify opportunities for small DSL/MDD, generators and product-lines
• More work on agile reuse, SEMAT/reuse, open source/crowd source