Introduction to Generative Programming

Krzysztof Czarnecki
University of Waterloo
czarnecki@acm.org

www.generative-programming.org

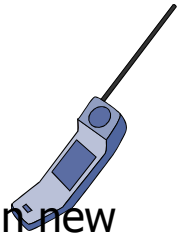
Goals

- What is to be achieved?
  - Basic understanding of Generative Programming (GP)
  - Based on first impressions and discussion you should be able to judge what you can benefit from GP
- How can it be achieved?
  - Overview of the most important steps of GP
  - Demonstration of tools
  - Presentation of case studies
  - Discussion in the plenum

Overview

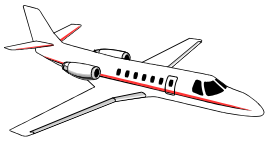
- ➢ Challenges
- Introduction
- First Example
- Further Information

Complexity



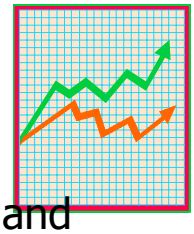
- Software systems are engineering in new and demanding application domains
- Software systems support increasingly complex tasks
- Requirements concerning functionality and efficiency are constantly growing

Quality



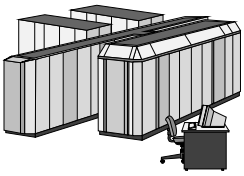
- Software is deployed in life-critical application areas
- Software malfunction causes economic losses
- Usability is a major acceptance criterion
- Users expect a specific quality that is determined by varying criteria

Productivity



- Growing demand for new systems and variants
- Increase of productivity and development staff cannot close the gap between demand and supply
- Standard software becomes more and more important, but it cannot be adapted to every specific requirement

Maintenance & Legacy Systems



- Long lifetime of software systems (sometimes 20 years and more)
- Software must be continuously adapted to new and changing requirements
- Maintenance of legacy systems absorbs development capacity to a high degree
- Software systems of today are the legacy of tomorrow!

Example "Object Technology"

- Theory
  - Productivity and quality should increase thanks to reuse
- Reality
  - Classes are too small as units of reuse
  - Frameworks are sufficiently large as units of reuse, but frameworks from different vendors do not integrate well
  - Design patterns are pieces of reusable knowledge, but they do not exist as executable code

Example "Component Technology"

- Standards and platforms improve interoperability and the distributed execution of components
- Many components provide a special mode for customization
- Reusing small components does not have a large impact on software development while large components require high customization efforts

One of the Causes ...

Currently dominating methods and tools are designed for developing ***single systems***

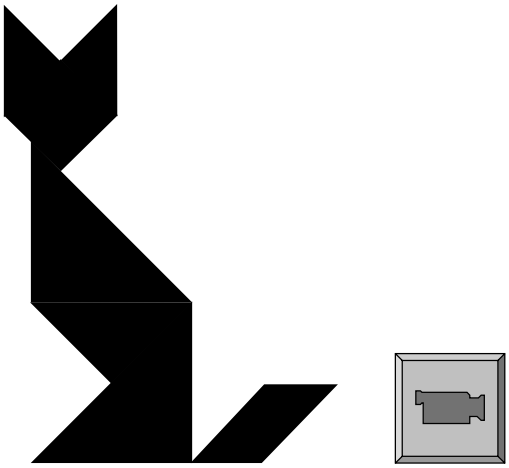
Overview

- Challenges
- ➢ Introduction
- First Example
- Further Information

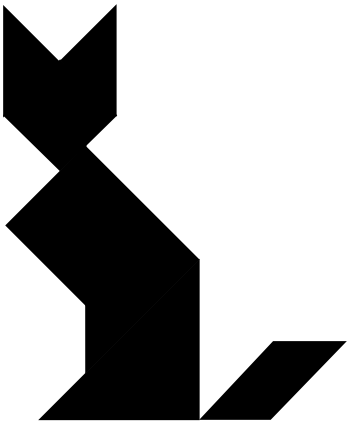
Single System Development

- Focus on analyzing, modeling, and implementing a single system
- Important goal: decomposition into components, i.e. modularization
  - Individual components are easier to maintain, to replace, and to reuse
  - But, what are the criteria for decomposition?

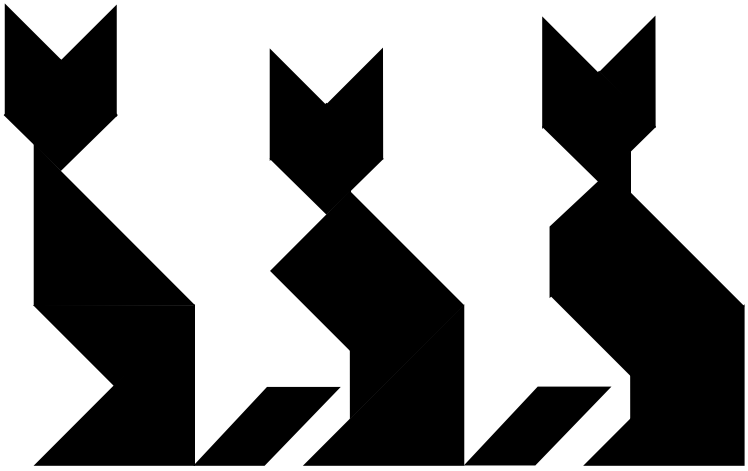
Experiment - Part I



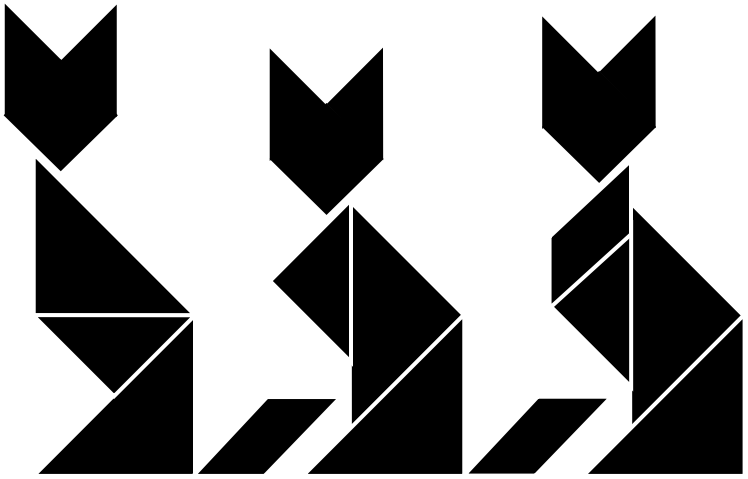
Experiment - Part II



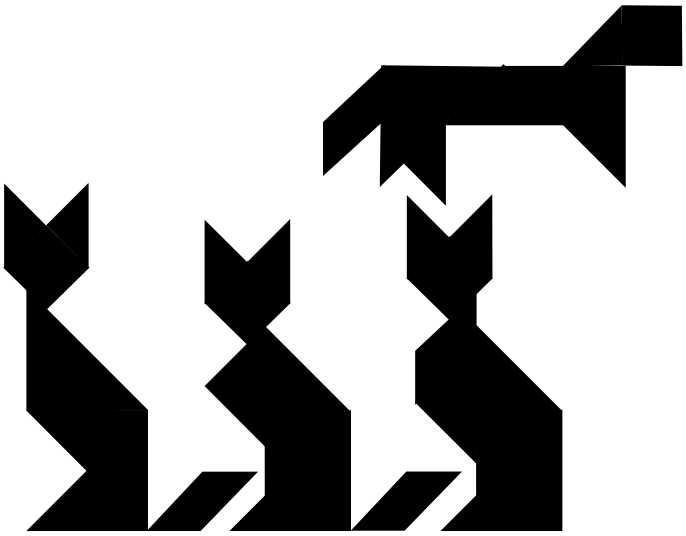
Exemplar Systems of a Family



The "Right" Modules



An Additional System ...



... Can Cause Surprise!

How To Order Systems?

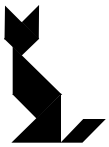
- By Components
  - Downward arrow on top
  - Large triangle below with its 90° angle at the lower left point
  - …

- By Specification
  - A cat
  - Sitting upright
  - Waiving its tail

From Order To Product

Creating the order
- Expert language
- Form, catalogue
- By example

Processing the order
- Completion
- Validation
- Construction rules
- Optimization



Product generation
- Assembling elementary, reusable components

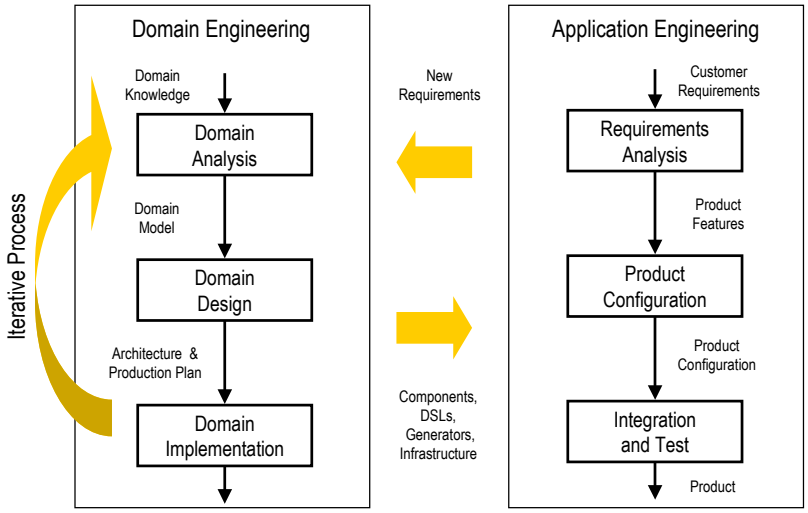
Generative Programming ...

... is a software engineering paradigm based on modeling software system families such that, given a particular requirements specification, a highly customized and optimized intermediate or end-product can be automatically manufactured on demand from elementary, reusable implementation components by means of configuration knowledge

Two Processes

- Domain Engineering
  - Analysis: Domain scoping and defining a set of reusable, configurable requirements for the systems in the domain
  - Design: Developing a common architecture for the systems in the domain and devising a production plan
  - Implementation: Implementing the reusable assets, for example, reusable components, domain-specific languages, generators, a reuse infrastructure, and a production process
- Application Engineering
  - Producing concrete systems using the reusable assets developed during Domain Engineering.

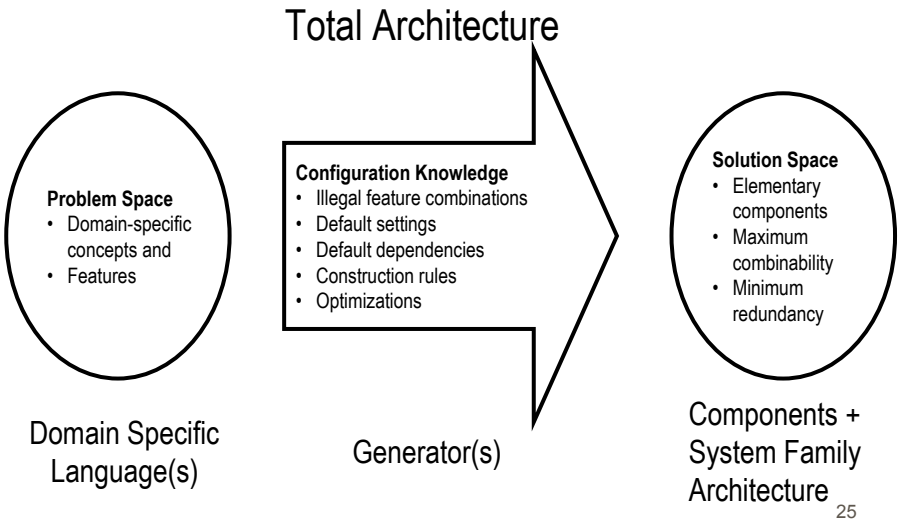
System Family Approach



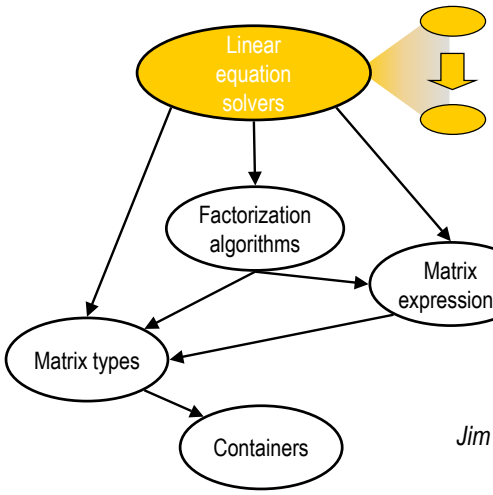
Maturity Levels in Domain Engineering

- Domain Engineering can be applied at different levels
  - Domain analysis
  - Reference architectures and architectural patterns
  - Design patterns and OO frameworks
  - Components and component frameworks
  - Generation of parts of application code
- Generative Programming aims at the highest level of automation of application engineering
  - Domain-specific languages and automatic configuration of components

Generative Domain Model



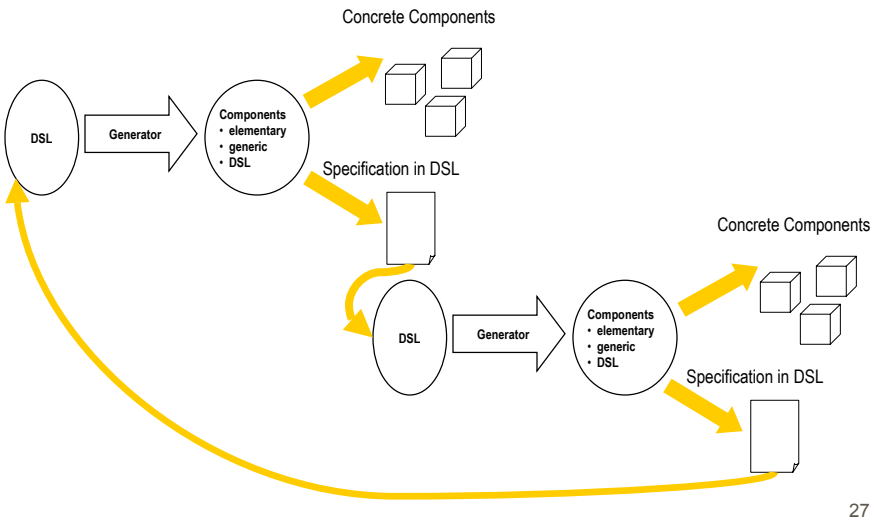
Domain Network



- Each subdomain is implemented as a generative domain model
- Higher order components

Jim Neighbors, Draco, 1980

Federated Generators



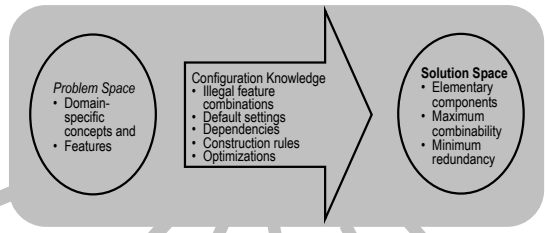
A Technology Projection ...

... is a mapping of the generative domain model onto

- other software development paradigms,
- a programming language,
- several development tools that are combined within one environment or on one platform

- Meanwhile, several technology projections are available

Existing Technology Projections



C++
- Template meta-programming
- Static configuration
- Language-specific means only

Java and JavaBeans
- Interactive parameterization of JavaBeans
- Dynamic configuration
- Language-specific means only

Java und AspectJ
- Using an AspectWeaver
- Static configuration
- Separate preprocessing step

ANGIE
- Employment of a frame processor
- Static configuration
- Separate preprocessing step

Small Components
- Usage of an XML configuration file
- Static and dynamic configuration
- Separate preprocessing step

XML, TL and Ada 83
- Graphically-interactive configuration
- Static configuration
- Separate preprocessing step

Essential Terms

In this presentation

- Domain Analysis is part of Domain Engineering
- A product family comprises the variants of a product
- A product line comprises multiple products that are targeted for a specific group of customers
- A product line can be based on system families

Different uses

- Domain Analysis precedes Domain Engineering
- Product family and product line are synonyms

Essential Terms

- Generic
  - *"relating to or characteristic of a whole group or class"* (Merriam-Webster Online)
  - Solution space technique for developing parameterizable components
- Generative
  - *"having the power or function of generating, originating, producing, or reproducing"* (Merriam-Webster Online)
  - System for producing other systems; it comprises problem space, configuration knowledge, and solution space

Scope

- Generative programming can be applied at any level of granularity – from individual functions and classes to large software systems
- Generative programming is not limited to implementing application code, testing, and creating GUIs
- System family engineering also includes
  - documentation
  - system installation
  - user training
  - maintenance, and many more