SOFTWARE PRODUCT LINES:
A TECHNIQUE FOR BUILDING
A RESEARCH INFRASTRUCTURE

John D. McGregor
Clemson University
Motivation

- Faculty and students develop a large amount of software.
- For faculty this is an on-going effort; for a student there is a terminal objective – a degree.
- For both, the algorithm is the immediate objective; the code is secondary.
- The student leaves and the software sits.
- Our hypothesis is that with a small delta in effort a research infrastructure that will be used for numerous projects can be built.
Professor has a product line of research results captured in code. The programs have much in common, but they do vary from each other.

Over time the commonality increases as does its usefulness. Every new effort can start from where others left off.
Multi-Scale Modeling of Single Ventricle Hearts for Clinical Decision Support

The Trans-Atlantic Network of Excellence for Cardiovascular Research

In this program, we will integrate our expertise in pediatric cardiology, surgery, imaging, engineering, and computer science to develop a modeling system that can assist and support the clinical management of complex congenital heart disease (CHD). Developing accurate and clinically relevant modeling is necessary in CHD for two reasons. Firstly, our patients are heterogeneous, and modeling systems that would enable clinicians to simulate virtual operations using patient-specific data in a validated, accurate...
Informally...

- The goal of the software product line strategy is to establish a production capability that can
  - rapidly and accurately produce multiple products within a well-defined scope
  - achieve specific business goals that can be affected by the way the organization produces products.
- A faculty member has a long-range plan that will be realized in several programs.
Informally ...

- We do this by
  - Designing a software infrastructure in anticipation of differences among the multiple products
  - Structuring the research and the research group to take advantage of common interests
  - Use the instructional program to contribute to the infrastructure
A systematic review is more rigorous than a traditional literature review and attempts to reduce the influence of bias in a number of ways:

- It addresses a clearly formulated question
- It uses systematic and explicit methods to identify, select and critically appraise relevant research
- It uses systematic and explicit methods to collect and analyze data from the studies that are included in the review
- Statistical methods (meta-analysis) may or may not be used to analyze and summarize the results of included studies which are considered similar enough to combine.
Defining the problems to be solved

- Defining “the” features needed to answer a research question is not easy but ...
- Features come from systematic review
- Scoping a software product line defines the features that will be shared by a set of products rather than just one product.

http://www.sei.cmu.edu/productlines/ppl
It is a reuse strategy, but on steroids

- A software product line organization builds and (re)uses assets, but the assets are all targeted at products within the well-defined scope.
- We will reuse anything (code, documentation, tests,...) as long as it is useful in building the products within the scope of the product line.
- Our goal is not reuse, our goal is to produce products quickly and economically.
Goal

- Getting answers as quickly and cheaply as possible is the primary goal (unless you just like writing code)
- By building products using assets in inventory, a software product line organization can more quickly assemble a product, provided it is within the scope of the product line.
- Cummins Engines reduced their time to develop the software for a new engine from about 12 person months to as little as 1 week
- Each new project begins with a delta on the systematic literature review
Some real numbers

- Improved productivity
  - by as much as 10x
- Increased quality
  - by as much as 10x
- Decreased cost
  - by as much as 60%
- Decreased labor needs
  - by as much as 87%
- Decreased time to market
  - by as much as 98%
A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

A frequent misconception is that the core assets, the reusable pieces, are the product line. As you can see from the definition, the product line comprises the products.
Product Line Definition - 1

- set of software-intensive systems
  - The product line is the products
  - The product line organization produces the products

- Set of airline reservation systems
- Software controllers for diesel engines
- Ground satellite control software systems
Product Line Definition - 2

- **common, managed set of features**
  - Common – identifying ahead of time common features of the products and the variations in products
  - Managed – evolution is anticipated, variation is controlled, and the inventory of features is what we sell

- **Data storage and management actions**
- **Image analysis techniques**
- **Information classification techniques**
Product Line Definition - 3

- particular market segment or mission
  - Focusing makes the percentage of commonality higher
  - The culture of the market segment determines specific quality levels

- Medical devices
- Video games
Product Line Definition - 4

- common set of core assets
  - A “core” asset is anything used to produce multiple products
    - Source code – yes, but also
    - Software architecture
    - Test infrastructure, test cases, and test data
    - Production plans
    - ....
  - The assets are designed to handle the range of variability defined in the product line scope
  - Each asset is accompanied by an attached process, which explains how to use the asset in building a product
- Implementation of doppler compensation algorithms
- Test scenarios for engine controllers
prescribed way

- A production strategy coordinates the business goals with the development of core assets and products
- A production plan describes the way in which each product is to be produced

- Architecture-centric management
- Traditional programmer-centric code development
- Model-driven automatic code generation
The payoff

- Initiating a software product line strategy requires some amount of up-front investment although it can be minimal.
- If the commonality is sufficiently high, payback can happen after a relatively small number of products.
- Many organizations have reached the payoff point.
Arcade Game Maker Pedagogical Product Line

The Arcade Game Maker (AGM) product line is an example product line created to support learning about and experimenting with software product lines. The product line encompasses three simple arcade games. The primary goal has been to be comprehensive rather than focus on high-quality graphics. The material available follows the basic product line approach described in [Clements 02].

This example has been incrementally created over 2 years with a great deal of evolution. This site, and the product line, is evolving. As the product line is used to illustrate topics in various courses, that material is added to the site. The initial release captures the current state of the product line artifacts. As time permits historical artifacts will be added to show the evolution from a single product example to a product line example.

The Arcade Game Maker product line is a simple, but comprehensive, example. Arcade Game Maker is a fictitious company. A context document is provided that describes meta-information about the organization, its personnel, and the general setting for the example. This information is important to understanding the decisions made at various points in building the product line.

The example has two distinct parts:

- The actual product line assets and products
  - business case,
  - scope,
  - concept of operations,
  - requirements,
  - architecture,
  - production plans,
  - test plans, and
  - code assets for three products.
- A pedagogical section that includes:
  - class-tested pedagogical elements of the product line, and
  - suggested exercises using the assets of the product line.

The complete AGM product line example is provided as part of a web-based set of documents. Learning activities that are based on the product line are available upon request.
Initial work

- Faculty member defines a basic infrastructure by
  - Creating a central Configuration Management repository
  - Identifying standard software projects
  - Writing some glue code

- Faculty member and students form affinity groups
  - Faculty member — mentors - PhD student
    - McGregor — Tacksso Im
  - PhD student — mentors - MS students
    - Tacksso Im — Soujanya Vullam and Lakshmi Kothapalli
How’s it done?

- Essential activities
  - Core asset development
  - Product development
  - Management
Core asset development

- What can we profitably reuse?
  - How many products will use it?
  - How much extra will it cost to make it reusable?
- We reuse ANYTHING that makes sense (money)
  - Source code – obviously – but non-software assets also
  - For example, we decompose a test suite into individual test cases, then compose as needed by a product
- Often a team is devoted to providing these assets
- This team has a vision that encompasses all products that would use its assets.
- An “attached process” accompanies each core asset to facilitate reuse of the asset

A student/faculty member is responsible for identifying new “chunks” of code that should interest others. That is documented and checked in.
Product development

- Product development is combining core assets with product-specific artifacts to produce products.
- Product development moves faster than in traditional development because of the assets and the small percentage of product-specific artifacts.
- A product team may continue to own the product it has built or it may hand it off to a maintenance team.

- This team is focused on one product.

Each student is responsible for designing within the guidelines, documenting code, and checking into the CM system.
Management

- A central authority, such as a product line manager, oversees the organization which may cut across multiple business unit boundaries.
- Coordinates the production of core assets and the assembly of products.
- Ensures that resources are available at the right time to optimize operation of the production capability.

A faculty member provides broad direction to fill in missing pieces or to take the next logical step.
Activities and practices

- **Carrying out the activities touches 29 practice areas**

  - **Software Engineering**
    - Architecture Definition
    - Architecture Evaluation
    - Component Development
    - Mining Existing Assets
    - Requirements Engineering
    - Software System Integration
    - Testing
    - Understanding Relevant Domains
    - Using Externally Available Software

  - **Technical Management**
    - Configuration Management
    - Make/Buy/Mine/Commission Analysis
    - Measurement and Tracking
    - Process Discipline
    - Scoping
    - Technical Planning
    - Technical Risk Management
    - Tool Support

  - **Organizational Management**
    - Building a Business Case
    - **Customer Interface Management**
    - Developing an Acquisition Strategy
    - Funding
    - Launching and Institutionalizing
    - **Market Analysis**
    - Operations
    - Organizational Planning
    - Organizational Risk Management
    - Structuring the Organization
    - Technology Forecasting
    - Training
Software architecture

- Perhaps the key core asset
- Captures early decisions about solving the problem
- Communication vehicle among the stakeholders
- Explicitly addresses the quality attributes
  - Reliability
  - Security
  - Dependability
The product line architecture is the architecture for a family of systems. Is more abstract, not everything is completely defined. There are holes in its specification, but the architecture constrains how the holes can be filled. This may be a fairly abstract architecture but one that will fit most of the software being created.
Variation

- Products vary from one another in specific ways - the allowable contents of the holes in the architecture.
- Strategic variations at the business unit’s level.
- Tactical variations at the technical manager’s level
- Variation points at the implementation level.

How is each research project different?
What is needed for each?
Commonality/Variability Analysis

- What do the products in the product line have in common?
- How are they different?
- A configuration is a selection of inclusive and exclusive OR feature choices to completely define a single member of the product line.
Each hole is plugged by a specific variant determined by the features selected.
What happens now

- Currently using this approach locally and globally
- I am open to collaboration on this research approach.
- I can provide advice about the mechanics.
Resources

- Software Product Line Conference www.splc.net and the associated group on LinkedIn.com
- Software Product Line group at www.LinkedIn.com
- My Strategic Software Engineering column in JOT www.jot.fm
- Catalog of software product lines http://www.sei.cmu.edu/productlines/casestudies/catalog/
- Technical literature at www.sei.cmu.edu/productlines
- Software product line tools http://www.splot-research.org/
Summary

- SPL is not a silver bullet, but it is an industry-proven strategy that produces dramatic results in exchange for hard work and discipline.
- Send questions and comments to johnmc@cs.clemson.edu
Questions?