


## Software Product Lines: Past, Present, and Future

Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213

Paul C. Clements  
SBCARS 2007  
31 August 2007

 Software Engineering Institute | Carnegie Mellon © 2007 Carnegie Mellon University

## Software Engineering Institute


**Who We Are:** Applied R&D laboratory situated as a college-level unit at Carnegie Mellon University, Pittsburgh, PA, USA


**Purpose:** Help others make measured improvements in their software engineering practices

**First objective:** Accelerate the introduction and widespread use of high-payoff software engineering practices and technology identifying, evaluating, and maturing promising or underused technology and practices.

We are a small organization. We have to pick these practices carefully.

One we have chosen is [software product lines](#).



 Software Engineering Institute | Carnegie Mellon Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University 2

## What is a Software Product Line?

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.

- Product lines in manufacturing have been around for centuries.
- Software product lines have not.



## Why software product lines?

Because of the high-payoff results they make possible in practice:

Improved productivity  
by as much as 10x

Decreased time to market  
by as much as 10x

Decreased cost  
by as much as 60%

Decreased labor needs  
by as much as 10X fewer software developers

Increased quality  
by as much as 10X fewer defects



## Real-world examples

Successful software product lines have achieved these results

- across multiple domains
  - Mobile phones
  - Command and control ship systems
  - Ground-based spacecraft systems
  - Avionics systems
  - Pagers
  - Engine control systems
  - Billing systems
  - Web-based retail systems
  - Printers
  - Consumer electronic products
  - Acquisition management enterprise systems
  
- in large organizations and small
  
- in Government and private sectors



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

5



Software Product Lines:  
The Past



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

6



## Product Lines in Building/Manufacturing

Product lines – building a family of products from interchangeable parts – have existed for centuries.

A thousand years ago, Li Chieh, the state architect of the Chinese emperor Hui-tsung, published a set of building codes for official buildings.

- Standard parts and ways to connect the parts
- Parameterized variations: lengths, loads
- Options for components
- Finishing with brackets, decorations

This book defined a set of reusable designs: a “product line” of buildings.



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

7

## Product Line of Computers, Circa 1964

In 1964, IBM introduced the System/360 family of computers. Each model differed widely in performance and features, but they all ran the same programs. The operating system (called OS/360) was also a family.



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

8



### Product Line of Commercial Aircraft, 1980s



Boeing 757,  
Boeing 767

These two very different aircraft were designed together and have about 60% of their parts in common.

[www.boeing.com](http://www.boeing.com)

 Software Engineering Institute | Carnegie Mellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

9

“A common type-rating enables pilots qualified to fly any 757 or 767 to fly all the others with minimal additional familiarization, saving training time and costs. The 757/767 allows crews to fly more models with a greater variety of fuselage widths, capacities, and range capabilities than any other common-rated family.”



[www.boeing.com](http://www.boeing.com)


 Software Engineering Institute | Carnegie Mellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

10

### A Similar Example: Airbus A-318/319/320/321 Family

- 100-200 seats
- 103-146 feet in length
- Same wing
- Same flight deck



[www.airbus.com](http://www.airbus.com)

Software Engineering Institute | Carnegie Mellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

11

### A Nearby Example:

### Embraer ERJ 135/140/145/145XR

70-120 seats




[www.embraercommercialjets.com](http://www.embraercommercialjets.com)

Software Engineering Institute | Carnegie Mellon


Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

12


### A Product Line of Restaurant Menu Items




Hamburger




Cheeseburger



Dbl. hamburger




Dbl. cheeseburger



Veggie burger

Do you see the  
*components, the  
architecture, and  
the reuse in these  
products?*

Source: [www.burgerking.com](http://www.burgerking.com)



Software Engineering Institute | CarnegieMellon


Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University


13

### Questions

What does a company gain from producing a product line of sandwiches? Of commercial passenger aircraft?

- Simplified production of components
- Simplified management of inventory
- Simplified training
- Streamlined production facilities and process
- Market recognition and mind share
- Flexibility – ability to add new products quickly





Software Engineering Institute | CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

14

## What about *Software Product Lines*?

Software engineers have long known about interchangeable parts.

We knew it was better to avoid designing and building software from scratch.

In other words, it seemed that *reuse of components* was the way to achieve greater productivity.

Long ago we started experimenting small-scale reuse.

- Subroutines were the first example.
- We recognized that different systems shared common parts.



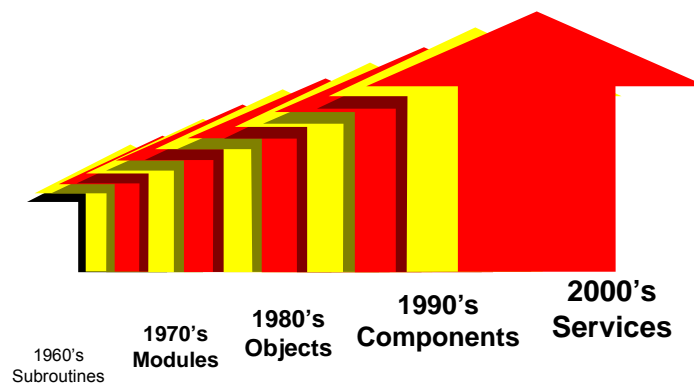
Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

15

## But reuse has a disappointing history.



Focus was small-grained and opportunistic.  
Results always fell short of expectations.



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

16







*“If you build it...  
... they will come.”*

It turns out they don't.

The *opportunistic* model  
of reuse has largely been  
a failure.





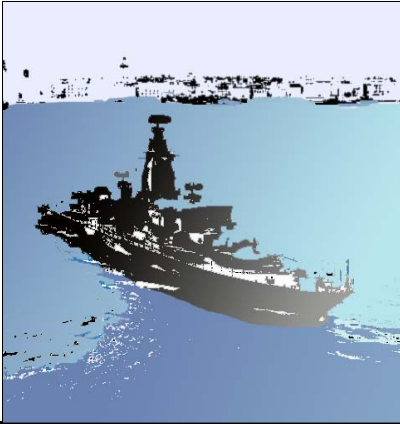
Software Engineering Institute | Carnegie Mellon


Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

### Meanwhile, some companies were trying a new approach

1985: CelsiusTech's ShipSystem 2000 is launched – a family of naval command-and-control systems

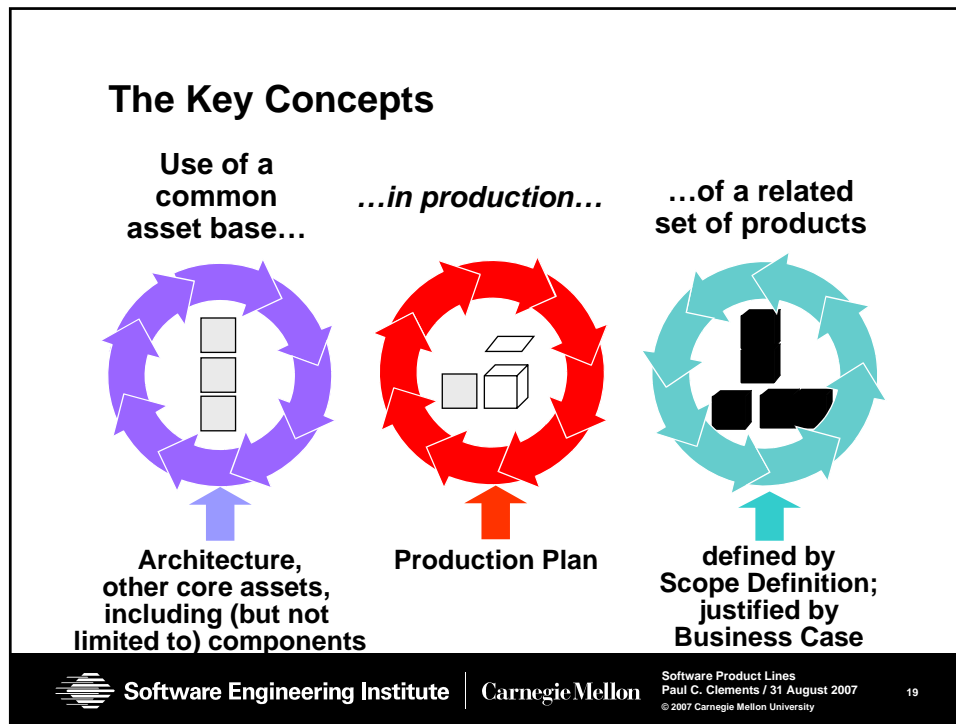
- Began in 1985 with two systems
- Built from single core asset base
- Hardware-to-software cost ratio changed from 35:65 to 80:20
- Software staff went from 210 to 30
- Cycle time went from 7-10 years to 2-3.
- Reuse ~ 90%
- Integration test of 1-1.5 million SLOC requires 1-2 people
- Re-hosting to a new platform/OS takes 3 months
- Cost and schedule targets are predictably met
- Customer satisfaction is high
- Over 55 versions in the family





Software Engineering Institute | Carnegie Mellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University



### What happened next?

Other companies were trying this approach

- On varying scales
- With varying success

Topics of interest at the time

- Domain analysis
- Software architecture
- Process improvement
- Component-based software development

- In 1996, the SEI published the CelsiusTech case study.
- In 1997, the SEI held its first product line workshop.
- In 2000, the first International Software Product Line Conference was held.

Software Engineering Institute | CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

20

## Knowledge began to be codified

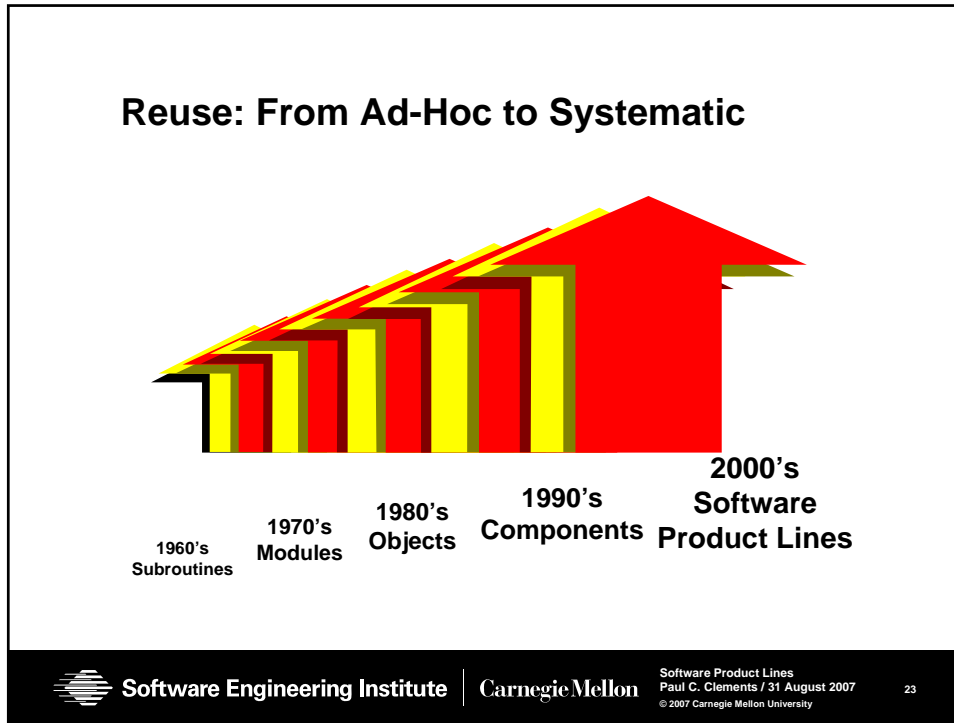
1999 2002 2005 2007

Software Engineering Institute | Carnegie Mellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

## Other Forces Fell Into Place

- Rapidly maturing, increasingly sophisticated software development technologies including *object technology*, *component technology*, *standardization of commercial middleware*.
- A global realization of the *importance of architecture*
- A universal recognition of the need for *process discipline*.
- *Role models and case studies* emerging in the literature and trade journals.
- *Conferences, workshops, and education programs* including product lines in the agenda.
- Company and inter-company *product line initiatives*.
- Rising recognition of the *amazing cost/performance savings* that are possible.



## What Do We Have Now?

Software product lines have emerged as an important, viable paradigm for software development.

- Convincing evidence that software product line practice can bring about significant improvements in software development
- A body of knowledge and a set of standard models for software product lines
- A growing and energetic community of software product line practitioners



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

25

## National Reconnaissance Office / Raytheon: Control Channel Toolkit

Ground-based spacecraft command and control systems

Increased quality by 10X

Incremental build time reduced from months to weeks

Software productivity increased by 7X

Development time and costs decreased by 50%

Decreased product risk



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

26



## Cummins, Inc.

World's largest manufacturer of large diesel engines.

- 20 product groups launched, yielding over 1000 separate engine applications
- 75% reuse, 360% productivity gain
- Product cycle time has plummeted. Time to first engine start went from 250 person-months to a few person-months.
- Software quality is at an all-time high.
- Product line approach let them quickly enter and then dominate the *industrial* diesel engine market.



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

27

## Nokia Mobile Phones

Product lines with >30 new products per year. Before product line, only 5-10 products per year.

Software product line is so successful, Nokia is selling their core asset base as a product.

Across products there are

- varying keys, displays, features, protocols
- 58 languages and 130 countries
- need for backwards compatibility
- need for product behavior change after release



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

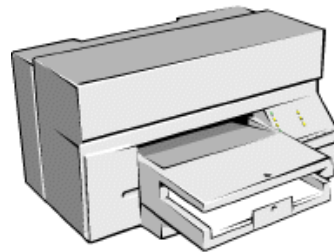
28



## Hewlett Packard

### Printer systems

- 2-7x cycle time improvement (some 10x)
- 400% productivity improvement
- Sample Project
  - shipped 5x number of products
  - that were 4x as complex
  - and had 3x the number of features
  - with 4x products shipped/person



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

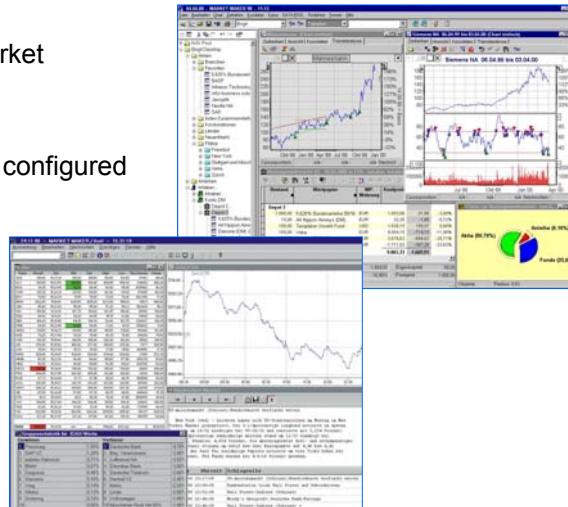
29

## Market Maker GmbH: MERGER

Internet-based stock market software

Each product “uniquely” configured

Three days to put up a customized system



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

30



## 2007: Catalog of Over 40 Published Examples

| Company                   | Software Product Line(s)  | Company            | Software Product Line(s)  |
|---------------------------|---|--------------------|---|
| AKVAsmart ASA             | Feed control and fish farm mgt. software  | Nokia              | Mobile phones   |
| Argon Engineering         | Various   | Nokia              | Mobile Browsers   |
| Asea Brown Boveri (ABB)   | Gas turbines  | Nokia Networks     | Telecommunication network products for public, private, and cellular networks |
| Axis Communications AB    | Train control; Computer printer servers, storage servers, network camera and scanner servers. | Nortel             | Digital loop carriers for telecommunications                                  |
| Boeing                    | Avionics  | Philips            | High-end televisions  |
| CelsiusTech               | Naval command and control   | Philips            | PKI telecommunications switching system                                       |
| Cummins                   | Diesel engine controls  | Philips Medical    | Diagnostic imaging equipment  |
| Deutsche Bank             | Financial global transaction/settlement   | Raytheon/U.S. NRO  | Satellite ground control station software                                     |
| Dialect Solutions         | Internet payment gateway infrastructures  | Ricoh              | Office appliances   |
| DNV Software              | Software products and customized solutions for transportation industries                      | Robert Bosch Corp. | Automotive gasoline systems   |
| E-COM Technology Ltd.     | Medical imaging workstations  | Rockwell Collins   | Commercial flight control system avionics                                     |
| Enea                      | Real-time OS for telecom and automotive applications and middleware                           | Rockwell Collins   | Avionics for U.S. Army helicopters  |
| Ericsson                  | Telecommunications Switches   | Salion, Inc.       | Revenue acquisition management systems  |
| Ericsson Mobile Data      | Packet based mobile communication   | Securitas Larm AB  | Safety and security systems   |
| General Motors P'train    | Engines/transmission/controllers s/w  | Siemens            | Software for viewing and quantifying radiological images                      |
| Hewlett Packard           | Firmware for computer peripherals   | Symbian            | EPOC operating system   |
| LG Industrial Systems     | Elevator control systems  | Telvent            | Industrial supervisory control and business process management systems        |
| LSI Logic - Engenio Group | RAID controller firmware for disk storage   | Testo              | Climate and flue gas measurement devices                                      |
| Lucent Technologies       | SESS telecommunications switch  | U. S. Navy         | Test range facilities   |
| Market Maker Software     | Stock market data / financial news mgt.   | U.S. Army          | Command and control simulator for fire support                                |
| NASA Jet Propulsion Lab   | Interferometer product line   | Unnamed            | Legal expert systems  |



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

31

## What Do We Have Now?

Software product lines have emerged as an important, viable paradigm for software development.

- Convincing evidence that software product line practice can bring about significant improvements in software development
- • A body of knowledge and a set of standard models for software product lines
- A growing and energetic community of software product line practitioners



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

32






### Information Sources


Case studies, experience reports, and surveys

Workshops and conferences



Applied research


Collaborations with customers on actual product lines

 Software Engineering Institute | CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

33


### Three Essential Activities



Core Asset Development

Product Development

Management

 Software Engineering Institute | CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

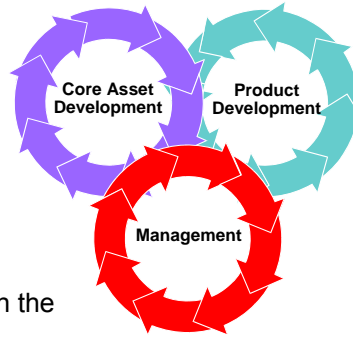
34

## The Nature of the Essential Activities

All three activities are interrelated and highly iterative.

There is no “first” activity.

- In some contexts, existing products are mined for core assets.
- In others, core assets may be developed or procured for future use.



There is a strong feedback loop between the core assets and the products.

Strong management at multiple levels is needed throughout. Management orchestrates the processes to make the three essential activities work together



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

35

## Building the core asset base

(proceed when ready)

Core assets include:

Requirements and requirements analysis

Domain model

Software architecture

Performance engineering

Documentation

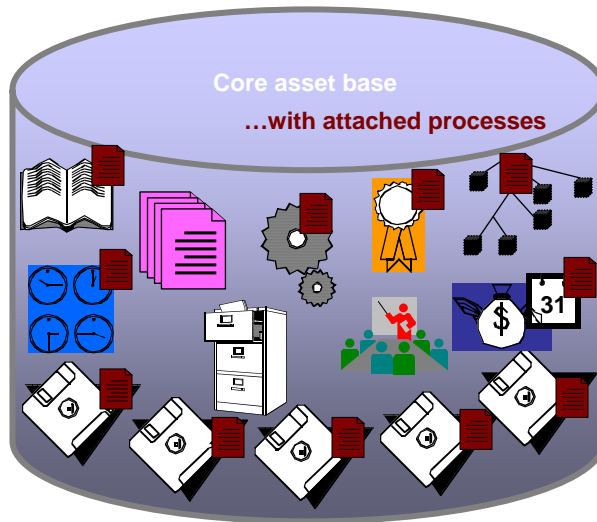
Test plans, test cases, and data

People knowledge and skills

Processes, methods, and tools

Budgets, schedules, workplans

...and Software



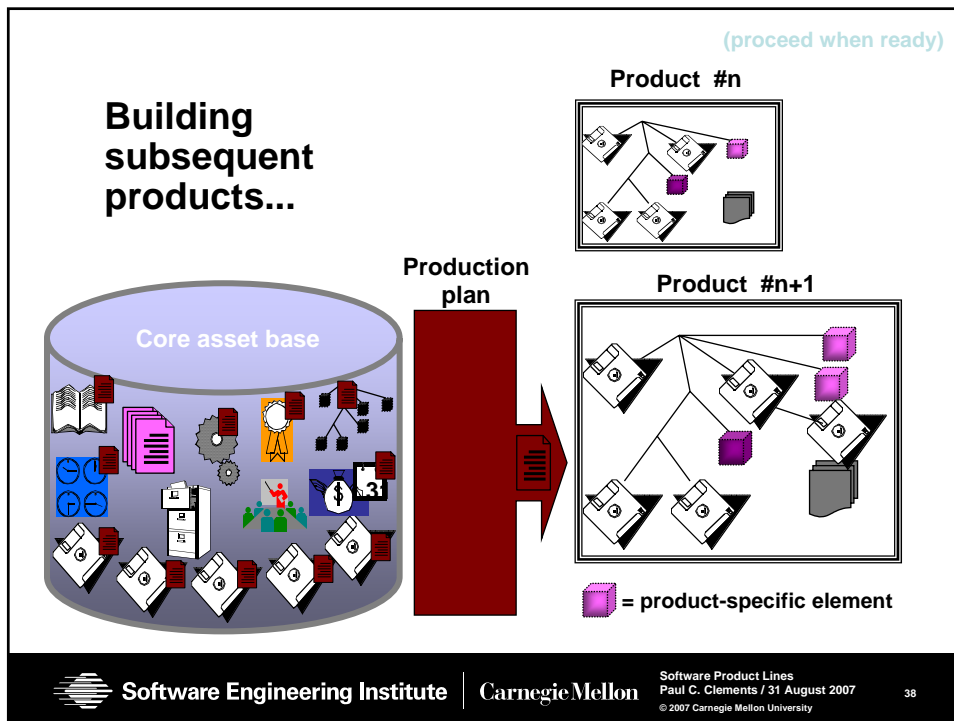
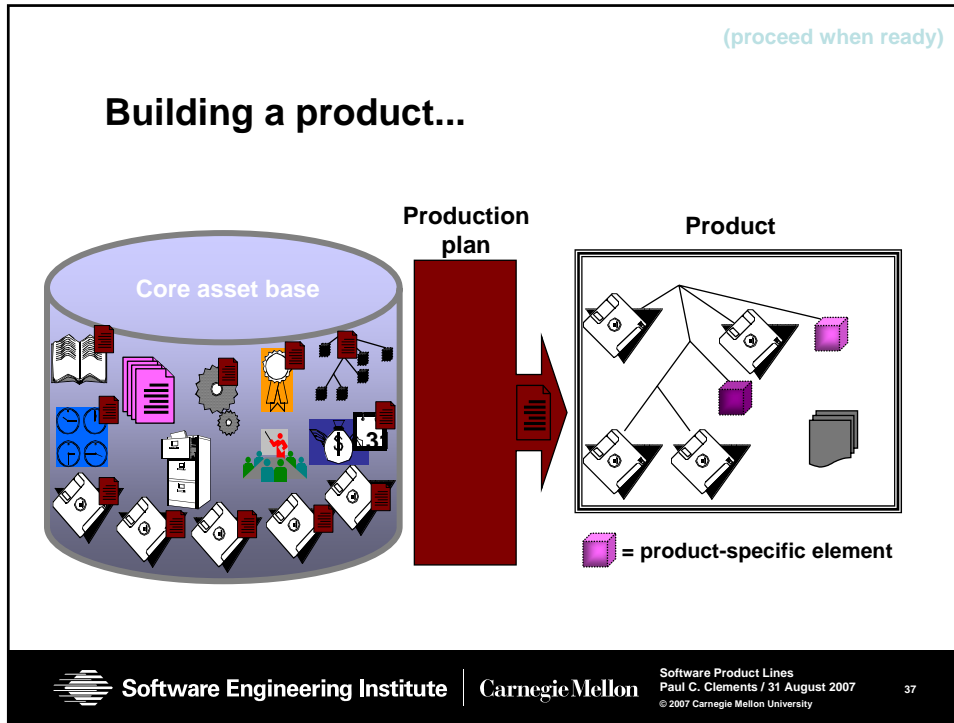
Software Engineering Institute

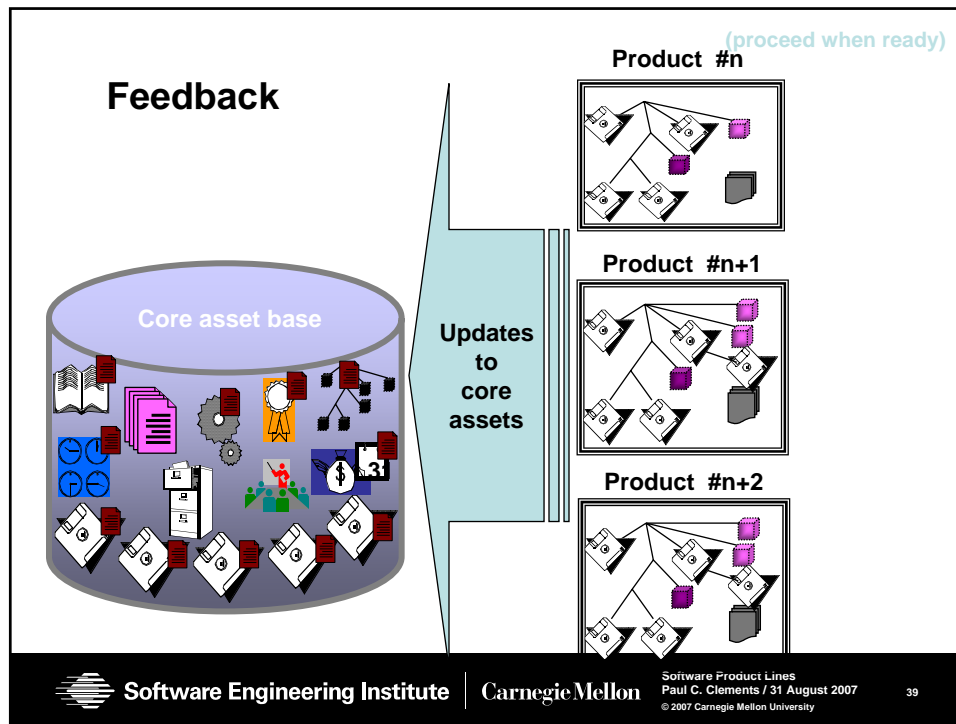
CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

36







## Product line experience has yielded important lessons

### Lessons in software engineering

- Architectures for product lines
- Testing variable architectures and components
- Importance of having and capturing domain knowledge
- Managing variations
- Important of large, pre-integrated chunks

### Lessons in technical/project management

- Importance of configuration management; why it's harder for product lines
- Product line scoping: What's in? What's out?
- Tool support for product lines

### Lessons in organizational management.

- People issues: how to bring about change, how to launch the effort
- Organizational structure: Who builds the core assets?
- Funding: How are the core assets paid for?
- Interacting with the customer has whole new dimension

### SEI Framework for Product Line Practice

We've distilled these lessons into *practice areas*, areas of expertise necessary for engineering any software system, but which take on a different flavor in a product line context. These form a *framework for product line practice*.

| Software Engineering  | Technical Management  | Organizational Management  |
|---|---|--|
| Architecture Definition<br>Architecture Evaluation<br>Component Development<br>COTS Utilization<br>Mining Existing Assets<br>Requirements Engineering<br>Software System Integration<br>Testing<br>Understanding Relevant Domains | Configuration Management<br>Data Collection, Metrics, and Tracking<br>Make/Buy/Mine/Commission Analysis<br>Process Definition<br>Scoping<br>Technical Planning<br>Technical Risk Management<br>Tool Support | Building a Business Case<br>Customer Interface Management<br>Implementing an Acquisition Strategy<br>Funding<br>Launching and Institutionalizing<br>Market Analysis<br>Operations<br>Organizational Planning<br>Organizational Risk Management<br>Structuring the Organization<br>Technology Forecasting<br>Training |

### What Do We Have Now?

Software product lines have emerged as an important, viable paradigm for software development.

- Convincing evidence that software product line practice can bring about significant improvements in software development
- A body of knowledge and a set of standard models for software product lines
- • A growing and energetic community of software product line practitioners

## We now have a *community* of software product line practitioners

### We seek to

- understand the principles and practices behind software product engineering.
- help others successfully adopt the paradigm
- share experience

### Major forums, workshops and conferences

- Software Product Line Conferences (SPLC 1-4)
- Product Family Engineering (PFE 1-5)
- **Merged in 2004 to become SPLC / SPLC-Europe**
- **First SPLC-Asia happens September 10-14 2007, Kyoto**

### Community web sites

- [www.sei.cmu.edu/productlines](http://www.sei.cmu.edu/productlines)
- [www.softwareproductlines.com](http://www.softwareproductlines.com)




Software Engineering Institute


CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

43



11th International  
**Software Product Line Conference (SPLC 2007)**  
Sept. 10-14, 2007, Kyoto, Japan



**Software Product Lines -  
Power for Competitive Advantage**

Home

Keynotes  
Program  
Tutorials  
Workshops  
Panels  
Registration  
Local Information

CFP  
Submissions  
Demonstration  
Doctoral symposium  
Product Line Hall of Fame

Organization  
Program Committee  
Sponsor

call-for-participation (pdf)  
call-for-participation (txt)

[日本語による情報](#)  
Information in Japanese

The current trend of globalization is pressuring industries to explore ways to meet diverse needs of the global market most effectively and efficiently. Over the last decade or so, software product line has emerged as one of the most promising software development paradigms in drastically increasing the productivity of IT-related industries, and the product line community has grown and is still growing rapidly. At this very important juncture, the Software Engineering Center (SEC) of Information-Technology Promotion Agency (IPA), Japan is proud to sponsor the Eleventh International Software Product Line Conference (SPLC 2007) in Kyoto, Japan, the first ever software product line conference held in Asia.


SPLC is the most prestigious and leading forum for researchers, practitioners, and educators in the field. SPLC 2007 will provide a venue for exchanging, sharing, and learning technologies and industrial experiences to the community. The conference will feature research and experience papers, tutorials, workshops, panels, and demonstrations. Especially, we will organize special workshops to address the needs of specific industrial segments, including automobile, mobile communications, embedded controllers, and enterprise software.

We encourage you to submit technical papers, and proposals for panels, tutorials, workshops, and demonstrations. We look forward to interacting with you at SPLC


**New!**

June. 15, 2007  
[Registration site open!](#)

May. 31, 2007  
[Doctoral Symposium deadline extended to June 10.](#)



Software Engineering Institute



CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

44



## SEI Software Product Line Curriculum

Five one- and two-day courses

- Software Product Lines
- Adopting Software Product Lines
- Developing Software Product Lines
- Product Line Technical Probe Team Training
- Product Line Technical Probe Leader Training



Three certificate programs

- Software Product Line Professional Certificate
- Product Line Technical Probe Team Member Certificate
- Product Line Technical Probe Leader Certificate



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

45



## Software Product Lines: The Future



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

46



## The Outlook

Software Product Lines will continue to grow as a development paradigm.

- Companies will look for opportunities to field software product lines.
- Tool- and environment-builders will eventually appreciate the enormous potential of this market.
- Product line engineering will be a standard part of software engineering curricula.
- Software architects will routinely consider building architectures for a family of systems, using variability mechanisms wisely.
- Product/project managers will always ask whether a project being launched can be a product line.
- Decision-makers will have the tools, models, and experience base necessary to help them make the right decisions and carry them out.



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

47

## Current research topics - 1

### Product lines and aspects

- An *aspect* is a concern that cuts across units of code in a program.
- Aspects represent a way to achieve multi-dimensional separation of concerns.
- Making a “global” change merely requires changing the aspect and then weaving the aspect through the code.
- Example: Changing a program’s overall fault-handling approach

Can aspects be used to represent the variation points of a software product line?



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

48

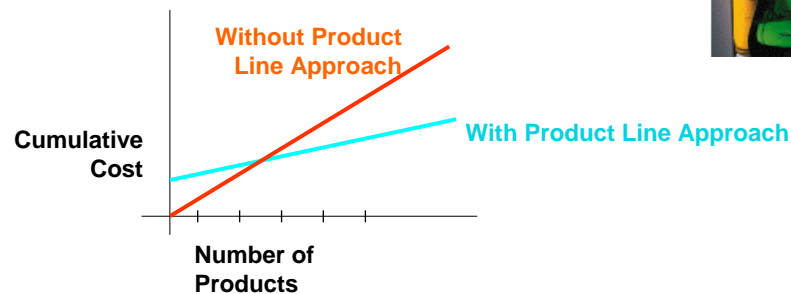




## Current research topics - 2

### Product line economic models

Here is our "classic" economic model:



This comes from a world where economic advantage is anecdotal, intuitive, and based on single data points.



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

49

## Example of a product line economic model

SIMPLE: Structured Intuitive Model for Product Line Economics

- $C_{org}()$  : cost to an organization of adopting the product line approach
- $C_{cab}()$  : development cost to develop a core asset base
- $C_{unique}()$  : cost to develop unique software that itself is not based on a product line platform.
- $C_{reuse}()$  : cost to reuse core assets in a core asset base

Example: Cost of building a product line of n products =

$$C_{org}() + C_{cab}() + \sum_{i=1}^n (C_{unique}(product_i) + C_{reuse}(product_i))$$

Next:

- Adding real options theory to the models.
- Accounting for more realistic product line scenarios (e.g., generations)



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

50



## Current research topics - 3

### Product line practice patterns

Patterns represent pre-packaged solutions to known problems.

Design patterns are well-known in software engineering.

Organizational or process patterns also provide solutions or solution strategies to attack recurring problems

- How do I determine the scope of my product line?
- How do I establish a production capability?
- How do I launch a product line effort?
- How do I adopt the product line approach across my whole organization?



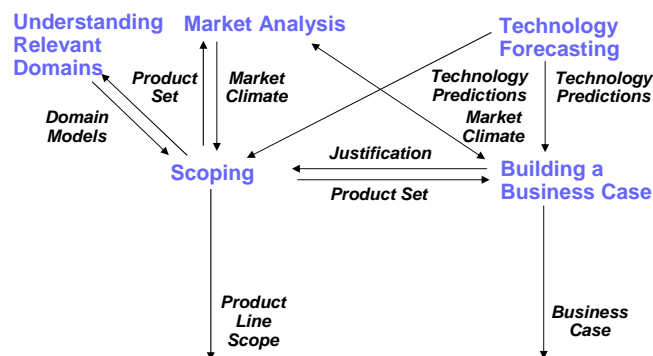
Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

51

## Example: "What to Build" Pattern



Patterns can provide

- Guidance for managers, blueprint for organizations
- Basis for process enactment, workflow and rules engines



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

52



## Current research topics - 4

### Production Planning

How do I best plan to turn out products in my product line?

#### Feature modeling

- Planning the features that are available
- Accounting for feature interaction

#### Choosing a production strategy

- Generator? Compile-time switches? Hand-coding the special bits?
- Depends on people available (where and when) and their skill levels

#### Choosing variability mechanisms in product line architectures

- Choosing based on economic models
- Minimize [ (cost to build) +  $\sum$ (cost to exercise) ]



## Other research topics

### Product line testing

- Driving example: Avionics. How do you keep from flight-testing every product as though it were unique?

### Engineering “non-traditional” product lines

- Product lines of components for other product lines
- Cross-organizational product lines

### Lightweight, low-cost, low-barrier product line approaches



## Software Product Lines and SBCARS

Software product lines epitomize strategic *reuse of components* (and much more) under the auspices of a strong *architecture*.

The strongest product line organizations we have seen view their main business as the care and nurturing of the core asset base, which contains the *architecture* and *components* and is *reused* to turn out products.



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

55

## Contact Information and Resources

### Contact Information

**Dr. Paul C. Clements**  
Product Line Systems Program  
Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213

Email: [clements@sei.cmu.edu](mailto:clements@sei.cmu.edu)

### World Wide Web:

[www.sei.cmu.edu/architecture](http://www.sei.cmu.edu/architecture)  
[www.sei.cmu.edu/productlines](http://www.sei.cmu.edu/productlines)

### Resources

**[www.sei.cmu.edu/productlines](http://www.sei.cmu.edu/productlines)**

- Detailed software product line case studies
- Software product line practice framework
- SEI software product line products and services
- Info about courses and training
- Upcoming events in the software product line community
- Pointers to community resources
- Catalog of published examples



Software Engineering Institute

CarnegieMellon

Software Product Lines  
Paul C. Clements / 31 August 2007  
© 2007 Carnegie Mellon University

56

