Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

CMMI®: A Knowledge Infrastructure

November 12, 2002

Stephen E. Cross, Ph.D.
Director and CEO
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

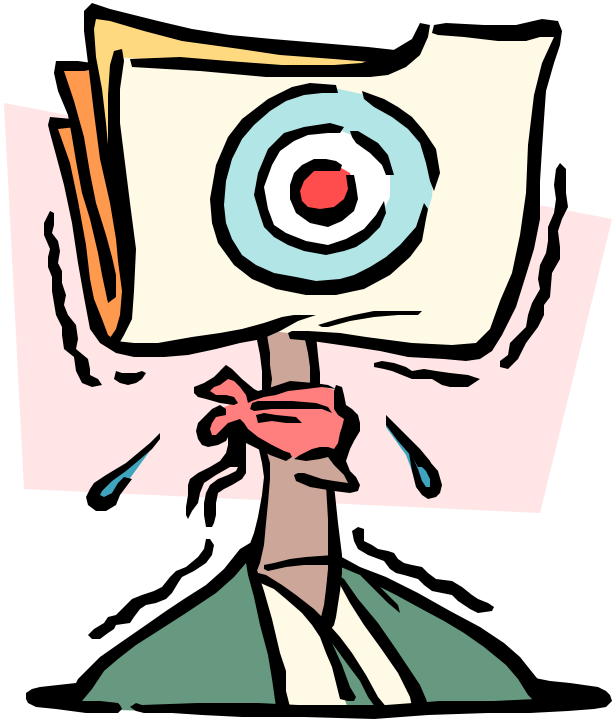
Carnegie Mellon
Software Engineering Institute

This Briefing Refers to the Following Service Marks and Trademarks

® Capability Maturity Model, Capability Maturity Modeling, CMM Integration; CMMI; are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

SM IDEAL; Personal Software Process; PSP; SCAMPI; SCAMPI Lead Assessor; SCAMPI Lead Appraiser; Team Software Process; and TSP are service marks of Carnegie Mellon University.

Carnegie Mellon SW-CMM Announcement, 10/28/02



Carnegie Mellon CMM Facts

A New SW-CMM Will Not Be Released

- The SEI is committed to the CMMI Product Suite.
- The sunset of the SW-CMM will proceed as planned.
- No one can release a new SW-CMM version without SEI permission.

CMMI Supports Software and IT Organizations

- Many software and IT organizations are already using CMMI.
- CMMI for Software (CMMI-SW) was released in August 2002.
- Development of CMMI interpretive guidance has been initiated.

The SEI Is the Steward of CMM Intellectual Property

- We support the transition of CMMs into broad use.
- We meet the needs of the community with CMM-based products.
- We maintain high quality in all CMM-based products.
- We ensure CMM-based products are reliable, valid, and consistent.

FAQs About Sunsetting the SW-CMM

- How does this relate to the CMMI work at the SEI?
- What is a derivative work?
- Will a new version of the SW-CMM be released?
- How does CMMI support software only and information technology (IT) organizations?
- How are Carnegie Mellon, the SEI and ISRI related?
- Can another organization release a new version of the SW-CMM?
- What responsibilities does the SEI have as steward?
- What are the top benefits of make the transition from SW-CMM to CMMI?
- What are the products that the SEI will continue to support? What items will not be supported, and why?

Carnegie Mellon
Software Engineering Institute

CMMI Q&A Sessions in SEI Booth

Tuesday, November 12, 2002	7:30 am - 8:30 am
	10:30 am -12:00 noon
	2:30 pm - 6:00 pm
Wednesday, November 13, 2002	7:30 am - 8:30 am
	10:00 am -12:00 noon
	3:00 pm - 3:15 pm
Thursday, November 14, 2002	7:30 am - 8:30 am
	10:00 am -12:00 noon

CMMI: A Knowledge Infrastructure

Knowledge - the fact or condition of knowing something with familiarity gained through experience or association

Experience - practical knowledge, skill, or practice derived from direct observation of or participation in events or in a particular activity

Engineering - the design and manufacture of complex products

Infrastructure - the underlying foundation or basic framework (as of a system or organization)

Ref: Merriam-Webster, http://www.m-w.com/cgi-bin/dictionary

What I'd Like to Share With You

A vision – what great engineering looks like

Some thoughts on the past, the present, and the future

Three key ideas for the engineering of complex (software intensive) products

A case study

Myths about the CMMI

Carnegie Mellon
Software Engineering Institute

So Steve

What Does Great Engineering Look Like?

Consider This Example from the Building Trades: A “Four-Hour House”

Start at t_0

t_0 + 2hrs 45 min

ã Building Industry Association, San Diego, CA, 1997.

Carnegie Mellon
Software Engineering Institute

Observations

Work is planned

Work is measured

Design before build

Analyze the design

Commit to the design

Create team experience before, *not during*, the build

Reuse knowledge of past designs and builds

Carnegie Mellon
Software Engineering Institute

What I'd Like to Share With You

A vision – what great engineering looks like

Some thoughts on the past, the present, and the future

Three key ideas for the engineering of complex (software intensive) products

A case study

Myths about the CMMI

The Past

era of "manufacturing in quality"

Augustine's 12th law
(it costs a lot of money to build bad products)

Crosby's Quality Management Maturity Grid
(historical footnote – CMM's genesis was manufacturing)

Quality Management Maturity Grid

Management Categories	Stage 1: Uncertainty	Stage 2: Awakening	Stage 3: Enlightenment	Stage 4: Wisdom	Stage 5: Certainty
Cost of quality as % of sales	Reported: unknown Actual: 20%	Reported: 5% Actual: 18%	Reported: 8% Actual: 12%	Reported: 6.5% Actual: 8%	Reported: 2.5% Actual: 2.5%
Summation of company quality posture	"We don't know why we have quality problems."	"Must we always have quality problems?"	"We are identifying and resolving our quality problems."	"We routinely prevent defects from occurring."	"We know why we don't have quality problems."

Crosby, P. *Quality is Free: The Art of Making Quality Certain*. New York: McGraw-Hill, 1979.

The Past

era of "manufacturing in quality"

Augustine's 12th law – Law of Counter Productivity

it costs a lot of money to build bad products

Crosby's Quality Management Maturity Grid

(historical footnote – CMM's genesis was manufacturing)

CMM for Software

focus was quality software processes

The Present

era of "engineering in quality"

CMMI enables a paradigm change
- systems and software engineering have merged
- focus is on
  - process and product quality
  - business results
  - "engineering in quality"

CMMI is being adopted quickly

Companies in key markets are adopting the CMMI
- defense
- aerospace
- automotive
- entertainment
- telecommunications

Implication for software engineers

Quality software

a process, product, and business focus

Software quality

a process focus

The Future

era of "innovating in quality"

Unprecedented engineering challenges

Customers demanding (quality) products faster and cheaper

Management expects higher productivity

Engineering fields continually evolve ... and merge

Organizations are dynamic

Knowledge and experience must be shared

The future is now!

CMMI is our knowledge infrastructure

What We Need To Do

Adopt

Use

Innovate

Share

Carnegie Mellon
Software Engineering Institute

What I'd Like to Share With You

A vision – what great engineering looks like

Some thoughts on the past, the present, and the future

Three key ideas for the engineering of complex (software intensive) products

A case study

Myths about the CMMI

Key Idea #1: Move to the Left

Software and systems engineering have merged.

"Systems engineering consists of two significant disciplines: the technical knowledge domain in which the systems engineer operates, and systems engineering management."

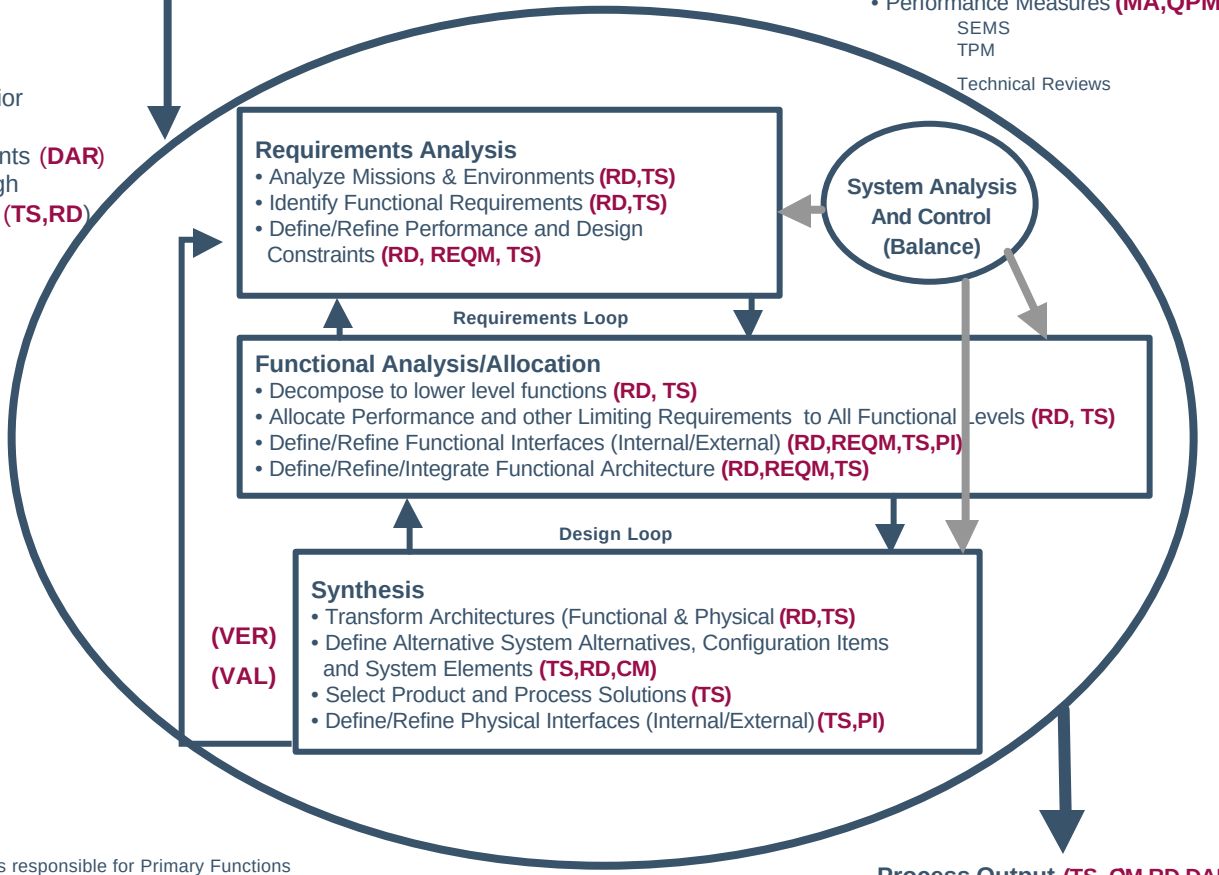
Ref: *Systems Engineering Fundamentals*, Defense Acquisition University Press, Dec 2000. http://www.dsmc.dsm.mil/pubs/gdbks/pdf/SEFGuide12-00.pdf

Process Input
- Customer Needs/Objectives/ Requirements (**RD**)
  - Missions
  - Measures of Effectiveness
  - Environments
  - Constraints
- Technology Base (**TS**)
- Output Requirements from Prior Development Effort **(RM,RD)**
- Program Decision Requirements (**DAR**)
- Requirements Applied Through Specifications and Standards (**TS,RD**)

- Trade-Off Studies **(RD,TS)**
- Effectiveness Analysis **(RD,TS,MA)**
- Risk Management **(RSKM)**
- Configuration Management **(CM)**
- Interface Management **(PI)**
- Data Management **(PP)**
- Performance Measures **(MA,QPM)**

SEMS
TPM

Technical Reviews



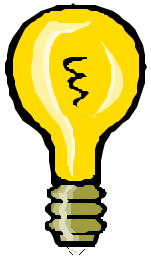
Related Terms:

Customer = Organizations responsible for Primary Functions
Primary Functions = Development, Productions/Construction, Verification Deployment, Operations, Support, Training, Disposal
Systems Elements = Hardware, Software, Personnel, Facilities, Data, Material Services, Techniques

Process Output (TS, QM,RD,DAR)
- Development Level Dependent
  - -Decision Database
  - -System/Configuration Item Architecture
  - -Specifications & Baselines

Ref: *Systems Engineering Fundamentals*, Defense Acquisition University Press, Dec 2000.http://www.dsmc.dsm.mil/pubs/gdbks/pdf/SEFGuide12-00.pdf

Key Idea #2: Never Make the Same Mistake Twice

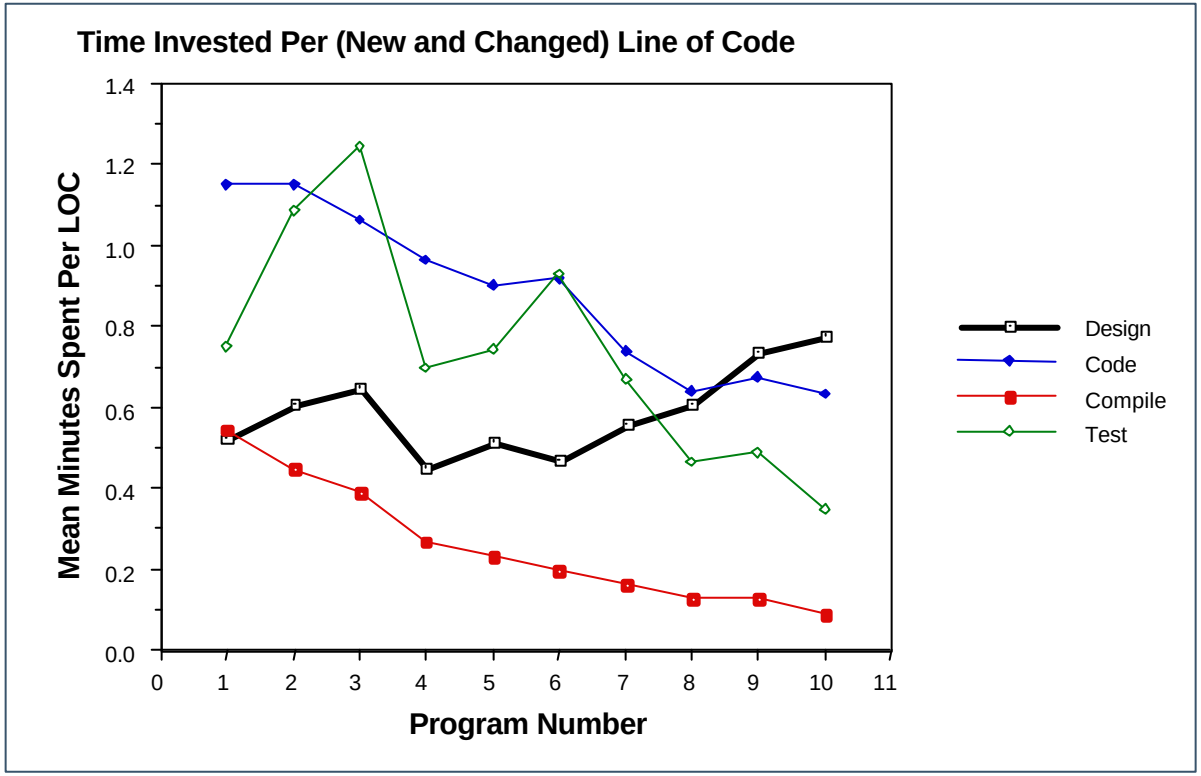


Continuously improve
(e.g., learning from mistakes)

Quality counts
(e.g., security correlation)

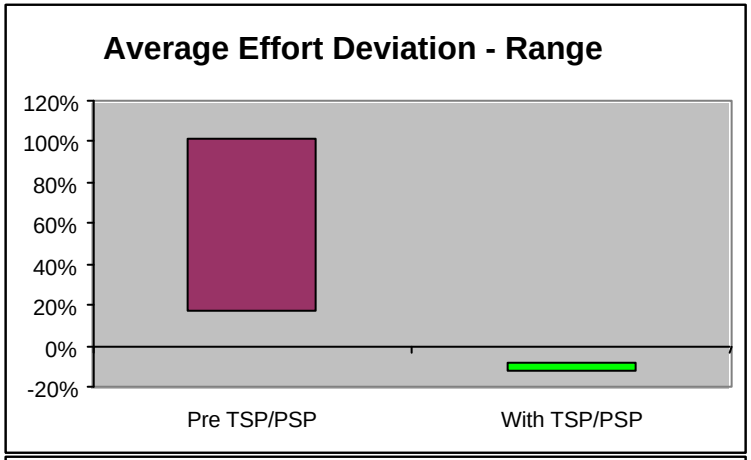
Advocate that your suppliers use disciplined practices and insist that they provide high-quality engineered products

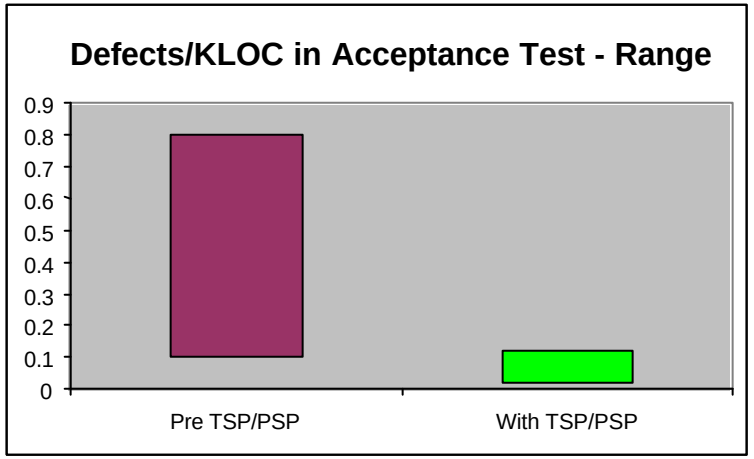
Personal Software Process Results

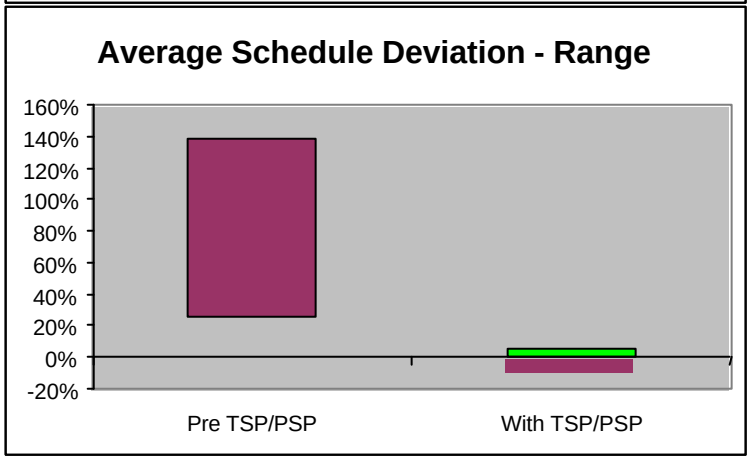


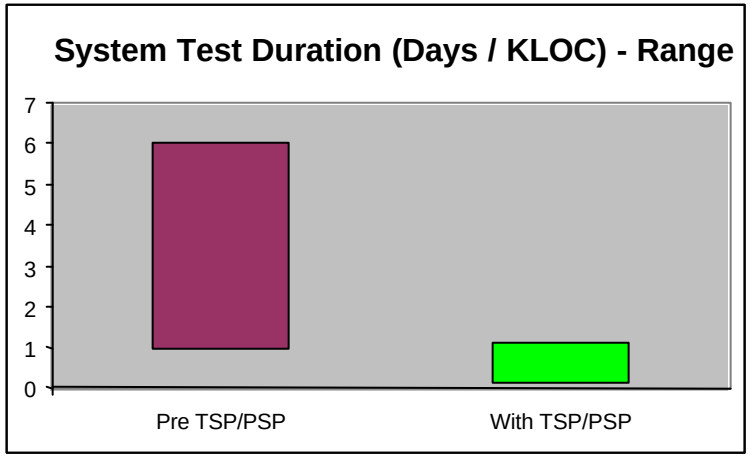
Ref: W. Hayes, J. Over, *Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers* (CMU/SEI-97-TR-001). See:http://www.sei.cmu.edu/publications

Team Software Process Results









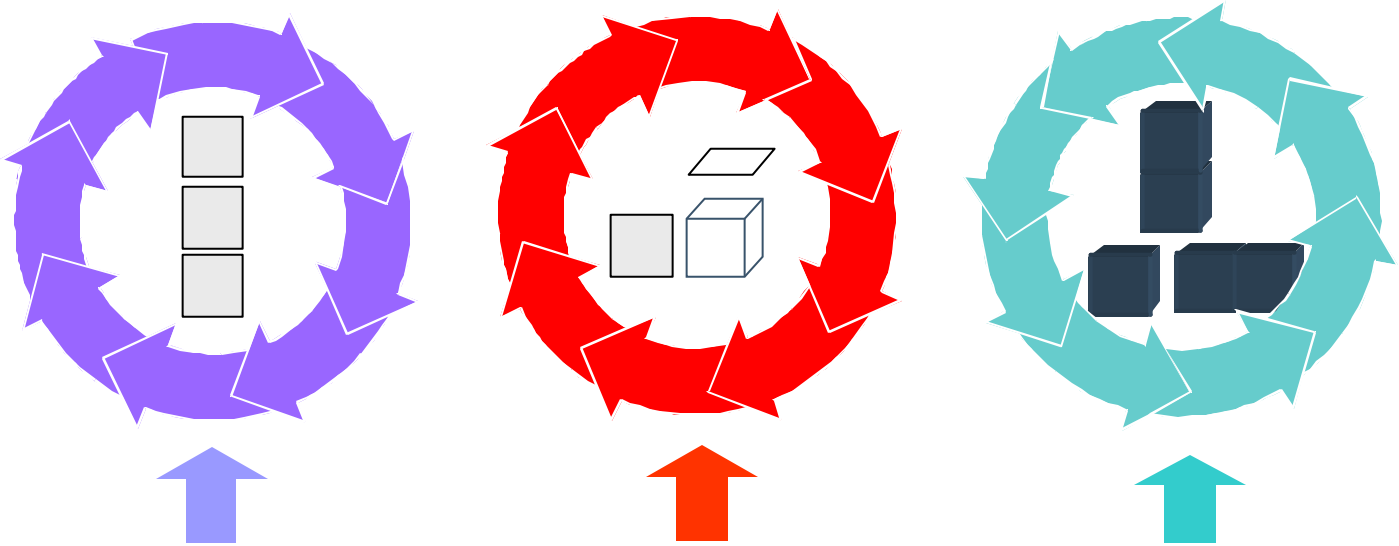
http://www.sei.cmu.edu/publications/documents/00.reports/00tr015.html

Key Idea #3: *Reuse Everything*

Use of a common asset base

in production

of a related set of products



Architecture

Production Plan

Scope Definition
Business Case

SEI Program Snapshot

CMMI is also a *technology transition* infrastructure

SEI purpose – help others make measured improvements in their software engineering practices

SEI focus areas

- Process management practices
- Acquisition practices
- Team training
- Architecture analysis
- COTS evaluation and integration
- Design practices (security, real-time)
- Measurement practices

Carnegie Mellon
Software Engineering Institute

What I'd Like to Share With You

A vision – what great engineering looks like

Some thoughts on the past, the present, and the future

Three key ideas for the engineering of complex (software intensive) products

A case study

Myths about the CMMI

Carnegie Mellon
Software Engineering Institute

Case Study: Cummins, Inc.

Moved to left

Never made same mistake twice

Reused everything

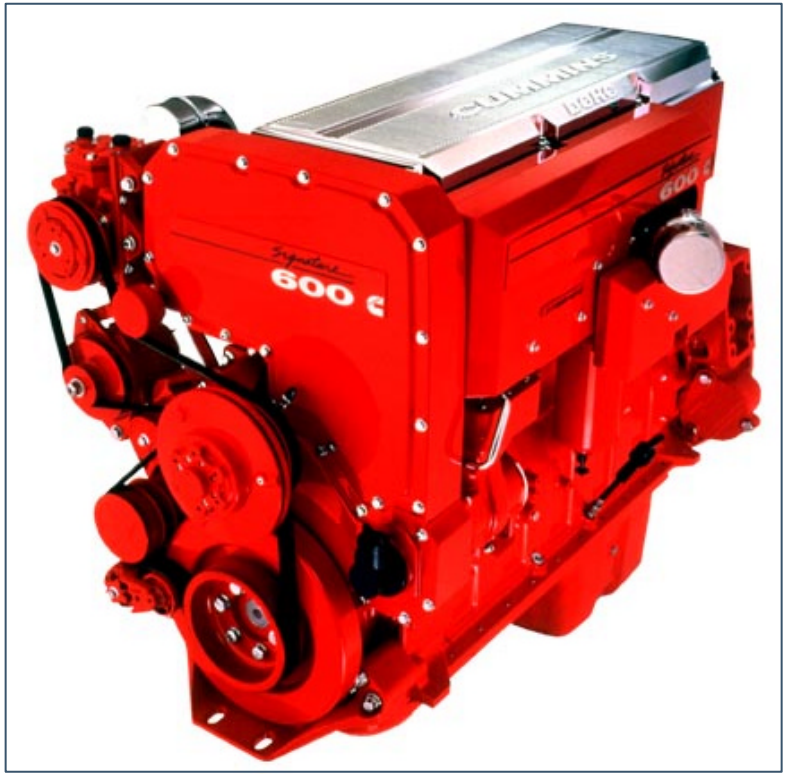
Cummins, Inc.

World's largest manufacturer of large diesel engines (over 200 hp)

25,000 employees

350 controls and electronics engineers

$7B annual sales



In 1993, Cummins Had a Problem

The market was demanding new products.

- six engine projects were underway
- another 12 were planned

Each project had complete control over its development process, architecture, even choice of language. Two were trying to use object-oriented methods.

Ron Temple (VP in charge) realized that he would need another 40 engineers to handle the new projects -- out of the question.

Temple realized this was no way to do business. In May 1994 he halted all the projects.

Key Insights

Applied key product line practices

- Organization management (e.g., business case)
- Engineering practices (e.g., configuration management)
- Software engineering practices (e.g., architecture definition and evaluation, components, mining legacy assets)

Investment strategy to create and maintain core assets

A disciplined, process-based culture already in place

Top management commitment

Cummins' Results

In early 1995, the product was launched on time (relative to re-vamped schedule) with high quality. Others followed -- on time and with high quality.

Achieved a product family capability with a breathtaking capacity for variation, or customization

- 9 basic engine types
- 4 - 18 cylinders
- 3.9 - 164 liter displacement
- 12 kinds of electronic control modules
- 5 kinds of microprocessors
- 10 kinds of fuel systems
- diesel fuel or natural gas

Carnegie Mellon
Software Engineering Institute

Quantitative Results 1

20 product groups launched, which account for over 1,000 separate engine applications

75% of all software, on average, comes from core assets

Product cycle time drastically decreased (e.g., from 250 person-months to a few person-months).

Projects are more successful (e.g., before: 3 of 10 were on track, 4 were failing, and 3 were on the edge;
now: 15 of 15 are on track)

Customer satisfaction is high (e.g., productivity gains enable new features to be developed and quickly introduced)

Quantitative Results 2

Supported Components	1992	1993	1994	1995	1996	1997	1998
Electronic control modules (ECMs)	3	3	4	5	5	11	12
Fuel Systems	2	2	3	5	5	10	11
Engines	3	3	5	5	12	16	17
Features * ECM	60	80	180	370	1100	2200	2400

Achieving this flexibility without the product line approach would have required 3.6 times the current staff.

Carnegie Mellon
Software Engineering Institute

What I'd Like to Share With You

A vision – what great engineering looks like

Some thoughts on the past, the present, and the future

Three key ideas for the engineering of complex (software intensive) products

A case study

Myths about the CMMI

Myths About the CMMI

It's too big, takes too long, costs too much ….

Ratings are meaningless, hence CMMI is useless

It is inconsistent with agility, evolutionary development, cycle time reduction, <your favorite new buzzword> …

It is not useful for software developers

Summary

Systems and software engineering have merged

Three key ideas
- Move to the left
- Reuse everything
- Never make the same mistake twice

Trends in engineering mean we need a common framework upon which to create, share, and use engineering knowledge

CMMI provides this knowledge infrastructure