Carnegie Mellon
Software Engineering Institute

Sponsored by the U.S. Department of Defense © 2005 by Carnegie Mellon University

A Superior Process: Requirements

Watts S. Humphrey

The Software Engineering Institute

Carnegie Mellon University

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
FEB 2005		**00-00-2005 to 00-00-2005**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Superior Process: Requirements	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,Software Engineering Institute,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**30**	

Carnegie Mellon
Software Engineering Institute

Agenda

Development Performance

Future Challenges

Current Problems

Scalability

Process Principles

Process Requirements

© 2006 by Carnegie Mellon University

Performance Examples

The IRS system – finally started to use in 2005

- 5 years of delays
- costs exploded to $2 B

FBI system killed

- 3 years of work
- $150 M
- 5 CIOs, 9 program managers

Britain's child-support project

- a year late
- $844 M
- didn't pay 50% of cases

© 2006 by Carnegie Mellon University

Transportation

Automobiles

- Mercedes Benz – batteries, windows, temperature
- Bendix Brakes
- Ford Explorer

Boeing 777

- Malaysian Airlines
- Singapore

© 2006 by Carnegie Mellon University

This Is No Joke!

These are not new problems.

- CONFIRM system 20 years ago
- Cancelled after 3 ½ years and $125 M

How long will society tolerate such performance?

Do we want government controls?

- Methods standards and approval?
- Pre-shipment reviews?
- Legislated warranties?

We had better solve our own problems or others will solve them for us.

© 2006 by Carnegie Mellon University

Sir Francis Bacon

"He who will not apply new remedies must expect new evils."

© 2006 by Carnegie Mellon University

Challenges of the Future

Our priorities must change. Systems are now

- larger
- distributed
- integrated
- pervasive
- critical

The methods of the past are not suitable today.

In the future, they could be dangerous!

© 2006 by Carnegie Mellon University

Systems Development Phases

Phase I – Feasibility
- after World War II
- mid 1960s

Phase II – Manageability
- early to mid 1960s
- feasibility had much lower priority
- still continuing

Phase III – Quality
- now
- manageability will be lower priority
- priorities: safety, security, privacy, usability, etc.

© 2006 by Carnegie Mellon University

Carnegie Mellon
Software Engineering Institute

Process Criteria

A superior process must meet three criteria.

- work effectively for the smallest and largest programs
- consistently produce superior results
- be recognized as producing superior results

The current generally-used processes

- do not consistently work for small projects
- rarely work effectively for large projects
- produce inconsistent and often poor-quality results

© 2006 by Carnegie Mellon University

Superior Results

A superior process must consistently deliver products on schedule and for their committed costs.

It must routinely deliver high-quality products.

- functions
- properties
- defects

It must respond to changing needs.

© 2006 by Carnegie Mellon University

Systems Problems – 1

Systems have emergent properties.

- Emergent properties are produced by synergies among the system's components.
- No single component supplies any emergent property.

Example emergent properties

- performance
- reliability
- safety
- security
- usability
- maintainability
- installability

© 2006 by Carnegie Mellon University

Systems Problems – 2

The quality of an emergent property is determined by the

- worst-quality component
- lowest-performing component
- least-reliable component
- least usable component
- least secure component

Therefore, a superior process must

- identify all poor-quality components
- fix all poor-quality components
- improve all poor-quality components
- prevent all poor-quality work

© 2006 by Carnegie Mellon University

The Scale-Up Problem

As systems become larger

- they become more complex
- their development is more challenging
- their management is more difficult
- new problems emerge at the systems level
- component problems are magnified at higher levels

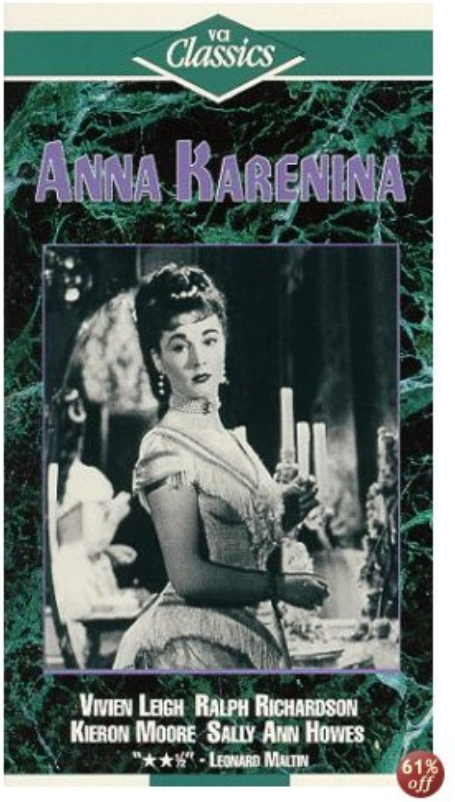
Therefore, a superior process must

- prevent all poor-quality work
- guide the work at all levels

© 2006 by Carnegie Mellon University

Tolstoy: Anna Karenina

"Happy families are all alike; every unhappy family is unhappy in its own way."



© 2006 by Carnegie Mellon University

The Common Strategy

The most common strategy:

- If it isn't broken – don't fix it.

This strategy has four problems.

- Processes break in an infinite number of ways.
- Fixing one problem will not fix the next.
- These random fixes do not fix causes.
- The process is not consistently or measurably improved.

© 2006 by Carnegie Mellon University

The CMMI Strategy

The only effective answer is a strategy that is

- cohesive
- comprehensive
- based on sound principles

This is the logic for CMMI.

It has gaps.

- very large systems
- very small projects and organizations

© 2006 by Carnegie Mellon University

Scalability

Scalability is fundamental.

Developers now typically use the same practices for

- a 1,000 LOC application
- a 1,000 LOC module for a 1,000,000 system

Examples from other fields.

- boat building
- building construction
- accounting

© 2006 by Carnegie Mellon University

Scalability Requirements

For a process to be scalable, every key aspect must scale up and down.

- practices
- measures
- quality management

Therefore, the work at every level must

- be based on precise and detailed plans
- produce high-quality products
- be based on measured, statistically usable, and auditable data, not
  - after-the-fact reports
  - third-party estimates
  - guesses

© 2006 by Carnegie Mellon University

Meeting Schedules

Fred Brooks: “Schedules slip a day at a time.”

© 2006 by Carnegie Mellon University

A Process Principle

To consistently maintain costs and schedules, the one-day slips must be recovered the next day. If not

- the delays will compound
- the commitments will become unmanageable

With current common practices, schedule slips cannot be detected until projects are weeks or months late.

Then it is generally too late to recover.

© 2006 by Carnegie Mellon University

System Schedules

For large-scale integrated-systems-development programs

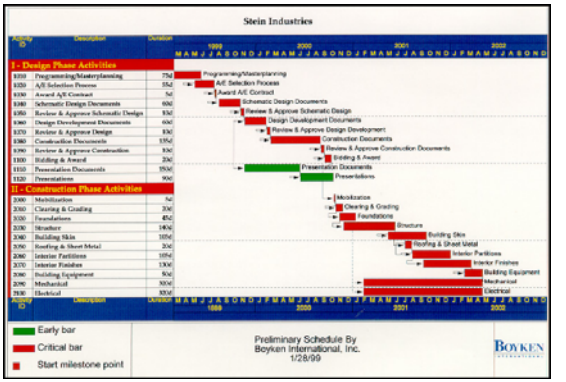
- Any subsystem delay will delay the system.
- Any component delay will delay the subsystem and system.
- Any project delay will delay the component, subsystem, and system.
- Any team delay will delay the project, component, subsystem, and system.
- Any developer delay will delay the team, project, component, subsystem, and system.

© 2006 by Carnegie Mellon University

Program Management

To manage large programs, every system level must be managed.

- subsystems
- components
- projects
- teams
- developers



Therefore, to consistently produce quality products on schedule and for planned costs, every developer must

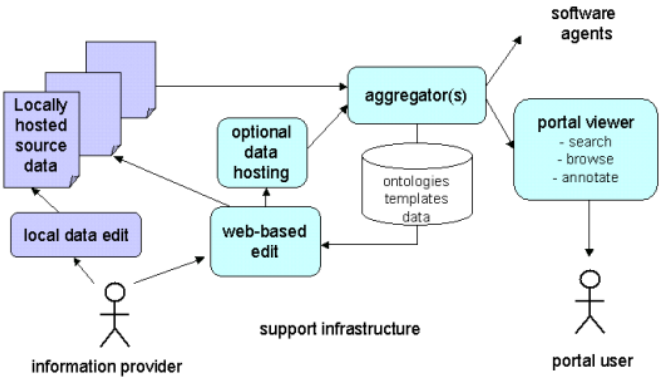
- consistently produce quality products
- predictably meet schedules

© 2006 by Carnegie Mellon University

Process Requirements

For a process to meet these needs, it must

- properly manage the knowledge work
- manage costs and schedules
- manage quality



© 2006 by Carnegie Mellon University

Knowledge Work - 1

The first rule for managing knowledge work is to recognize that you can't manage it.

The knowledge workers must manage their own work.

The second rule for managing knowledge work is that the teams and developers must

- know how to manage themselves
- negotiate their commitments with management
- manage with data
- own their own work

© 2006 by Carnegie Mellon University

Knowledge Work - 2

To manage themselves, developers must

- use personal processes
- follow personal plans
- measure, track, and report status
- measure and manage quality

© 2006 by Carnegie Mellon University

Knowledge Work - 3

To follow a defined, planned, measured, and quality-controlled process, developers must

- work on self-directed teams
- define their own processes and plans
- negotiate their own commitments
- be competently led and coached

© 2006 by Carnegie Mellon University

Cost and Schedule

There is no secret to meeting cost and schedule commitments.

In every field, only four things are required.

- The developers estimate and plan their own work.
- Everyone precisely and regularly tracks and reports status and progress.
- Schedule delays are addressed every day.
- When requirements change, everyone
  - revises their plans
  - renegotiates their commitments

© 2006 by Carnegie Mellon University

Quality Management - 1

To successfully manage quality, the development process must be based on four facts.

- The longer a defect remains in a product, the more it costs to find and fix it.
- No test can find more than a fraction of the defects in a product.
- The larger and more complex the product, the smaller this faction will be.
- To get a high-quality product out of test, you must put a high-quality product into test.

© 2006 by Carnegie Mellon University

Quality Management - 2

A quality process must strive to find and fix all defects before test entry.

To do this, the process must

- have multiple early defect-removal steps
- measure product quality at every step
- measure process quality at every step
- make and follow personal and team quality plans
- regularly track product and process quality
- use testing to
  - verify product quality
  - gather quality data
- promptly correct quality deviations from plan

© 2006 by Carnegie Mellon University

Conclusions

These principles have been proven with CMMI and TSP and are being piloted for systems engineering with TSPI.

The critical challenge is to get people to work this way.

Consistently superior knowledge work must include the following elements – at all levels.

- well-defined and consistently practiced principles of personal performance
- teambuilding and empowerment
  - ownership and commitment
  - leadership and coaching
- organizational support and institutionalization
- quantitative performance measurement and appraisal

© 2006 by Carnegie Mellon University