PROCESS MODELS AND METRICS

Models and metrics capture information about the processes being performed

We can model and measure the definition of the process, process performers conformance to the process definition, process performers understanding of the domain to which the process is being applied

Process models and metrics can be used to

- evaluate the process
- gain insight into the product
- find weakness in the environment in which the process is being applied
- provide insight into process improvement
- help tailor and evolve the processes over time

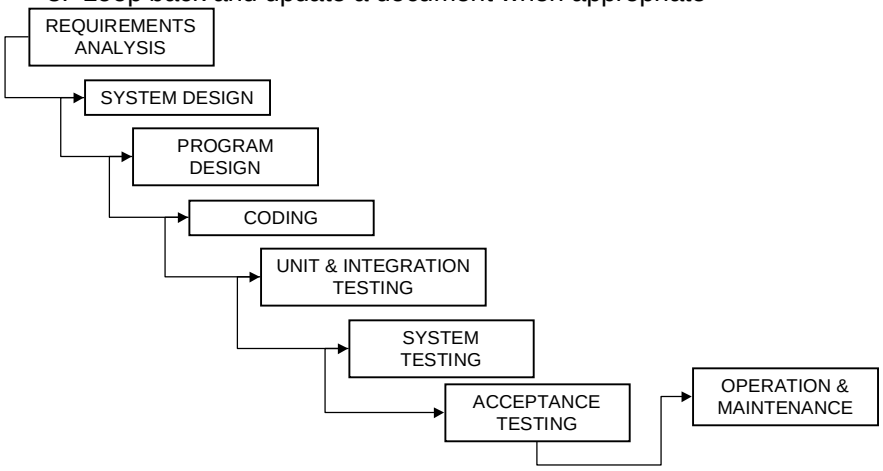
PROCESS MODELS AND METRICS

Terminology

- Technique: A series of steps (requiring skill) for producing a desired effect, i.e., constructing or assessing software, e.g., testing, reading
- Method: An organized approach for applying techniques, e.g., design inspections, test plan. Should include entry and exit criteria (when, how, how long to apply) and management supports (evaluation criteria)
- Life Cycle Model: an integrated set of methods that cover the entire life cycle, e.g., an incremental development model, using structured design, design inspections, etc.
- Engineering: The application and tailoring of techniques, methods, and life cycle models to the problem, project, and organization

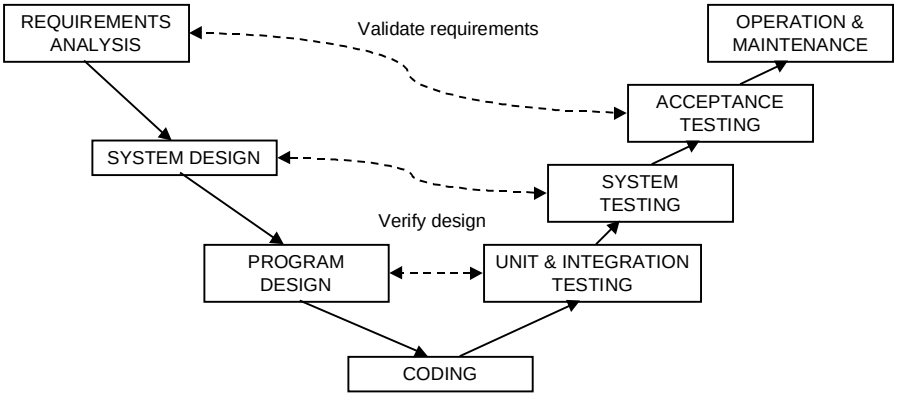
Waterfall Model

1. Start with the requirements and completely determine them
2. Pass through each of the phases sequentially
3. Loop back and update a document when appropriate



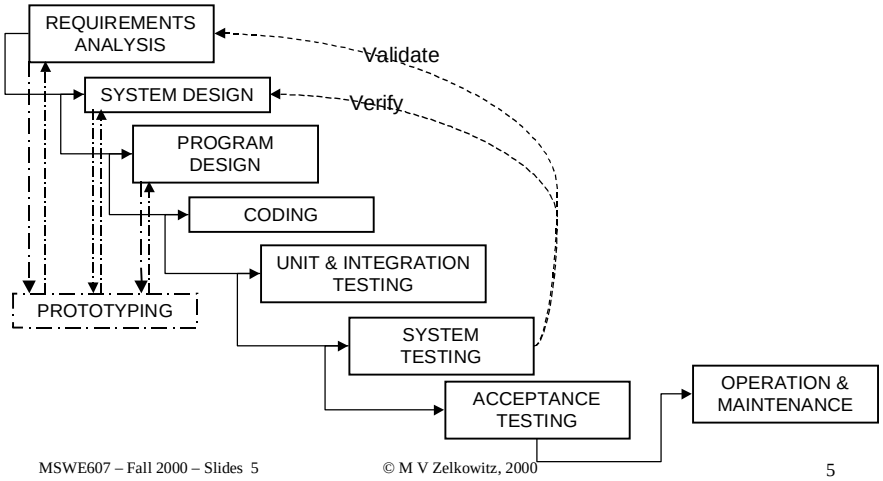
V-Model Modification of Waterfall

Same as the Waterfall model except for the emphasis on the pairing of test stages with development documents



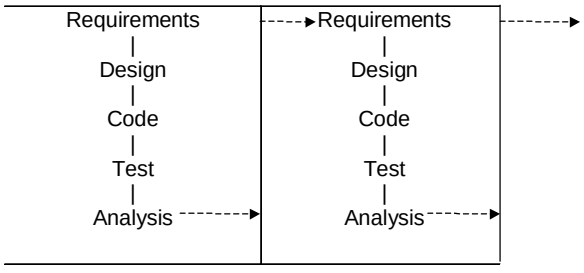
Prototyping Model

1. Find the part of the system in need of further study
2. Build a prototype directly or build a simulation
   - Prototype is typically built in a very high level language
   - Prototype is typically thrown away



Iterative Enhancement Model

1. Start with old requirements, design, code, test, and analysis documents
2. Redevelop, based upon the analysis, starting with the appropriate system document, propagating changes through the full set of documents
3. At each step of the evolutionary process, continue to redesign, based upon analysis



The Spiral Model

It is an iterative enhancement style, risk driven model that allows the developer to choose the life cycle model based upon the risks involved at each stage (version) of development.

Dimensions:

radial - cumulative cost incurred

angular - progress made in completing each cycle of the spiral

Each cycle has the following phases:

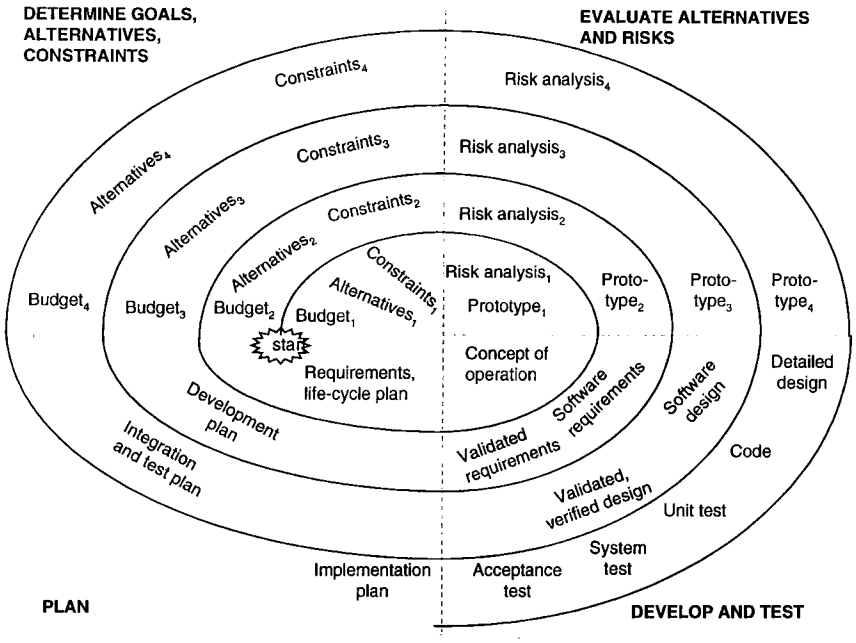
Identify the objectives for that cycle and the alternatives that are possible for achieving those objectives

Evaluate the different alternatives based upon objectives and constraints (identify uncertainties and risks)

Develop strategies that resolve the uncertainties and risks (using benchmarking, prototyping, simulation) and develop the software

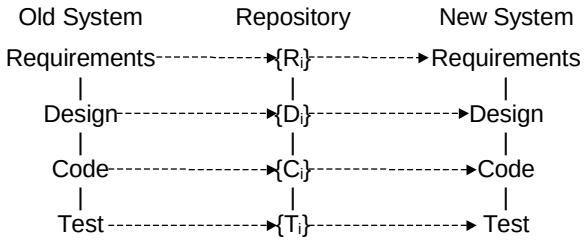
Plan the next stage, allowing any of the possible life cycle models to be used

To initialize the spiral, the feasibility of the basic project objectives are analyzed and any environmental needs are addressed



Reuse Development Model

1. State the requirements for the new system, reusing as much of the old system as feasible
2. Build a new system, using components from the old system or other systems available in the repository, developing new components where appropriate



Choosing the Life Cycle Model

Waterfall model

- Sequential
- Good when problem and solution well understood

Iterative enhancement model

- Incremental versions developed
- Good when problem or solution not well understood, schedule for full function a risk, requirements changing

Prototyping model

- Experimental version
- Good when user unsure of needs, some aspect of the system unclear, experimental analysis needed

Spiral model

- Risk oriented iterative enhancement
- Good when high risk, customer unclear of evolving requirements

Producer/Consumer Model

For each document or intermediate product

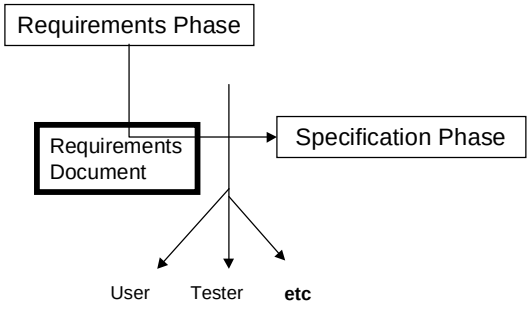
- who are the consumers?
- What are there operational profiles?
- What are their goals or needs for that document or product?

Each document should be evaluated from the perspective of all customers

Example: Requirements Document

Consumers: User, Tester, Developer, Maintainer, SQA, Hardware, …

Producer/Consumer Model



Each phase can be gated to allow for an evaluation from different perspectives

Producer/Consumer Model Perspectives

- Customer
- System Analyst
- Designer
- Coder
- System Tester
- Maintainer
- Quality

Producer/Consumer Model Perspectives

Perspective	Requirements	Specification	Design	Code	Test
Customer	X	X			
System Analyst	X		X		
Designer		X	X		X
Coder			X		
System Tester	X			X	
Maintainer		X	X	X	
Quality					X

Defining Process Goals

What are the goals of the Software Development Process?

From the project management perspective:

- Develop a set of documents that all represent the same system

From the user's perspective:

- Develop a system that satisfies the user's needs with respect to functionality, quality, cost, etc.

From the corporate perspective:

- Improve the organization's ability to develop quality systems productively and profitably

Defining Process Goals

Given process goals, the selection of the activities depend upon the environmental characteristics

What are the goals of the requirements activity?

- To characterize the user's needs in order to specify a system that satisfies them

The selection of activities depends upon the environment, e.g., contract vs. general product

What does this say about user input?

- How do we create customer/user scenarios and models?
- How do we get the user involved in the requirements definition (e.g., prototype screens), and test plans?

Defining Process Goals

Goals help define the life cycle model, methods, and techniques, e.g., What are the goals of the test activity?

- To assess quality or to find failures?

Selection of activities depends upon the answer

Assess quality:

Tests based upon user operational scenario

- Statistically based testing technique
- Reliability modeling for assessment

Find failures:

- Test a function at a time
- General to specific
- Reliability models not appropriate

Defining Process Guidebook

- Need to provide flexible process definition appropriate information for process selection support process integration and configuration, via tailorable definitions and characterizations for life cycle models, methods and techniques

Examples:

If problem and solution well understood choose waterfall process model

- If high number of faults of omission expected emphasize traceability reading approach embedded in design inspections
- When embedding traceability reading in design inspections, make sure traceability matrix exists

Process Needs

There is a reusable software technology, but it is not simple.

We need flexible definitions that can be easily tailored to the problem goals for processes

We need experimentation and analysis of the various technologies to understand

- their strengths and weaknesses
- when and where they are appropriate
- how to tailor them to a specific project

We need education and training in the

- application
- life cycle models
- methods
- techniques
- tools

e.g. what training is done in reading?

Building Measurable Process Models

Example: Modeling the education and training process

Assume the organization has a process in place for training with respect to a method or technique

In our example, let us assume a simple process consisting in a set of steps:

1. provide the individual with training manuals they must read
2. provide a course, educating the individual in the process
3. provide training by applying the process to a toy problem
4. assign the individual to a project that is using the process, mentored by an experienced method user
5. after this the individual is considered fully trained in the process

Building Measurable Process Models

If we need to capture the experience of an individual with respect to a particular method or technique, we can convert this process to an operational model by associating ordinal values with the process steps, representing various stages of experience maturity, by letting each step represents a further passage along the ordinal scale

Thus a value of

0 implies no training,

1 implies the individual has read the manuals,

2 implies the individual has been through a training course,

3 implies the individual has had experience in a laboratory environment,

4 implies the process had been used on a project before, under tutelage, and

5 implies the process has been used on several projects

Building Measurable Process Models

If the education and training process model is valid, the ordinal scale ratings are valid

We can us this ordinal scale rating on a questionnaire

Characterize your experience with method X (subjective rating per person)

0 - none

1 - have read the manuals

2 - have had a training course

3 - have had experience in a laboratory environment

4 - have used on a project before

5 - have used on several projects before

Building Measurable Process Models

Now let us assume we want to characterize the experience of a team with respect to method X.

How do we combine individual data to create a team model?

We can build an interpretive model of the form:

0 implies no member of the team has more than a 1

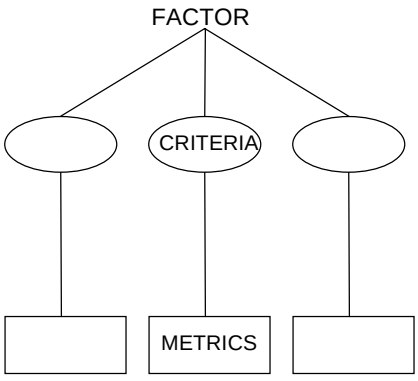
1 implies the team lead has a 4 and each team member has at least a 2

2 implies the team lead has a 5 and each team member has at least a 3

3 implies the team lead has a 5 and each team member has at least a 3
and half the team has at least a 4

This model can be tested over time and can be improved with experience

Factor/Criteria/Metric Paradigm



Management Oriented View of Product Quality

(CORRECTNESS)

Software Oriented Attributes which provide quality

(COMPLETENESS)

Quantitative Measures of those attributes

(UNAMBIGUOUS REFERENCES)

- product characteristics
- customer point of view

Software Quality Metrics

Substantiation of the Model: e.g., User Subjective Evaluation

- How responsive is the system to user request for functionality?
- How responsive is the system to user request for performance?
- How does the user rate the system with respect to:
  - ease of use, functionality, performance, ease of understanding the documentation?
- Why did you buy it?
  - Did you get more, less, same as expected?
  - How many others have you used?
  - Would you recommend it to a friend? A competitor?
  - Would you reorder?
  - Which competitors did you consider?
- How many requests are there for functional enhancements?

Software Quality Metrics

Performance: How well does it Function?

User Concern	Quality Factor
How well does it utilize a resource?	Efficiency
How secure is it?	Integrity
What confidence can be placed in what it does?	Reliability
How well will it perform under adverse conditions?	Survivability
How easy is it to use?	Usability

Software Quality Metrics

Design: How valid is the Design?

User Concern	Quality Factor
How well does it conform to the requirements?	Correctness
How easy is it to repair?	Maintainability
How easy is it to verify its performance?	Verifiability

Software Quality Metrics

Adaptation: How adaptable is it?

User Concern	Quality Factor
How easy is it to expand or upgrade its capability or performance?	Expandability
How easy is it to change?	Flexibility
How easy is it to interface with another system?	Interoperability
How easy is it to transport?	Portability
How easy is it to convert for use in another application?	Reusability

Software Quality Metrics

Evaluating Factors

Based upon project needs, various quality factors are chosen for monitoring, analysis, and feedback for corrective action.

Each Factor is defined by a rating formula whose data is not available till project completion

Each Factor is also defined by a set of criteria that are collected during project development and are used as predictors of the quality factors

These metrics are collected on worksheets

Software Quality Metrics

Quality Factor Formulas

Efficiency	1 -	$\frac{\text{Actual Utilization}}{\text{Allocated Utilization}}$
Integrity	1 -	$\frac{\text{Errors}}{\text{Lines of Code}}$
Reliability	1 -	$\frac{\text{Errors}}{\text{Lines of Code}}$
Survivability	1 -	$\frac{\text{Errors}}{\text{Lines of Code}}$
Usability	1 -	$\frac{\text{Labor-Days to Use}}{\text{Labor-Years to develop}}$
Correctness	1 -	$\frac{\text{Errors}}{\text{Lines of Code}}$
Maintainability	1 -	0.1(Average Labor-Days to Fix)

Software Quality Metrics

Quality Factor Formulas

Verifiability	1 -	Effort to Verify / Effort to Develop
Expandability	1 -	Effort to Expand / Effort to Develop
Flexibility	1 -	0.05 (Average Labor-Days to Change)
Interoperability	1 -	Effort to Couple / Effort to Develop
Portability	1 -	Effort to Transport / Effort to Develop
Reusability	1 -	Effort to Convert / Effort to Develop

Software Quality Metrics

Quality Criteria: Each factor has associated with it a set of quality criteria:

Factor	Criteria
Efficiency	Effectiveness-Communication
	Effectiveness-Processing
	Effectiveness-Storage
Integrity	System Accessibility
Reliability	Accuracy
	Anomaly Management
	Simplicity
Survivability	Anomaly Management
	Autonomy
	Distributedness
	Modularity
	Reconfigurability

Software Quality Metrics

Quality Criteria:

Factor	Criteria
Usability	Operability
	Training
Correctness	Completeness
	Consistency
	Traceability
Maintainability	Consistency
	Modularity
	Self-Descriptiveness
	Simplicity
	Visibility
Verifiability	Modularity
	Self-Descriptiveness
	Simplicity
	Visibility

Software Quality Metrics

Quality Criteria

Factor	Criteria
Expandability	Augmentability
	Generality
	Modularity
	Self-Descriptiveness
	Simplicity
	Virtuality
Flexibility	Generality
	Modularity
	Self-Descriptiveness
	Simplicity
Interoperability	Commonality
	Functional Overlap
	Independence
	Modularity
	System Compatibility

Software Quality Metrics

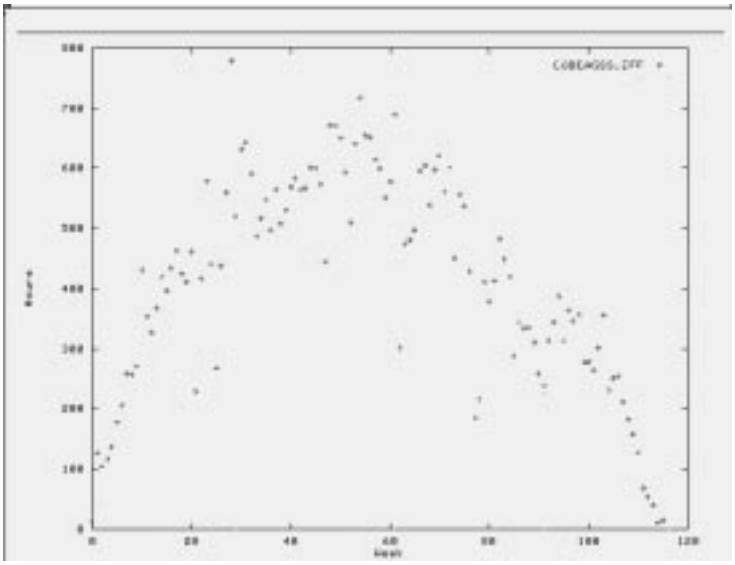
Quality Criteria

Factor	**Criteria**
Portability	Independence
	Modularity
	Self-Descriptiveness
Reusability	Application Independence
	Document Accessibility
	Functional Scope
	Generality
	Independence
	Modularity
	Self-Descriptiveness
	Simplicity
	System Clarity

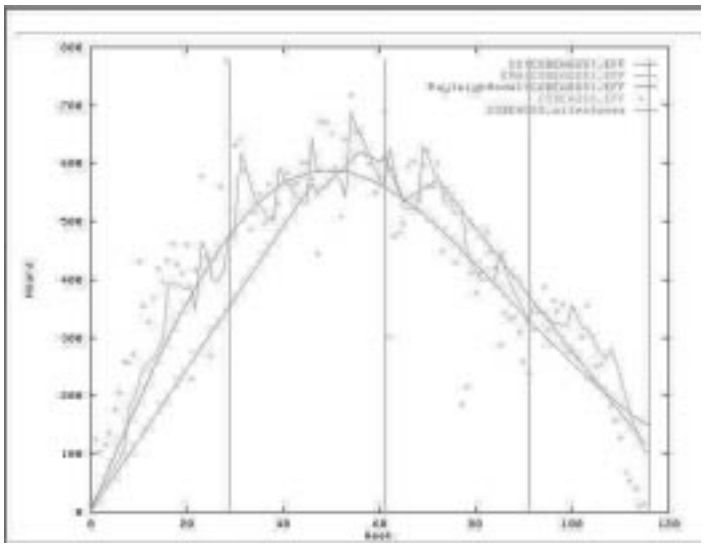
WebME:
Sample Project analyses

Sample tool to display measurement data

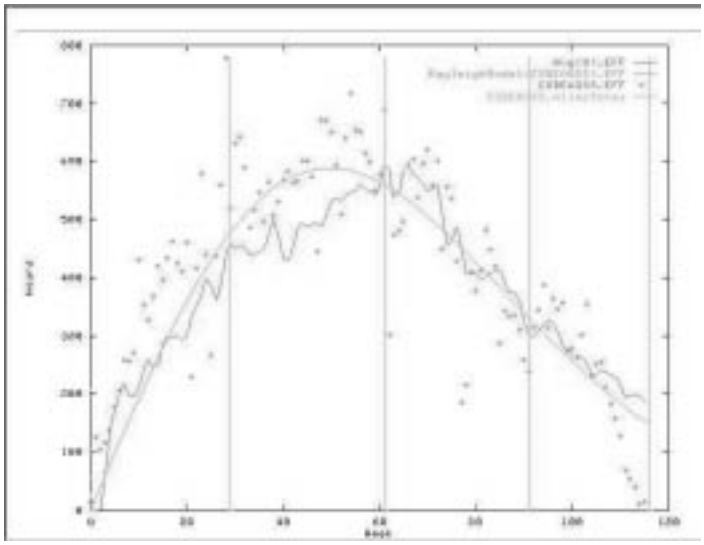
Raw Project Data



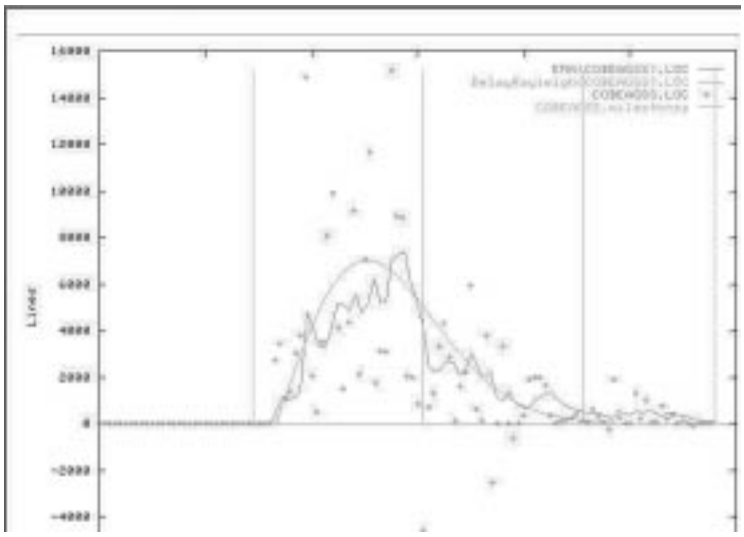
Models: Rayleigh, Moving average, Characteristic Curve



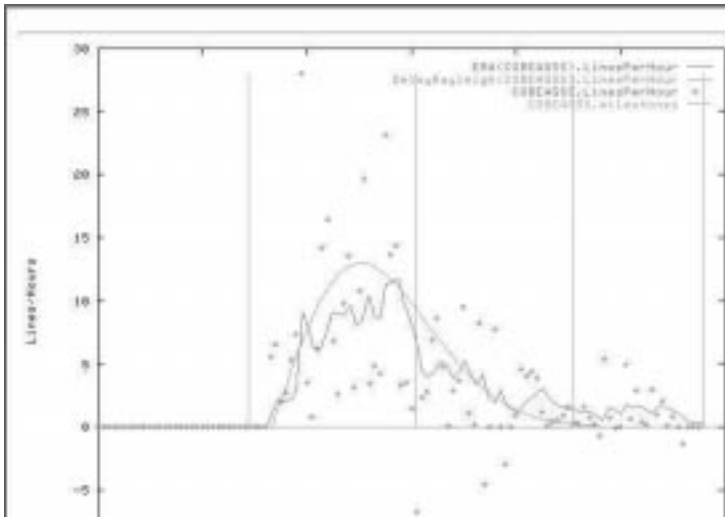
Baseline data



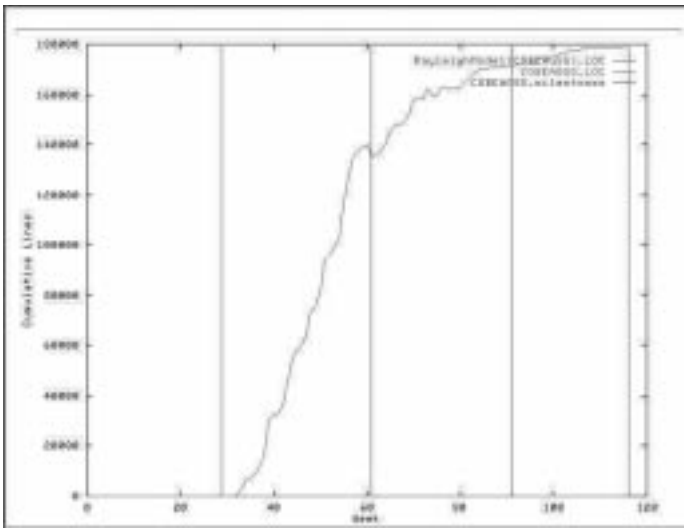
Rayleigh lines of code estimator



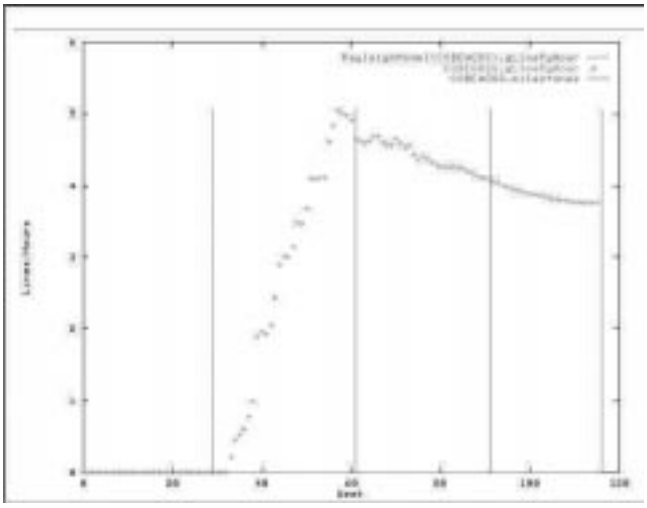
Lines of code per hour



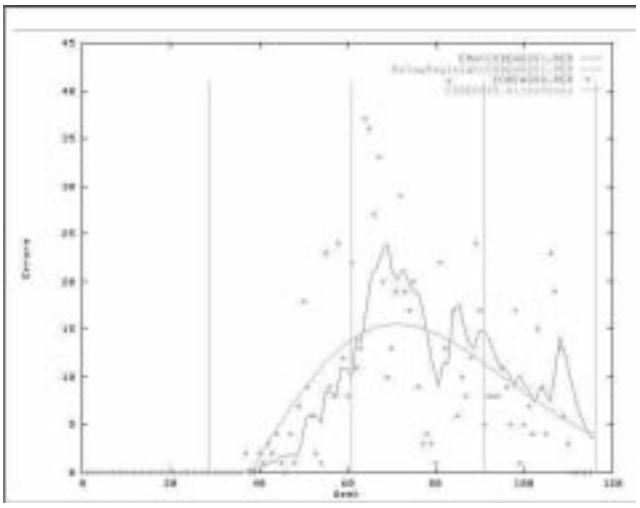
Cumulative Lines developed



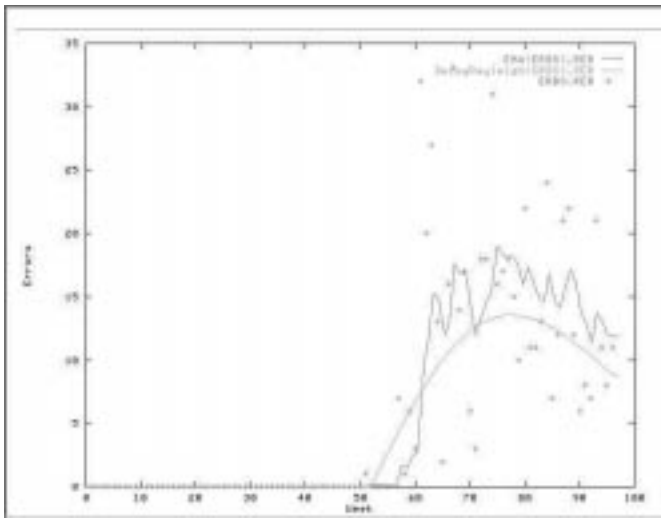
Productivity by week - Project A



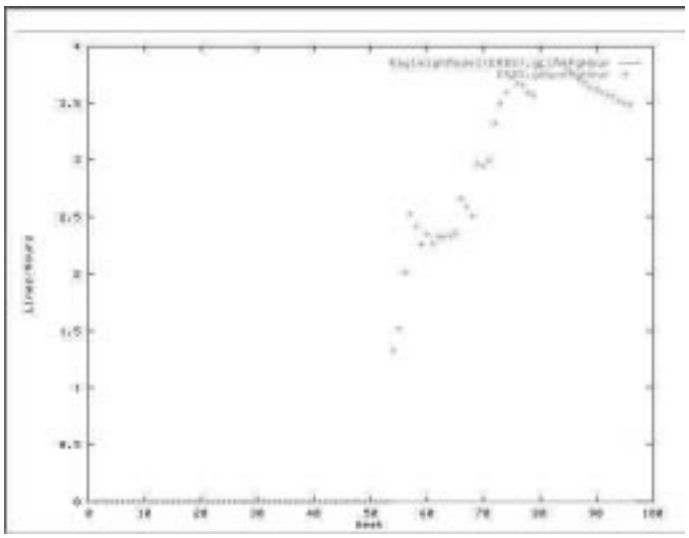
Defects per week - Project A



Defects per week - Project B



Productivity by week - Project B



Summary

- Tools like SME and WebME provide for insightful feedback on project development
- Few tools available for such data manipulation
- Most companies do not collect relevant data

- Costs of collecting this data only a few percent