Advanced Topics in Software Engineering

Peter Lo

Software Metrics

- Software Metrics refers to a broad range of measurements in computer software for a variety of purposes:
  - Applied to the software processes with the intent of improving it on a continual basis.
  - Used throughout a software project to assist in estimation, quality control, productivity assessment, and project control.
  - Used by software engineers to help assess the quality of technical work products and to assist in tactical decision making as a project proceeds.

Software Measurement

- Direct Measures:
  - Lines of Code (LOC) produced
  - Execution speed
  - Memory size
  - Defects reported over some period of time
- Indirect Measures:
  - Functionality
  - Quality
  - Complexity
  - Efficiency
  - Reliability and maintainability

Sized-Oriented Metrics

- Consider the “size” of the software
- A common factor or normalization value can be used as Lines of Code (LOC).
- Size oriented metrics can now include: Errors per KLOC (thousand lines of code), Defects per KLOC, Cost per LOC, Pages of documentation per KLOC.

Project	LOC	Effort	$(000)	pp. doc.	Errors	Defects	People
Alpha	12,100	24	168	365	134	29	3
Beta	27,200	62	440	1224	321	86	5
Gamma	20,200	43	314	1050	256	64	6
(etc.)							

Function-Oriented Metrics

- Use a measure of the functionality delivered by the application as a normalization value.
- Based on the calculated value of function points, can then be used like LOC to define software measures:
  - Errors per FP
  - Defects per FP
  - $ per FP
  - Pages of documentation per FP

$$FP = counttotal \times [0.65 + (0.01 \times \sum F_i)]$$

Function-Oriented Metrics

- Counttotal is the sum of all entries

measurement parameter	count		Weighting Factor simple	average	complex	
number of user inputs		×	3	4	6	=
number of user outputs		×	4	5	7	=
number of user inquiries		×	3	4	6	=
number of files		×	7	10	15	=
number of external interfaces		×	5	7	10	=
count = total						

Function-Oriented Metrics

- F(i) (where i = 1 to 14) are "complexity adjustment values".

COMPUTING FUNCTION POINTS

Rate each factor on a scale of 0 to 5:

0 No influence | 1 Incidental | 2 Moderate | 3 Average | 4 Significant | 5 Essential

F_i:

1. Does the system require reliable backup and recovery?
2. Are data communications required?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require on-line data entry?
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?
8. Are the master files updated on-line?
9. Are the inputs, outputs, files, or inquiries complex?
10. Is the internal processing complex?
11. Is the code designed to be reusable?
12. Are conversion and installation included in the design?
13. Is the system designed for multiple installations in different organizations?
14. Is the application designed to facilitate change and ease of use by the user?

Example

- For system Z:
  - Does the system require reliable backup and recovery? ➔ Average (3)
  - Are data communications required? ➔ Essential (5)
  - Is performance critical? ➔ Incidental (1)
- The summation for F(i) would be 3 + 5 + 1 = 9 for the first three values then.
- FP = 40 x [0.65 + (0.01 x 52)] = 46.8

Technical Software Metrics

- A high level architectural design metric proposed by Henry and Kafura makes use of fan-in and fan-out in program architecture.

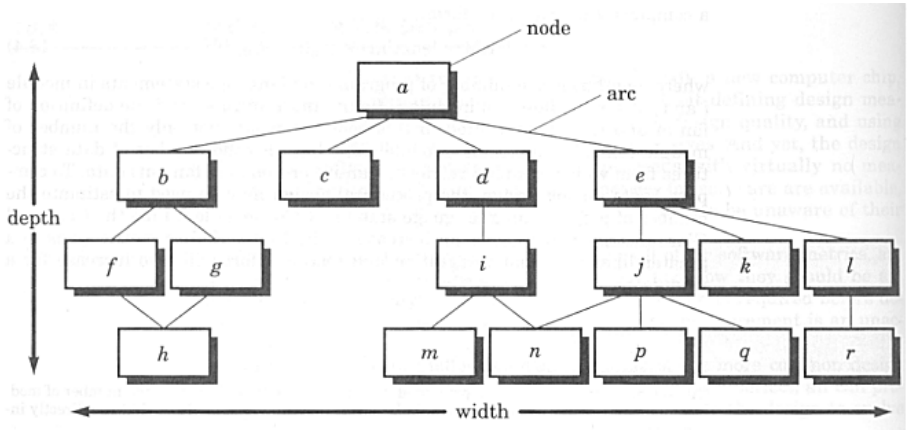
$$HKM = length(i) \times [f_{in}(i) + f_{out}(i)]^2$$

length(i) = the number of programming language statements in module i

f_{in} = fan-in for module i

f_{out} = fan-out for module i

Technical Software Metrics



Example

- For the previous diagram, assuming if length(i) is 100 programming language statements for module ***e***, HKM for module ***e*** would be equal to 400, since fan-in and fan-out for module ***e*** is 1 and 3 respectively.

Size & Arc-to-node Ration

- Fenton proposes other metrics in relation to program architecture:
  - **Size** = n (the number of nodes/module) + a (number of arcs/lines of control)
    = 17 + 18 = 35
- **Depth** = the longest path from the root (top) node to a leaf node. (Depth = 4)
- **Width** = maximum number of nodes at any one level of the architecture. (Width = 6)
- **Arc-to-node Ration**, r = a/n
  = 18/17 = 1.06

Software Maturity Index

$M_T =$ The number of modules in the current release

$F_c =$ The number of modules in the current release that have been changed

$F_a =$ The number of modules in the current release that have been added

$F_d =$ The number of modules from the preceding release that were deleted in the current release

- The software maturity index is then computed in the following manner:

$$SMI = \frac{[M_T - (F_a + F_c + F_d)]}{M_T}$$

Computer Aided Software Engineering

- CASE can be defined as the disciplined and structured engineering approach to software and systems development. It emphasizes structured methods, with defined and standardized procedures.
- CASE tools substantially reduce or eliminate many of the design and development problems inherent in medium to large software projects.
- CASE tools allow the software designer to focus on the systems architecture rather than on the actual implementation.

Classification of CASE Tools

- Information Engineering Tools
  - Represent business objects, their relationships, and how these data objects flow between different business areas within a company.
- Project Planning Tools
  - Focuses on two primary areas: software project effort, and cost estimation, and project scheduling.
- Risk Analysis Tools
  - Identify potential risks and develop a plan to mitigate, monitor and manage them.

Classification of CASE Tools

- Requirements Tracing Tools
  - Provide a systematic approach to the isolation of requirements, beginning with the customer request for proposal (RFP) or specification.
- Metrics and Management Tools
  - Capture project-specific metrics and provide an overall indication of productivity or quality.
- Analysis and Design Tools
  - Enable a software engineer to create models of the system to be built.
- Programming Tools
  - Encompass compilers, editors, and debuggers.

Benefits of Integrated CASE Tools

- Smooth transfer of information (models, programs, documents and data) from one tool to another;
- A reduction in the effort required to perform umbrella activities such as software configuration arrangement, quality assurance, and document production;
- An increase in project control that is achieved through better planning, monitoring, and communication;
- Improved coordination among staff members who are working on a large software project.

Integrated CASE Environments

- Provide a mechanism for sharing software engineering information among all tools contained in the environment;
- Enabled a change to one item of information to be tracked to other information items;
- Enable the users of each tool to experience a consistent look and feel at the human-computer interface;
- Support communication among software engineers;
- Collect both management and technical metrics that can be used to improve the process and the product.

Potential Risk Areas

- Loss of creativity on the part of the software developer
  - Software development is an occupation that requires skilled artistry
    - CASE-being tools that often follow strict rules for software development
    - Possibly stifle creativity
- CASE is still relatively new
  - Not all CASE tools are well integrated with each other or the software development process itself

Factors in Consideration

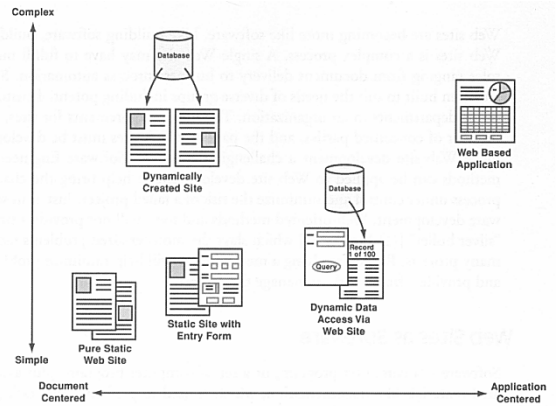
- Existing company standards and methods
  - The CASE tool selected should support existing development methodologies in use, and should not introduce new standards or methods of development.
- Existing computers and future computer procurement
  - Unless the organization is willing to support the purchase of any additional equipment required to use the intended CASE tool, the CASE tool selected should ideally run on existing computers.

Factors in Consideration

- The class of applications to be developed
  - The CASE package selected supports the type of application that the organization normally develops.
- Cost of the package.
  - CASE tools do not come cheaply
- Ability to customize package
  - A CASE package that allows for custom modification to suit specific notation or symbols used in the current development methodology adopted would ideal.

Web Site Engineering

- Progression of web sites from simple, passive document displays to software-centric applications



Web Site Engineering

- Static Web Sites
  - Collection of static documents created in HTML and tied together with links
- Static with Form-Based Interactivity
  - Forms are generally used to collect information from the user
- Sites with Dynamic Data Access
  - Provide a web-based front end for accessing a database
- Dynamically Generated Sites
  - Dynamically creates documents with content often generated from a database
- Web-Based Software Applications
  - An inventory tracking system in the form of a Web page

Problem Definition in Web

- Many web sites today typically lack a clear purpose
  - Often arising from a lack of conducting a proper problem definitional phase before development actually starts.
- The first aspect of defining purpose is to ask objective “why” and “what” questions from a macroscopic perspective
  - Why the site is needed in the first place
  - What the main motivation for the site is
  - The purpose that the web site will be intended to serve
  - And the audience that this web site should target

Problem Definition in Web

- Problem definition phase is facilitated through information gathering.
  - Fact-finding techniques- similar to that of traditional System Analysis- can be applied
- Measurement of success will also be required.
  - Hard and quantifiable metrics can be used.
- Determining who will develop the web site, and who will be paying for the development of this site

Requirement Analysis and Specification

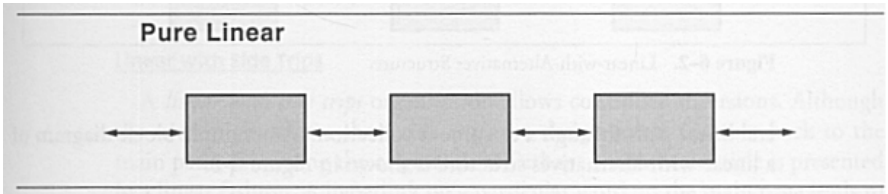
- Requirements should specify not just the functional expectations of the web site to be developed, but also constraints of both design and resource availability
- The type of web site will need to be defined in terms of its environment and infrastructure:
  - Public Internet, Intranet, or Extranet
  - The web site can also be either static, dynamic, or somewhere in between

Designing the Web site and System

- Information Design
  - Information retrieval should be facilitated by a design that not only makes the information presentation visually appealing, but consistently laid-out and organized.
- Web sites typically break information into fragments, which are further inter-connected through hyperlinks.

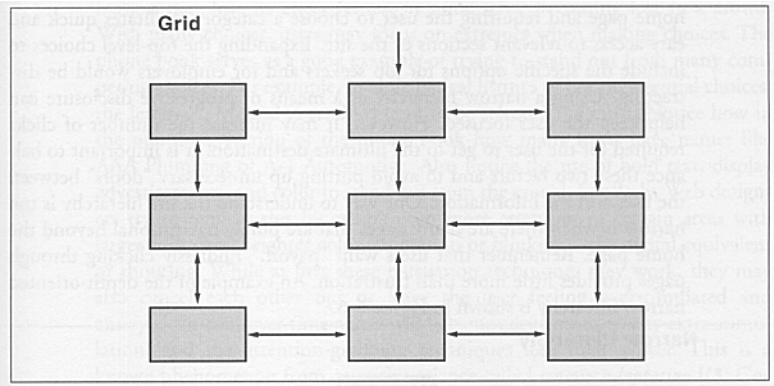
Linear

- Organizes information as an orderly progression in an order as defined by the designer of the web site



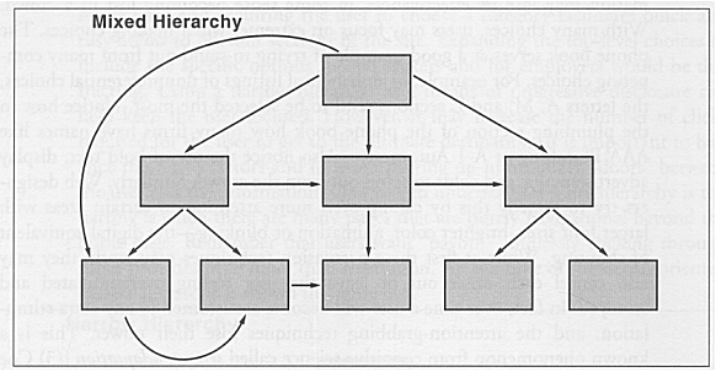
Grid Structures

- matrix-based, where fragments are linked in both horizontal and vertical perspectives.



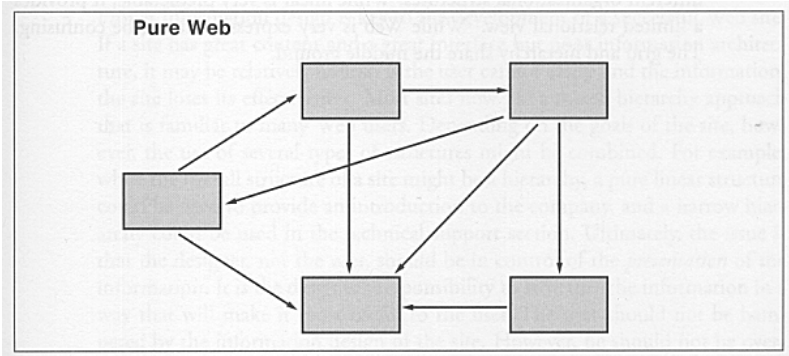
Mixed Hierarchy

- Combines the advantages of both the narrow and wider structures, but yet allowing the user to quickly navigate to any part of the web site as well

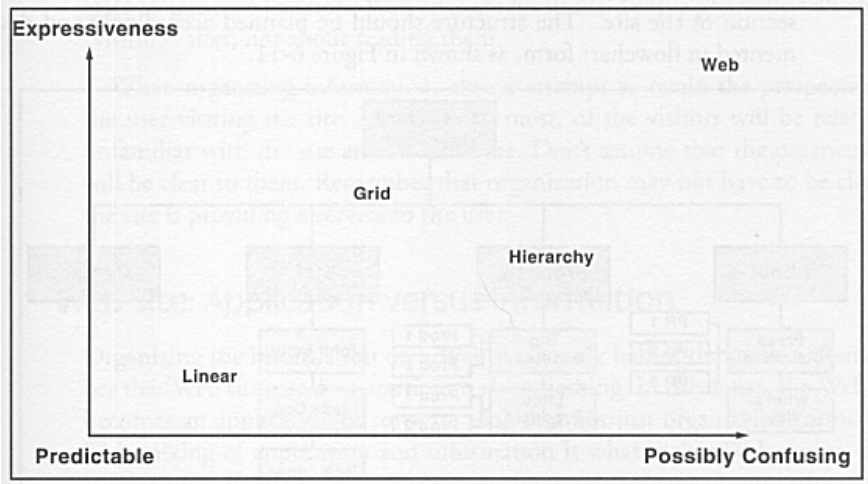


Pure Web

- Occurs when there is no discernible structure.
- The orientation here is extremely similar to a mesh-based topology of nodes.



Expressive vs. Predictability



Considerations

- Nature of data (e.g. text, graphic, video, sound) on the site
  - Smaller fragments of information often work better because the difficulty of reading information online
  - Collecting small fragments of information for printing purposes is difficult, larger document may be warranted.
- Overall purpose of the site should be considered as well
  - How much of control does the designer allow the user?
- Linear web sites while predictable provide a limited relational view.
- Pure webs though expressive also tend to be confusing
- Grid and hierarchical structures share the middle ground

Additional Considerations

- File Size
  - Download time is probably one of the biggest considerations about Web sites in the mind of the user.
- Color
  - If colors outside the palette of a computer system are used, and the user's video system does not support the specified color, the image may be degraded by attempting to approximate the color using a process called dithering.
- Fonts
  - Reading on screen is difficult enough, it could be made either easier or even more difficult through the selection of fonts.
- Layers and Margins
  - In complex web sites, it is possible to create images that have transparent regions so that they can be layered on background.
- Resolution and Screen Region
  - The effective region available on the screen should be taken into account.

Implementing the Web site: Client-side Technologies

- HTML is an example of one such technology that resides on the client side
- Helpers: frequently 3rd party application programs that will be called upon to execute some component of the web site from other HTML code.
  - Greatly extends web site functionality
  - Browser plug-ins, and is integrated and run as part of the browser itself.
- Another category of client-side technology includes Microsoft's ActiveX and Sun Microsystem's Java.

Implementing the Web site: Server-side Technologies

- Server side technologies include CGI (Common Gateway Interface)
- Alternative category of technology includes Parsed HTML Middleware, essentially are technologies that isolate functionality from the database on the server.
  - The query retrieve data from the server and presented in a dynamically generated web page
- Content technologies include static imagery, motion video, audio-effects, etc.
  - Compression technologies help, but often compression is lossy, which results in the loss of such data when the file is compressed to a smaller size.
- Strongly advisable that programming standards be applied and follow in the actual writing of code for the web page; and these typically include naming of files, pages and code structuring.

Testing the Web site

- Legal issues when web sites become more software-centric and in the event that the web site proves dysfunctional.
- Web site should be tested on multiple web browsers, and also on differing hardware and software environments.
- Functionality testing which ensures that the final implementation is compliant to the design specification.
- Unit testing is conducted at the component or page level first, and then move towards the integration of tested pages.
- Browser Testing is focused on the use of the web site within the browser.
- Configuration testing would also examine web site functionality across the spectrum of hardware; and delivery testing would examine the network and server aspects of the site in operation.

Post Implementation and Maintenance: Publicizing the Web site

- Important for public-access web sites, and all types of advertisement should be employed for this purpose. This includes word-of-mouth, printed, audio and visual media.
- Publicity also through the submission of the completed web site to search engines, extremely popular facilities that customers will use to look for information they require.
- Use of META tags within the web site facilitates this automated categorization, and thus should be used within the creation of the web site.

Post Implementation and Maintenance: Maintaining the Web site

- Web sites are only effective if the information provided on it is current.
- Design the web site from the ground up as a modular structure.
- Allow for abstraction from component to component, and that pages can now be modified without too much effect on other pages within the site.
- Content maintenance should address several aspects of the web site; what to be maintained, who is to maintain, and how frequently is maintenance to be performed.

Post Implementation and Maintenance: Feedback after Implementation

- Avenues for direct feedback through the provision of online surveys, feedback forms, discussion boards, or email addresses to which comments can be fed back.
- Indirect Feedback:
  - Information that is automatically generated without active user or developer involvement.
  - E.g. Hit counters, access logs