Metrics-Driven Enterprise Software Development

Subhajit Datta

Presentation Plan

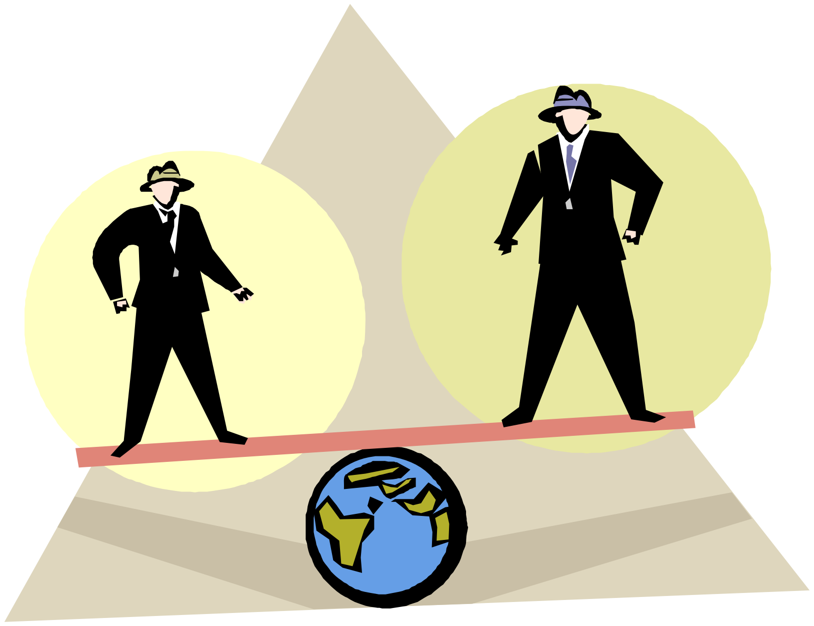
- The Metrics Odyssey
- Developing Enterprise Software
- A Holistic Role for Metrics
- A Quick Case Study
- Conclusion

The Software Metrics Odyssey

- 1970s – Quest for “laws” of software and complexity measures
  - [McC76], [Hal77], [BL79], …
- 1980s – Towards enterprise-wide metrics culture
  - [SHV86], [GC87], [DL87], …
- 1990s – OOAD measures and quality concerns
  - [LK94], [CK94], [Whi97], …
- 2000s - Measuring across the spectrum: product, people, process, project
  - [Lan01], [CSE02], [vS04], …

Rigor versus Expediency

Some metrics are strongly grounded in theory [CK94, Whi97,…]

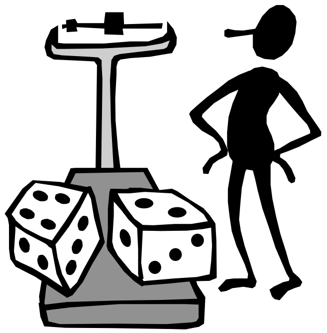


Others focus more on practice [DL87, LK94,…]

Choice of metrics depends on a project's needs

Metrics: Thinking Inside the Box

- So far, software engineering metrics have addressed size, defect density etc.
- These are useful as management “numbers”
- Or, for *a posteriori* scrutiny of product or process
- But metrics can do more …



Towards a More Holistic View

- Metrics driven development guides practitioners at every step of the life cycle
- Helping analysis, design, implementation, testing, and deployment of solutions with
  - Greater confidence
  - Purpose
  - Sensitivity to changing business needs
- Metrics are vital to the success of today’s enterprise software projects

Enterprise Software Systems

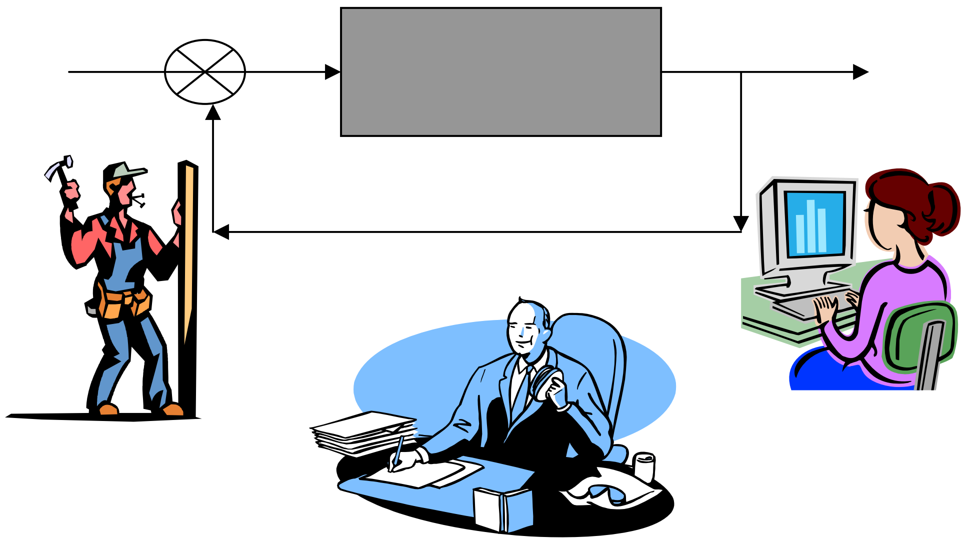
- Support large scale business processes, with high demands of
  - Usability, Reliability, Performance, Supportability
- Subject to continuous change in requirements, driven by
  - New business, competition, technology …
- Other characteristics include [Fow03]
  - Concurrent data access, complex business “illogic”, need to integrate with other enterprise systems

New Frontiers, Newer Challenges

- Enterprise software is at the cornerstone of major changes today
  - Global development
  - Teams distributed across continents
  - Open source software
  - Cross cultural contact
- Iterative and incremental development (IID) is widely used to build enterprise software

The Power of IID …

- The system *grows* incrementally, over iterations
- Users are able to test and give feedback
- Developers understand user needs better
- Managers can fine tune deliverables continually



And its Pitfalls

- What is the scope of an iteration?
- How to decide on the *granularity* of an increment?
- “Juicy Bits First”?
- Or, big risks at the beginning?
- Will iterations and increments finally *converge* into a cohesive system?
- Or, will they just give a potpourri of loosely slung modules?

Metrics from Within

- Metrics can monitor and regulate development from within, by helping
  - Define, evaluate, and decide in the process space
  - Resolve stakeholder objectives
  - Address the continuum of change
- How?
- Let us illustrate by example

A Quick Case Study

- *Yet Another Software Company* (YUSC) is building a Web application for *Just Another Client* (JAC)
  - Usual disclaimers about YUSC and JAC being purely fictional hold, of course!
- JAC is a large financial company, looking to offer “new and improved” online services to its customers
  - “Sprucing up” the existing website
  - Adding new functionality
  - Integrating a suite of legacy applications

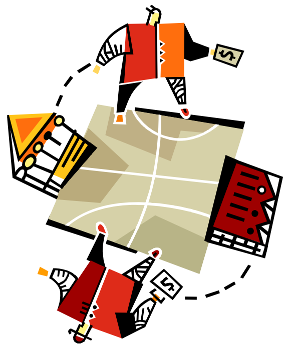
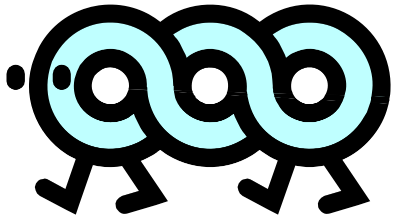
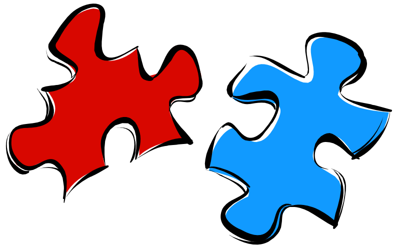
Points of Interest

- A project like this has several areas of concern
  - Tweaking of existing code
  - Design and implementation of new functionality
  - Interfacing with legacy applications
- Most importantly, requirements are prone to continual change
  - Stakeholders demand their respective pounds of flesh
  - Customers understand their needs only when developers flesh them out

Two Typical Situations

- Requirements are oscillating too much
- Unending cycles of design change

- Every iteration seems to start afresh
- Increments do not grow the system

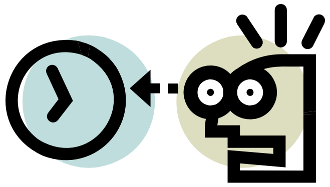




The story of YUSC and JAC …

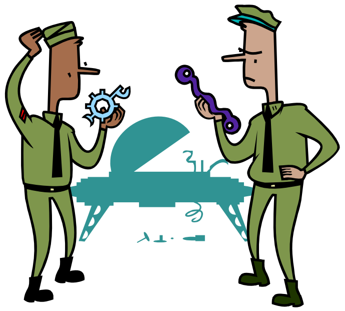
Doing it the Usual Way

Confer with customers

Hope requirements freeze



Over and over again,
as deadline looms



Tweak the system

Try and figure what changed

Doing it the Metrics Way

- Is there a better way?
- Let us see how two simple and intuitive, tailor-made metrics can help us
  - *Morphing Index*
  - *Specific Convergence*

Morphing Index

$$RI(k) = \frac{\sum_{i=1}^{m} w(C_i)}{\sum_{j=1}^{n} w(M_j)}$$

Comparing the Morphing Index values across iterations help quantify the changes in design

- How components collaborate via messages at some iteration *k*
- *$w(C_i)$* = weight of the i'th component, based on whether it is *primary, secondary,* or *tertiary*
- *$w(M_j)$* = weight of the j'th message, based on whether it is *creational, computational,* or *transmissional*

Specific Convergence

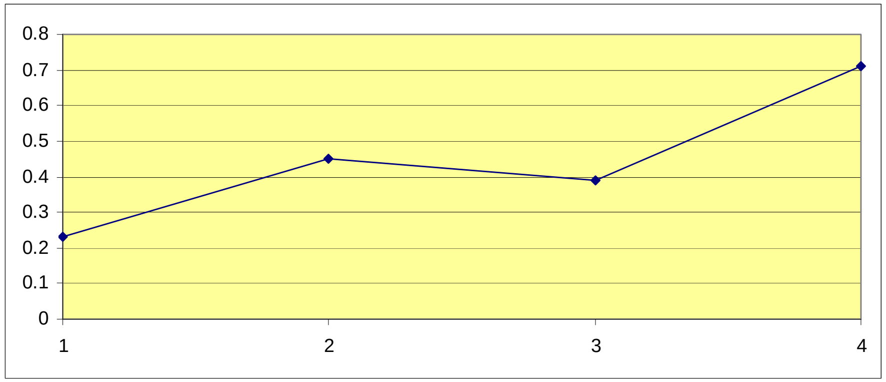
$$SC(k) = \frac{\sum_{i=1}^{m} RF(DU_i) * EF(DU_i)}{\sum_{j=1}^{n} RF(DU_j) * EF(DU_j)}$$

The Specific Convergence value for each iteration indicates how close the development effort is getting to convergence

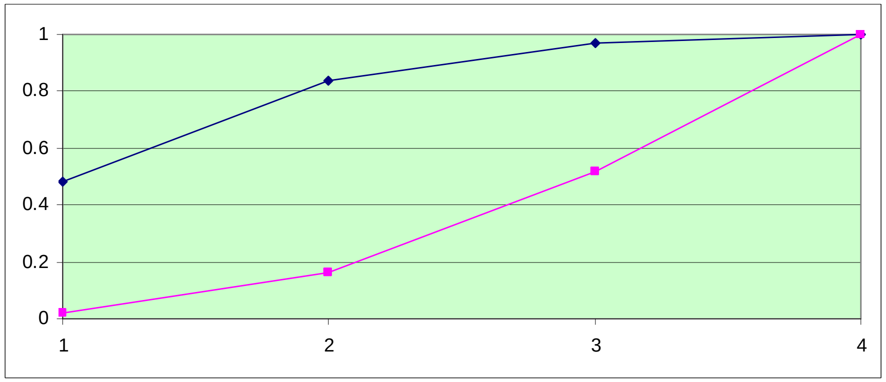
- How activities in an iteration *k* contribute towards the final deliverable
- DU_i = i'th *Deliverable Unit*
- $RF(DU_i)$ = *Risk Factor* associated with DU_i
- $EF(DU_i)$ = *Effort Factor* associated with DU_i

The Metrics Message

k	RI(k)
1	0.23
2	0.45
3	0.39
4	0.71



Variation of design across iterations; the curve should flatten as the project progresses



	SC(1)	SC(2)	SC(3)	SC(4)
Plan A	0.48	0.84	0.97	1
Plan B	0.02	0.16	0.52	1

In choice of iteration plans, Plan A tackles higher risk and higher effort first, Plan B keeps them for later

Net Value Add

- Simple, intuitive metrics like the *Morphing Index* and *Specific Convergence* help practitioners
  - Moderate the development process at the micro level
  - Manage customer expectations better
  - Evaluate changes and their effects
  - Decide on the most expedient course of action
- Without metrics, all of these are
  - Ad-hoc
  - Instinct driven
  - Often, unreasonable

Making Your Own Metrics

- How do you get good metrics, or metrics that are good for you?
- You can try out different metrics, and see how work, or do not work
- Or, you can *make* your own metrics
- Metrics making is the surest test of your grasp on a scenario

Metrics: N Commandments …

- No silver bullet
- Metrics hunt in groups
- There are always assumptions
- Customize a metric when necessary
- Be ready to build your own metrics
- Keep it simple
- Collect and compile over time
- Use automation
- Be clear about scope and workings
- Metrics give feedback – the rest is yours

Conclusion

- A metrics culture is essential for the latest challenges of enterprise software development
- Metrics driven development help practitioners analyze, design, implement, test, and deploy faster and better solutions
- Simple, intuitive metrics can greatly help monitoring and decision making within the development process
- With experience and innovation, practitioners can build and apply their own metrics

References …

- [McC76] T.J. McCabe. A software complexity measure. In *IEEE Trans. Software Engineering, vol. SE-2*, December 1976, pages 308–320, 1976.
- [Hal77] Maurice H. Halstead. *Elements of Software Science*. Elsevier North-Holland, Inc.,1977.
- [BL79] L. A. Belady and M. M. Lehman. *The characteristics of large systems*, 1979. In Research Directions in Software Technology, Page 106-138, MIT Press.
- [SHV86] S.D.Conte, H.E.Dunsmore, and V.Y.Shen. *Software Engineering Metrics and Models*. The Benjamin/Cummins Publishing Company, Inc, 1986.
- [GC87] Robert B. Grady and Deborah L. Caswell. *Software metrics : establishing a company-wide program*. Prentice Hall, 1987.
- [DL87] Tom DeMarco and Timothy Lister. *Peopleware : productive projects and teams*. Dorset House Pub. Co., 1987.
- [LK94] Mark Lorenz and Jeff Kidd. *Object-oriented software metrics : a practical guide*. PTR Prentice Hall, 1994.

References contd. & Thank You!

- [CK94] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Trans. Softw. Eng*., 20(6):476–493, 1994.
- [Whi97] Scott A. Whitmire. *Object-oriented design measurement*. Wiley Computer Pub, 1997.
- [Lan01] Michele Lanza. The evolution matrix: recovering software evolution using software visualization techniques. In *IWPSE '01: Proceedings of the 4th International Workshop on Principles of Software Evolution*, pages 37–42, New York, NY, USA, 2001.
- [CSE02] CSE-Center for Software-Engineering. Cocomo. http://sunset.usc.edu/research/COCOMOII/, 2002.
- [vS04] Rini van Solingen. Measuring the ROI of software process improvement. *IEEE Softw*., 21(4):32–34, 2004.
- [Fow03] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003.

Thank you! Questions, comments, feedback?