THE UNIVERSITY OF BRITISH COLUMBIA

On Software Architecture, Agile Development, Value and Cost

Philippe Kruchten

SATURN
Pittsburgh, April-May 2008

1

Philippe Kruchten, Ph.D., P.Eng., CSDP

Professor of Software Engineering
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC Canada
pbk@ece.ubc.ca
604 827-5654

Founder and president

Kruchten Engineering Services Ltd
Vancouver, BC Canada
philippe@kruchten.com
604 418-2006

2

Outline

- A story
- Agile processes and software/system architecture
- Value and cost
  - Cost
  - Value
- Value and cost of architecture
- A proposed simple strategy to put value on architecture

3

Story of a failure

Synthetic example from
2 projects (finance, aerospace)

- Large re-engineering of a complex distributed world-wide system; 2 millions LOC in C, C++, Cobol and VB
- Multiple sites, dozens of data repositories, hundreds of users, 24 hours operation, mission-critical ($billions)
- xP+Scrum, 1-week iterations, 30 then up to 50 developers
- Rapid progress, early success, features are demo-able
- Direct access to “customer”, etc.
- *A poster project for scalable agile development*

5

Hitting the wall

- After 4 ½ months, difficulties to keep with the 1-week iterations
- Refactoring takes longer than one iteration
- Scrap and rework ratio increases dramatically
- No externally visible progress anymore
- Iterations stretched to 3 weeks
- Staff turn-over increases; Project comes to a halt
- Lots of code, no clear architecture, no obvious way forward

6

Agile Processes & Architecture

- No BUFD (no Big Up-Front Design)
- Incrementally develop (& deliver) value
- xP: Metaphor
- FDD: Features
- Earned-value system -> burn-down charts
- Very short iterations (a.k.a. sprints)
- Refactoring
- Gradual emergence of the design…

7 Agile Alliance 2001

Context does Matter

- For medium to large software-intensive systems, or in novel systems, an architecture will NOT gradually emerge as the result of constant refactoring.

- The Wall
- Architecture lacks sex appeal

8 Kruchten 2007

Value and Cost

- Value: to the business (the users, the customers, the public, etc.)
- Cost: to design, develop, manufacture, deploy, maintain
- Simple system, stable architecture, many small features:
  - statistically value aligns to cost
- Large, complex, novel systems ?

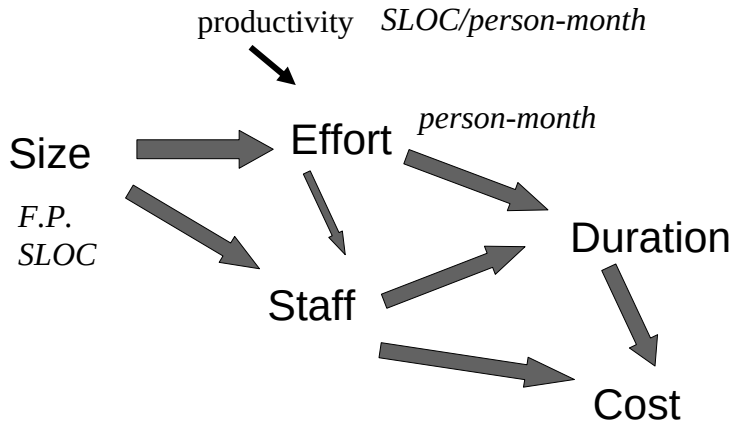
9

Cost

- The old:
  - Function points, SLOC
    => time
    => $$$
- The "new":
  - Story points, ideal days, velocity
  - Backlog (Scrum) used to drive increment content
  - Self-tuning process

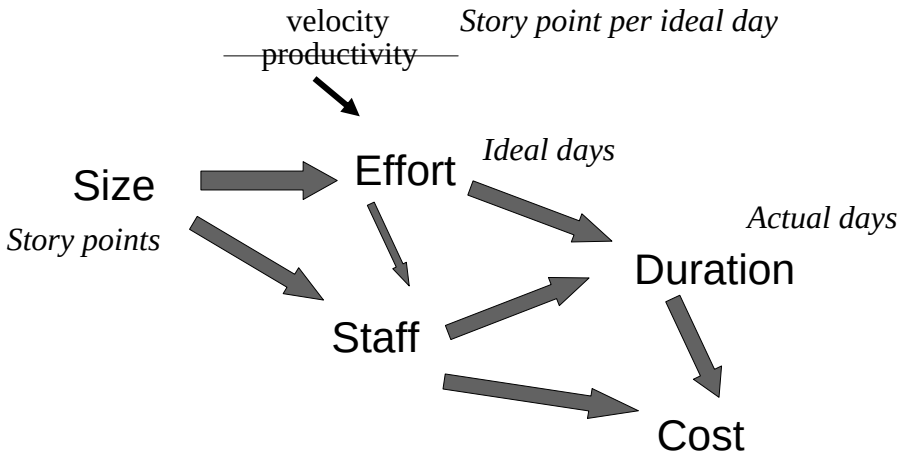
10

Cost: the classical view



11

Cost: the agile view



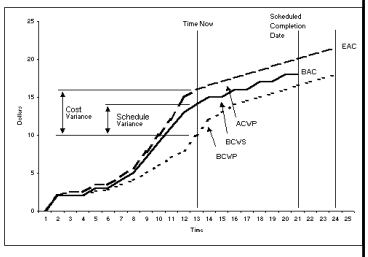
12 Cohn 2006

Value ?

- Traditionally in dollars
- Decomposition, interdependencies
- Priorities (used in XP's planning game)

Beck 2001

- Not very successful
- Earned Value System muddies the water even more
  - Are we speaking about value?
  - Cost? Both?



- What is the value of architecture? About nil
  - Seen only as an additional cost

13

A Proposed Strategy

- Cost in points
- Value in units

- Get valuation done in relative units (what a unit mean is irrelevant)
- Try to break-down big value elements

- Keep value independent from any notion of cost

- *Relate to: 100-point method, Karl Wiegers' prioritization scheme, …. AHP, Theory W (?)*

14 Leffingwell 2003, Wiegers 1999, Saaty 1990, Boehm 1989

(cont.) Valuing architectural design

- Value architecture by taking units from top level, user-visible features, and flowing them down to non-visible development elements

- How?
  - The "revenue taxation" model: 12 % across
  - The "head tax" model: collect fixed amount of units
  - The "pay-per-use" model: pay a percentage only if it makes use of it
  - *The "auction" model ??*

15

(cont.) Flowing-down value

- Need a rich dialog between
  - Developers
    - Architecture, design
    - Dependencies between design “chunks”
    - Costing development in points
  - Business representatives
    - Features, prioritization
    - Valuation in units

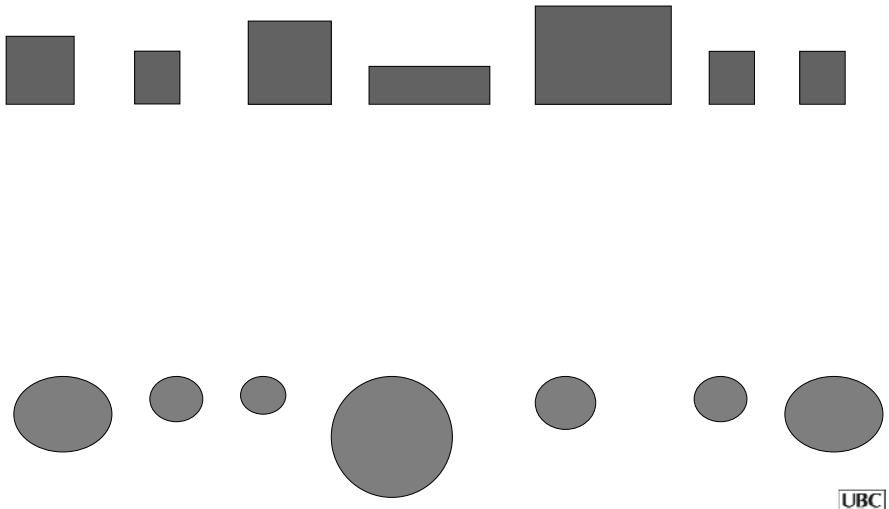
… during early phases to jointly “flow down” value to development elements

- ICM: valuation and architecting phases

Boehm & Lane 2007

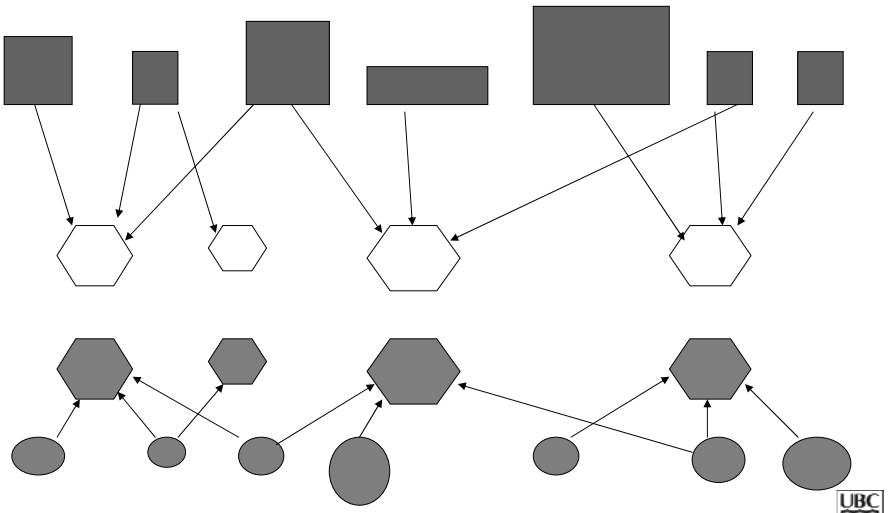
16

Points (cost) and Units (value)



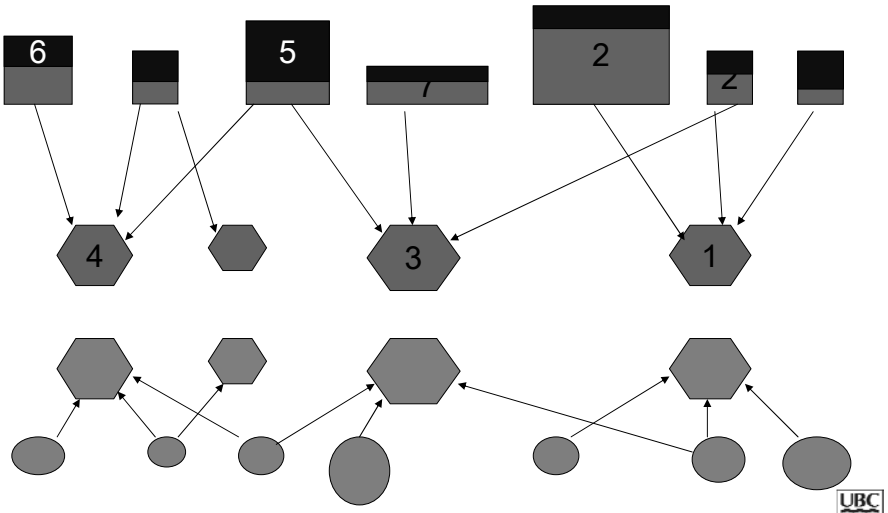
18

Points (cost) and Units (value)



23

Points (cost) and Units (value)



25

Some rules for the game

- Total value is constant, through flowdown, hence throughout architectural design
- Adding requirements adds value (using relative units to evaluate)
- Total cost evolves with architectural design (it should go down, or maybe not)
- Costs re-evaluated as development progress (agile concept of velocity)
- *Value cannot be changed after implementation to change priorities*
- Keep costs and values well separated
- Can't deliver architectural bits without user visible bits (and vice versa)
- MMF

26

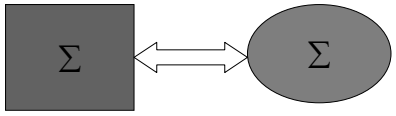
Key points

- Value often not correlated to cost
- Express value in relative terms, not absolute $$$
- Proceed to architectural design
- Re-allocate some of the user-visible value to non-visible element, with constant sum
- MMF = minimum marketable features
- Schedule iteration sequence based on fully valued development elements + dependencies
- Better fit to a *revised* Earned-Value System
- Many benefits in the dialog itself (value is in the journey)

27

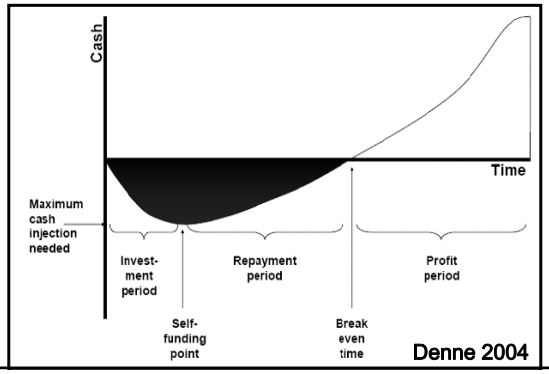
Value (units), Costs (points), and real $$$

Rev. $$$ / ∑ units



Dev. $$$ / ∑ points

Time



28

Alternative approaches

- CBAM = Cost Benefit Analysis Method
  - Chap 12 in Bass, Clements, Kazman 2003
- IMF: Incremental Funding Method
  - Denne & Cleland-Huang, 2004
- Analytic Hierarchy Process
- Evolve* - Hybrid
  - Günther Ruhe & D. Greer 2003, etc…

29

CBAM: Cost Benefit Analysis Method

- Concept: Utility
  = Value (?)
  Utility-response curves: linear, steps, exp.,…
- Concept: Scenario
  - And priorities
- Architectural strategies
  - Their value, and utility
  - Their cost
- Benefit and ROI (Return on Investment)

30

IFM: Incremental Funding Method

- MMF = Minimum marketable Features
- AE = Architectural elements
  - Cost
  - MMF depends on AE
- Time and NPV = Net Present Value
- Strands = Sequences of dependent MMFs

31

… but the same issues

- How to assign realistic
  - Value
  - Cost
  - Priority
- to each chunk of software?
- And how to make it appealing to the agile projects?
  - Separation between the visible (feature) and the invisible (architectural element)
  - Make it practical for small and medium teams

32

Conclusion

- There is both value and cost in Software Architecture
- They may be articulated in simple, non financial terms
- to assist planning iterative development
- and avoid “hitting a wall”.

- Start small and simple.
- Get fancy later.

33

References

- Agile Alliance (2001) *Manifesto for Agile Software Development*. http://agilemanifesto.org/
- Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice* (2nd ed.). Reading, MA: Addison-Wesley.
- Beck, K., & Fowler, M. (2001). *Planning Extreme Programming*. Boston: Addison-Wesley.
- Boehm, B. and Lane, J.A. (2007) *Using the incremental commitment model to integrate system acquisition, system engineering, and software engineering*, University of Southern California, Los Angeles, September 2007.
- Boehm, B. & Ross, R. (1989) "Theory-W Software Project Management: Principles and Examples." *IEEE Transactions on Software Engineering 15* (4) 902-916.
- Cohn, M. (2006) *Agile Estimating and Planning*. Upper Saddle River, N.J.: Prentice-Hall,
- Denne, M., & Cleland-Huang, J. (2004). *Software by Numbers: Low-Risk, High-Return Development,* Prentice Hall.

36

References (cont.)

- Denne, M., & Cleland-Huang, J. (2004). The Incremental Funding Method: Data-Driven Software Development *IEEE Software, 21*(3), 39-47.
- Karlsson, J. & Ryan, K. (1997). A Cost-Value Approach for Prioritizing Requirements, *IEEE Software*, 14 (5) 67-74.
- Kruchten, P. (2007). Voyage in the Agile Memeplex: Agility, Agilese, Agilitis, Agilology. *ACM Queue, 5*(5), 38-44.
- Leffingwell, D. & Widrig, D. (2003) *Managing Software Requirements: A Use Case Approach,* 2nd ed. Boston, MA: Addison-Wesley.
- Maier, M.W. (2006) System and software architecture reconciliation. *Systems Engineering*, *9* (2) 146-159.
- Ruhe, G. and Ngo-The, A. (2004) Hybrid Intelligence in Software Release Planning. *International Journal of Hybrid Intelligent Systems,* 1, pp 99–110.
- Saaty, T. (1990). How to make a decision: The analytic hierarchy process. *European journal of operational research*, 48(1), 9-26.
- Wiegers, K. (1999). First Things First: Prioritizing Requirements. *Software Development Magazine, 7*(9), 48-53.

37