Critical Factors Characterizing Projects and Lifecycle Models

30th Pacific NW Software Quality Conference (PNSQC)
October 8-10, 2012

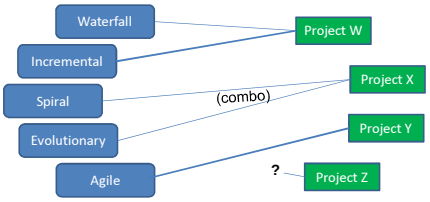
Kal Toth, kalmanctoth@gmail.com
Herman Migliore, herm@cecs.pdx.edu

Context (1)

Methodologists / proponents of development lifecycles:

- Boost their favorite approaches … often with built-in bias
- Rarely a balanced perspective i.e. both benefits and challenges
- Rarely discuss which problems they are best suited to solve or why

Assertion: A given lifecycle is not "best" for every project!



Context (2)

Prudent engineering manager should seek objective evidence:

- Which life-cycle process(es) best support the project's unique needs?
- Which … yield positive ROI given core competencies?
- What are the risks of project overrun and downstream failures?

Should we take the project? retrain? pass?

Aim of this Paper

Advance the discovery of a process that will select [most] appropriate lifecycle for the planned project

- Assumption: Certain critical factors characterize lifecycles & projects
- Hypothesis: a practical characterization matching process exists

Preliminary Remarks:

- Herein postulating and describing a possible approach
- Encourage motivated practitioners to explore this problem further
- Hopeful side-effects of this paper:
  - Recognition that any given lifecycle is not suitable for all projects
  - Practitioners should consider adapting & combining lifecycles

Flow of this Paper (contents)

Puts forward 8 critical factors characterizing lifecycles & projects

- Factored out existing process cultures, competencies, and biases

Describes suggested process for matching lifecycles & projects

Characterizes five common / generic lifecycles:

- Waterfall, Incremental*, Spiral*, Evolutionary*, Agile* [*iterative variants]
- Their distinguishing features, merits, shortcomings
- Assesses & depicts lifecycles in terms of the 8 factors

Illustrates selection process with 2 hypothetical projects

8 Critical Factors Characterizing SW Dev.

Quality/Maintainability: Completeness, sufficiency and currency properties of the processes, delivered software, and delivered documentation (reqts, design, test etc.)

Application Domain: Relative problem difficulty ranging from casual web-sites, games, financial transaction systems, health IT systems, medical devices, aircraft navigation systems, space vehicles

Size and Complexity: small, simple, linear programs < 1000K vs. large, complex systems > 500K LOC (size and complexity tend to correlate)

Uncertain Requirements: Degree of requirements precision / ambiguity whether documented or not

Requirements Volatility: Rate at which customer, context, and functional / non-functional requirements change (may be related to prior item)

User Involvement: Users review and approve documents vs. getting intensively involved in writing user stories, requirements specs, design, software development, testing, and acceptance …

Urgency/Time to Market: Relative urgency to deliver to market or to the customer

Progress Visibility: May be provided by way of informal functional demonstrations, high level progress reports, reporting of tasks, modules, and deliverable completion levels, various metrics

Postulated Model

1. [Pre-condition]: each lifecycle characterized in terms of 8 critical factors
   - collect, categorize and analyze data from a large sample of actual projects
2. For each project, compile collected data from stakeholders (incl. cust/users):
   - objectives, context, assumptions, resources, constraints and priorities
3. Prune candidate lifecycles eliminating the obvious including incompatibilities with competencies and culture
4. From compiled project data, assess / estimate nominal values for the 8 critical characterizing factors
5. Apply a matching algorithm to compare project's characterization data with each lifecycle model's profile
   - Goal … estimate the "degree of fit" selecting the "best"
6. Conduct sensitivity and trade-off analyses:
   - Vary project characterization data
   - Incorporate project costing and scheduling estimates

Figure 2: Waterfall Lifecycle Model Characterized

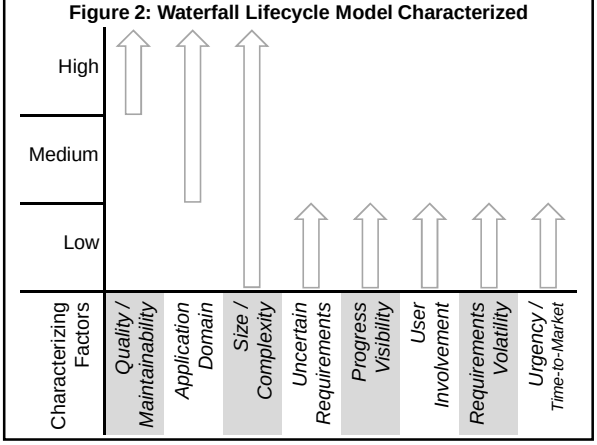
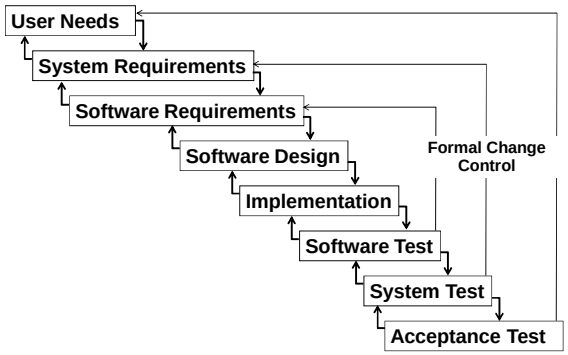


Figure 1: Waterfall Lifecycle Model [11]



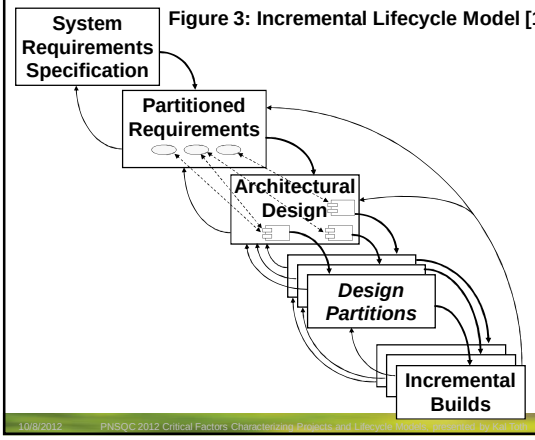
Iterative Lifecycle Models

	Iterative
Attributes	Variants: WP, Incremental, Spiral, Evolutionary and Agile Repeated cycles, ongoing rework Parallel / concurrent development
Benefits Advantages	Parallel / concurrent development allows better schedules than waterfall Early discovery of problems Customer feedback – more likely to meet requirements Visibility into progress Process improvement (PI), lessons learned (LL)
Shortcomings Disadvantages	Harder to manage project than waterfall Harder to write contracts and subcontracts

Waterfall Model

	Waterfall
Attributes	Relatively sequential with development phases, major milestones, & specified deliverables reviewed by stakeholders At each phase loop back to prev. phase to correct problems Formal change control procedures to correct problems in earlier phases which may modify costs and schedule
Benefits Advantages	Fosters thorough requirements, architecture and design before implementation Formalizes documentation and deliverables which facilitates project and contract management
Shortcomings Disadvantages	Not very adaptive to project changes or market demands Project visibility limited to documentation Customer and user feedback and refinement (too) late to incorporate lessons learned into the current project

Figure 3: Incremental Lifecycle Model [11]



Incremental Lifecycle Model

	Incremental
Attributes	An iterative process that partitions large complex problems into independent parts, some of which may be mission-critical, and concurrently develops and integrates the parts Requirements & architecture should be stable prior to partitioning and change controls should be in place after baselining Appropriate for multiple delivery and release of capabilities
Benefits Advantages	Supports concurrent development, partial/progressive deliveries Each part can be managed relatively independently Separate parts can be monitored separately enhancing visibility
Shortcomings Disadvantages	Mapping requirements to increments can be challenging Unanticipated changes to requirements & architecture can break across increments and imply major rework later on

Spiral Lifecycle Model

	Spiral
Attributes	A risk-driven plan-oriented iterative model where each spiral is a development iteration that aims to establish a plan for the next spiral (a.k.a. iteration). Risk assessments prior to each spiral determine the activities scheduled for a given spiral/iteration Reviews at the end of each iteration include an assessment of "lessons learned" that feed the next spiral
Benefits Advantages	Early iterations (spirals) systematically focus on consolidating the requirements and exploring technical problem areas through prototyping and simulating Later iterations transition in more waterfall-like iterations of development – concurrent spirals represent increments of development
Shortcomings Disadvantages	Project management and contracting more challenging as it requires more discipline to incorporate concurrency, risk assessment, and lessons learned

Figure 4: Incremental Lifecycle Model Characterized

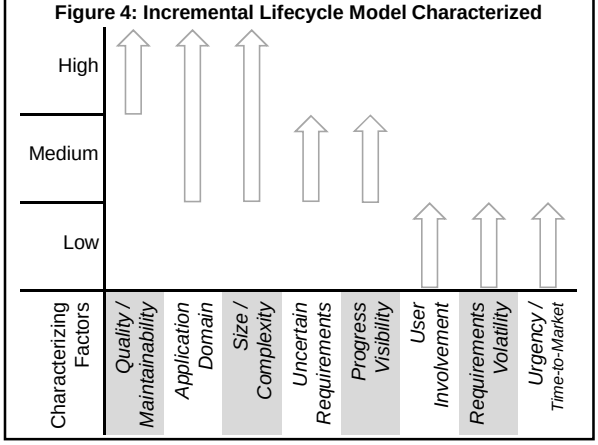


Figure 6: Spiral Lifecycle Model Characterized

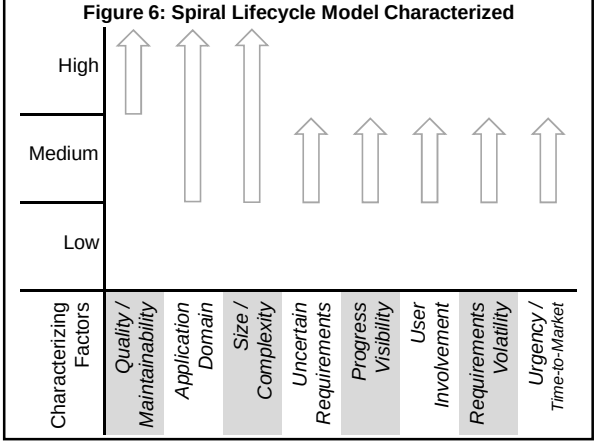


Figure 5: Boehm's Spiral Model [11]

Adapted from [7] (considerably simplified)

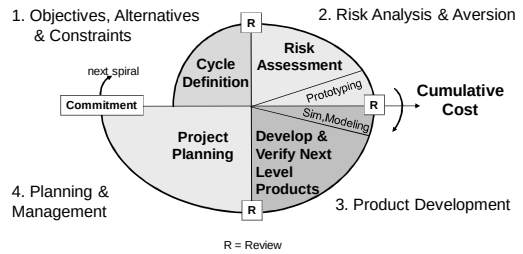
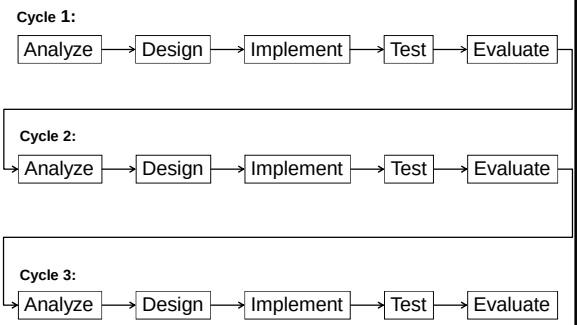


Figure 7: Evolutionary Lifecycle Model [11]



Evolutionary Lifecycle Model

	Evolutionary
Attributes	An iterative exploratory development model for solving hard (non-trivial) technical problems and uncertainties Work products of this model are designed to discover technical solutions and elicit customer / user feedback Work products of evolutionary development are not considered to be of operational/deliverable quality
Benefits Advantages	Focuses project stakeholders (developers, managers, customers, and users) on feasibility and requirements rather than a solution. Detailed functions and features, as well as product qualification tasks such as reviews and testing can be avoided.
Shortcomings Disadvantages	There is a danger that managers and customers assume the prototypes to be of deliverable quality - they are not! And their expectations of actual progress will be inflated

Agile Lifecycle Model

	Agile Development
Attributes	An incremental strategy that builds solutions from "stories" over short development iterations (typically 1-2 weeks) Focus is on working software over documentation Embraces change and close customer involvement Stories are typically prioritized and put into a backlog Planning is typically "time-boxed" Some methods advocate pair-programming (e.g. XP) Often employ "test-driven development" (TDD)
Benefits Advantages	Adaptive to change due to light-weight documentation Higher acceptance rate due to close customer involvement Informal stories and constant design refactoring reduces time and schedule defining requirements
Shortcomings Disadvantages	Customers don't always participate Frequent re-factoring can cause brittle systems Vulnerable to turnover and lack of documentation Harder to write contracts to meet vaguely stated requirements May not scale to large, complex and mission-critical projects

Figure 8: Evolutionary Lifecycle Model Characterized

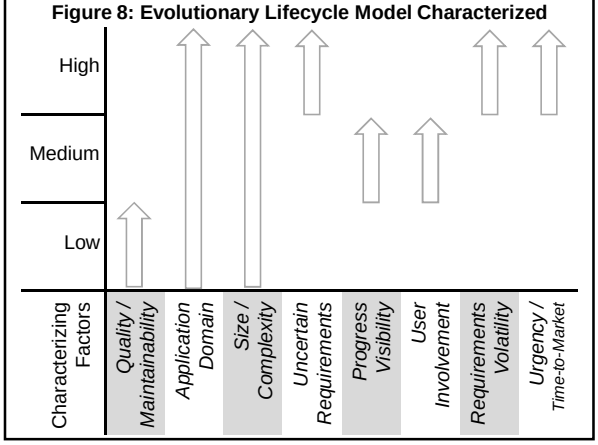


Figure 10: Agile Lifecycle Model Characterized

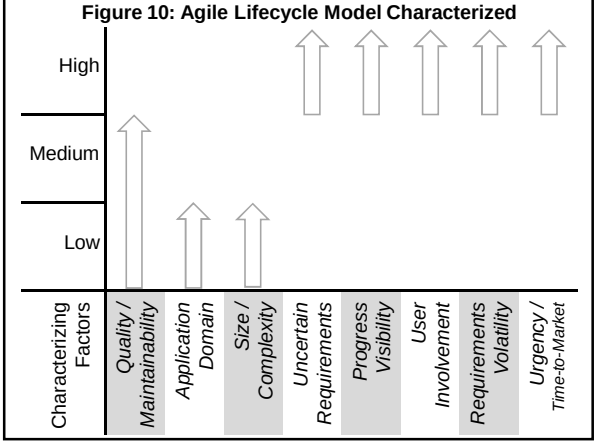
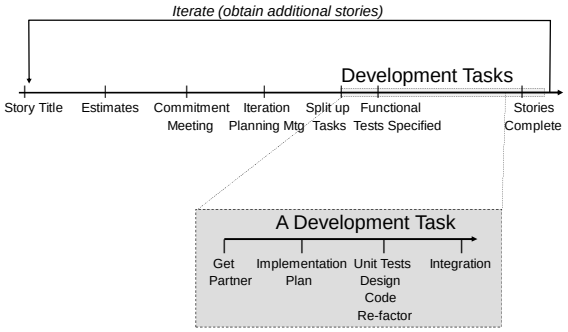


Figure 9: The Life of an Agile Story [11]

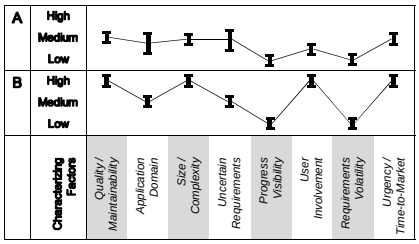


Summary Characterization of Lifecycle Models

	Waterfall	Incremental	Spiral	Evolutionary	Agile
Quality/ Maintainability	H	H	H	L	L-M
Application Domain	M, H	M, H	M, H	L, M, H	L
Size / Complexity	L, M, H	M, H	M, H	L, M, H	L
Uncertain Requirements	L	M	M	H	H
Progress Visibility	L	M	M	M	H
User Involvement	L	L	M	M	H
Requirements Volatility	L	L	M	H	H
Urgency	L	L	M	H	H

Selecting a Lifecycle

(Hypothetical Projects A and B)



Summary

Goal: explored possibilities – not reams of data

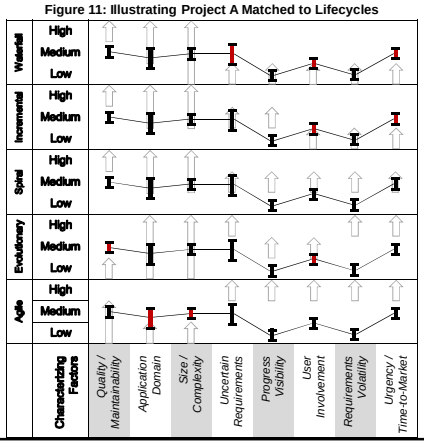
Proposed 8 critical factors for characterizing lifecycles & projects

Proposed a process for characterizing lifecycles & projects:

Biggest challenges:

- Collecting & analyzing data to empirically characterize lifecycles
- Semi-quantitative techniques for characterizing new projects
- Developing an effective project-to-lifecycle matching process
  - May be possible to adapt software estimating and COTS selection techniques [refs]

Figure 11: Illustrating Project A Matched to Lifecycles



Questions?

Welcome constructive criticism and validation

Hopefully this will motivate research & assessment projects that build on the ideas presented

Figure 12: Illustrating Project B that Fails to Match a Lifecycle