Critical Factors Characterizing Projects and Lifecycle Models

30th Pacific NW Software Quality Conference (PNSQC)
October 8-10, 2012
31th Pacific NW Software Quality Conference (PNSQC)
October 14-15, 2013

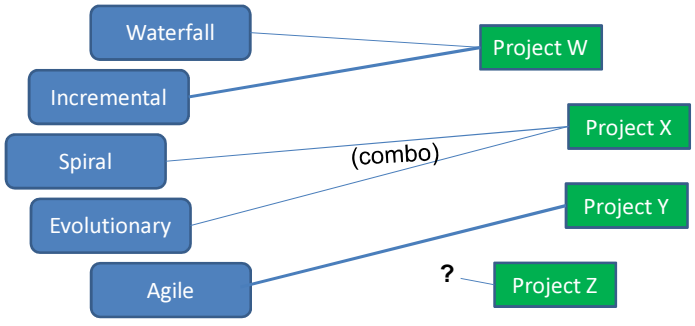
Kal Toth, kalmanctoth@gmail.com
Herman Migliore, herm@cecs.pdx.edu

Context (1)

Methodologists / proponents of development lifecycles:

- Boost their favorite approaches … often with built-in bias
- Rarely articulate a balanced perspective (benefits vs. challenges)
- Rarely discuss problems they are best suited to solve/why

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



Context (2)

Prudent SPM should seek objective evidence:

- Which life-cycle process(es) best support the project?
- …. align best with core team competencies, culture, tools?
- …. contain project risks - cost & schedule overruns, failure?

Should we take the project? retrain? pass?

Should …. select, adapt and/or combine processes to address the problem at hand …

…. BUT HOW?

What this is About

Aim: Advance the discovery of a process that will select the most appropriate lifecycle for a planned project

Assumption: Critical factors characterize lifecycles & projects

Hypothesis: a project characterization & matching process can be synthesized

Postulating: a possible approach / model

Encouraging: further exploration

Hopeful side-effect: wider recognition that a given lifecycle is not suitable for <u>all</u> projects

Flow of this Paper (contents)

Puts forward 8 critical factors characterizing lifecycles & projects

- ignores other factors e.g. process cultures, competencies, and biases

Suggests a process for matching lifecycles and projects

Characterizes five generic lifecycles (ignores many variants):

- Waterfall, Incremental*, Spiral*, Evolutionary*, Agile*
  *iterative variants
- Distinguishing features, merits, shortcomings of each in terms of the selected 8 critical factors

Illustrates the idea with two hypothetical projects

Possible Lifecycle Selection Process

CREATE PROJECT DATABASE

1. Consider lifecycle models in common practice [start with 5 generic ones]
2. Each lifecycle characterized by **M** critical factors [say 8]
3. Collect large enough sample of project data to characterize each lifecycle [i.e. the relative merit/capabilities of each … this is hard work]

EXECUTE PROJECT-TO-LIFECYCLE MATCHING PROCESS

1. For the project at hand:
   a) Eliminate the obvious: lifecycles that don't align with competencies, culture, tools
   b) Estimate the characterization factors [i.e. attributes of the problem]
2. For each lifecycle:
   a) Assess the "degree of fit" between project's characterization data and each lifecycle's characterization data [hard part]
   b) Conduct sensitivity and trade-off analyses:
      - Vary project's characterization data
      - Validate selection by estimating project costs, schedules and risks

8 Critical Factors Characterizing SW Dev.

High

Quality/Maintainability: Completeness, sufficiency and correctness 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)

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

Medium

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

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

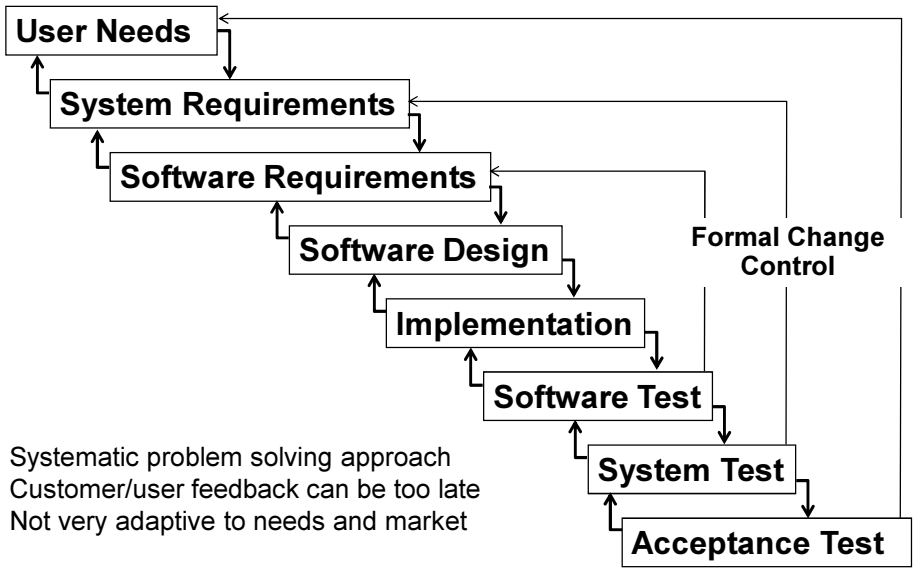
Urgency/Time to Market: Relative urgency to deliver to market or to the customer

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

Low

PNSQC 2012 & 2013 Critical Factors Characterizing Projects and Lifecycle Models, presented by Kal Toth 7

Figure 1: Waterfall Lifecycle Model [11]



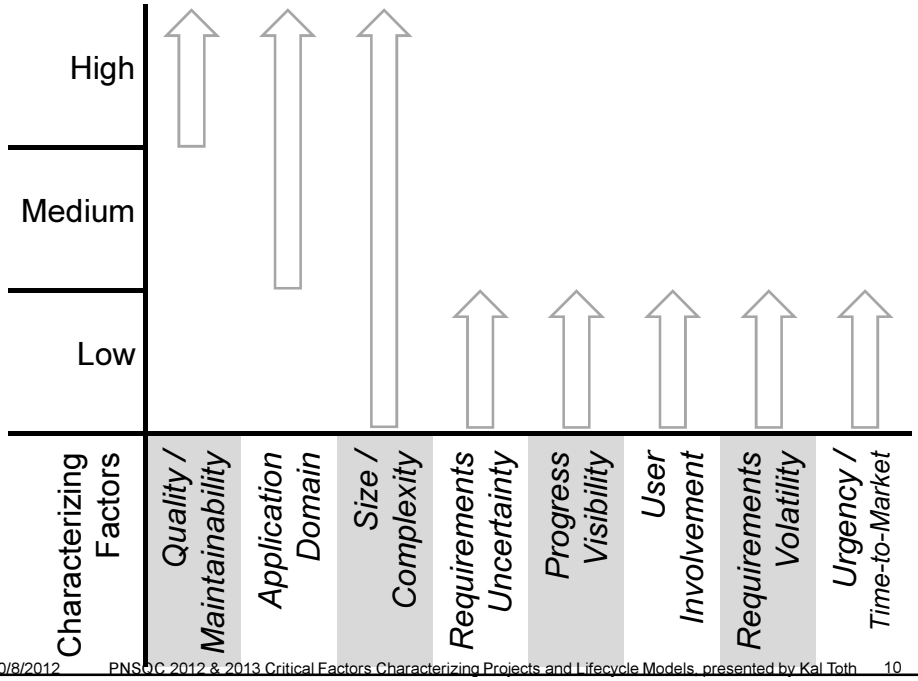
Systematic problem solving approach
Customer/user feedback can be too late
Not very adaptive to needs and market

PNSQC 2012 & 2013 Critical Factors Characterizing Projects and Lifecycle Models, presented by Kal Toth 8

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

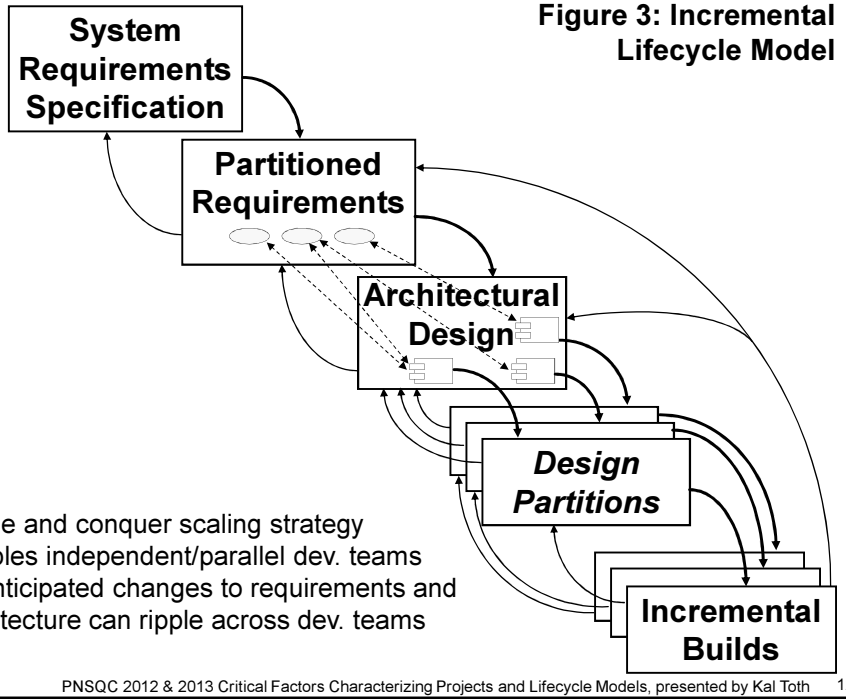
Figure 2: Waterfall Lifecycle Model Characterized



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 control project than waterfall Harder to write firm contracts and subcontracts

Figure 3: Incremental Lifecycle Model



Divide and conquer scaling strategy
Enables independent/parallel dev. teams
Unanticipated changes to requirements and architecture can ripple across dev. teams

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

Figure 4: Incremental Lifecycle Model Characterized

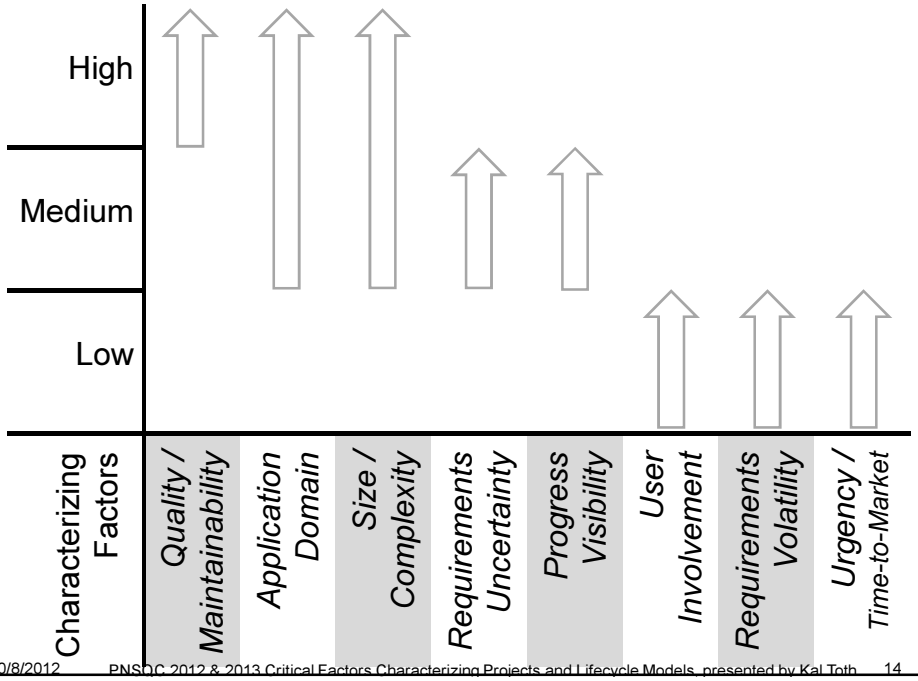
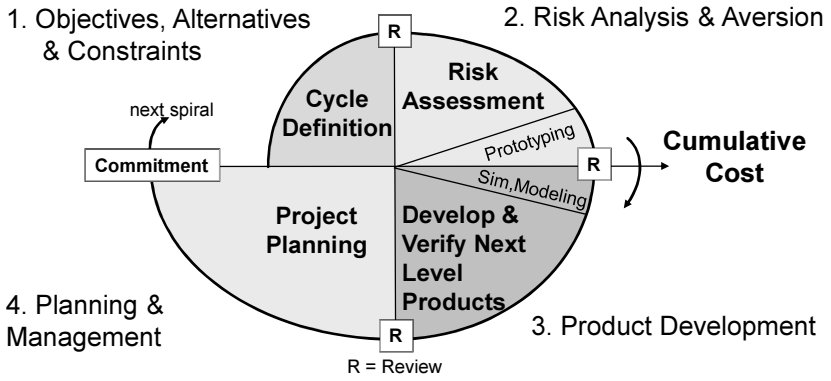


Figure 5: Boehm's Spiral Model [11]

Adapted from [7] (considerably simplified)



Early iterations consolidate requirements & tech problem areas [RUP-like]
Later iterations more waterfall-like and incremental (concurrent indep. dev.)
PM/contracting harder -> concurrency, risk assessment, and lessons learned

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 6: Spiral Lifecycle Model Characterized

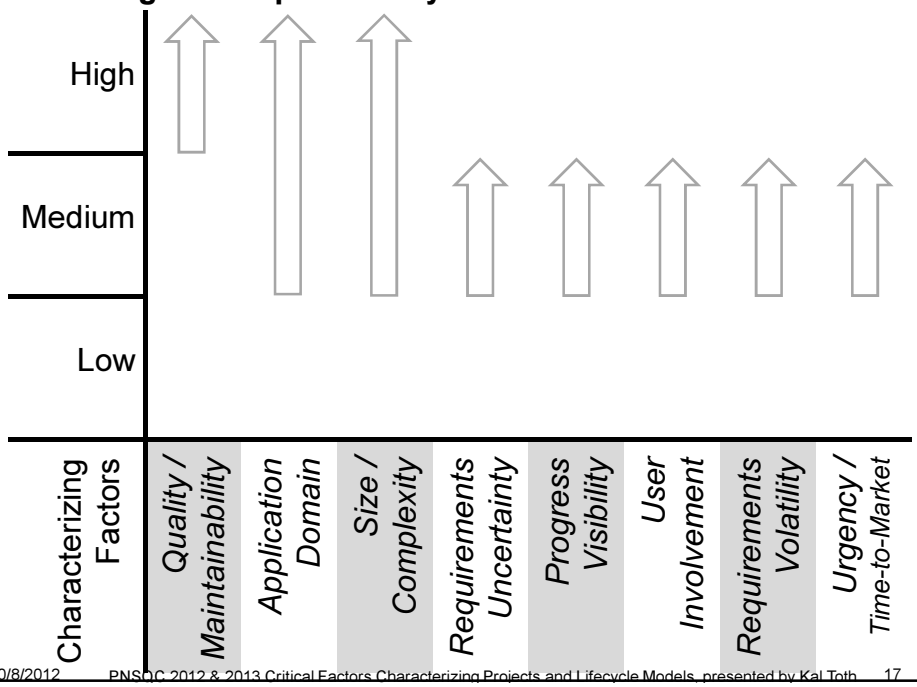
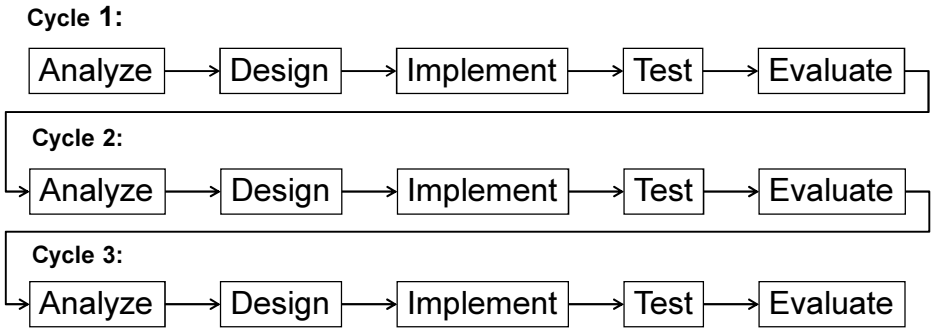


Figure 7: Evolutionary Lifecycle Model

Exploratory prototyping for solving hard technical problems & uncertainties
Work products not necessarily of deliverable quality



Focuses stakeholders on needs, hard problems, and project feasibility
Can avoid / defer time doing thorough design, reviews and testing
Managers and customers may assume prototypes are of deliverable quality

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

Figure 8: Evolutionary Lifecycle Model Characterized

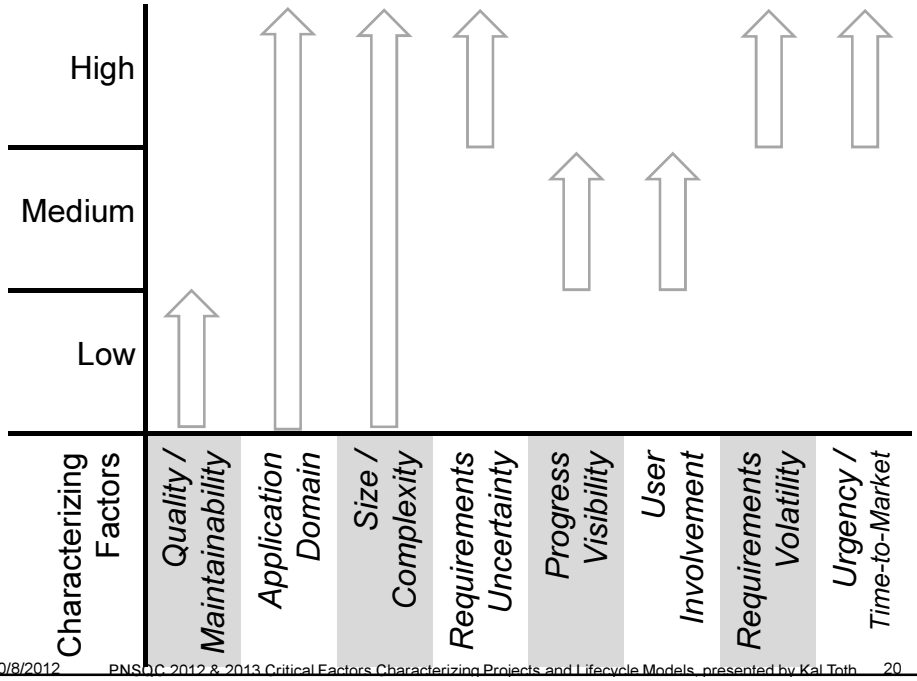
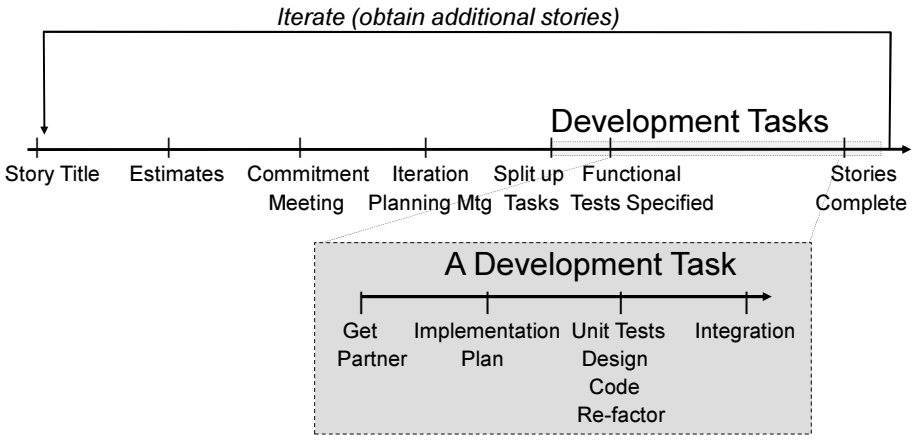


Figure 9: The Life of an Agile Story

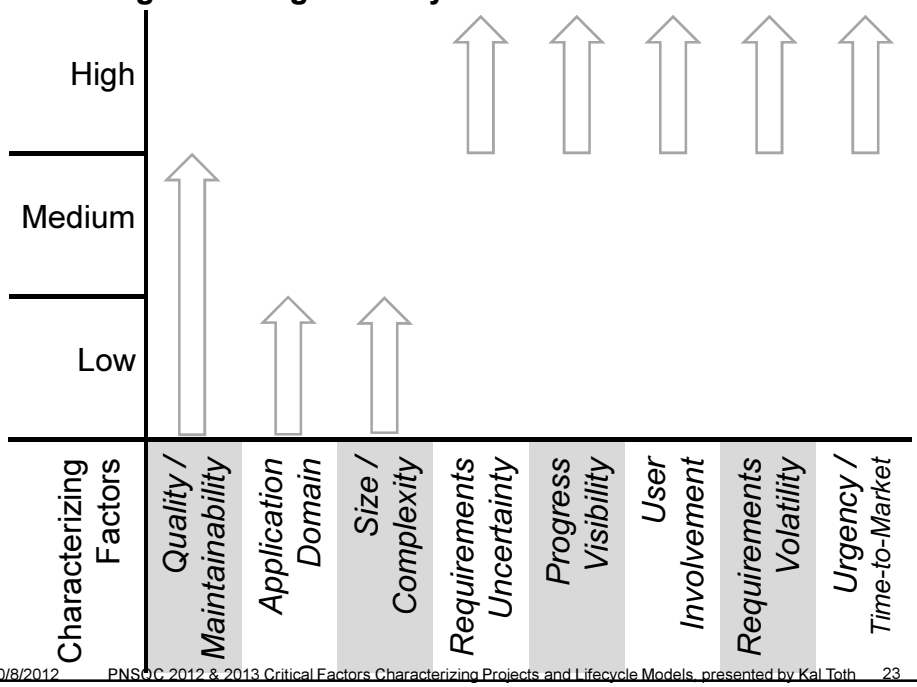


Adaptive to change; close customer involvement; ongoing refactoring
Lightweight documentation may escalate risks for mission-critical projects
Not much published data on how well Agile processes scale

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 10: Agile Lifecycle Model Characterized



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
Requirements Uncertainty	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)

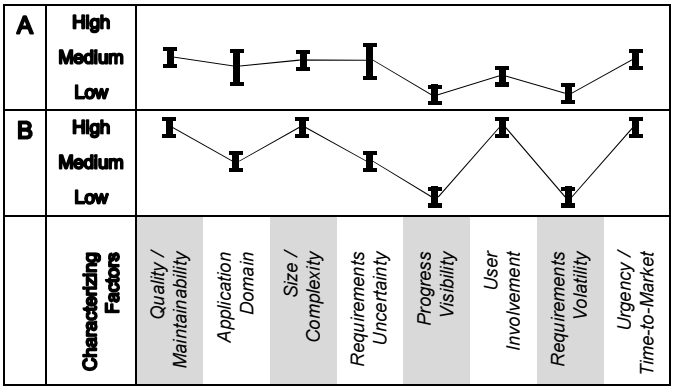
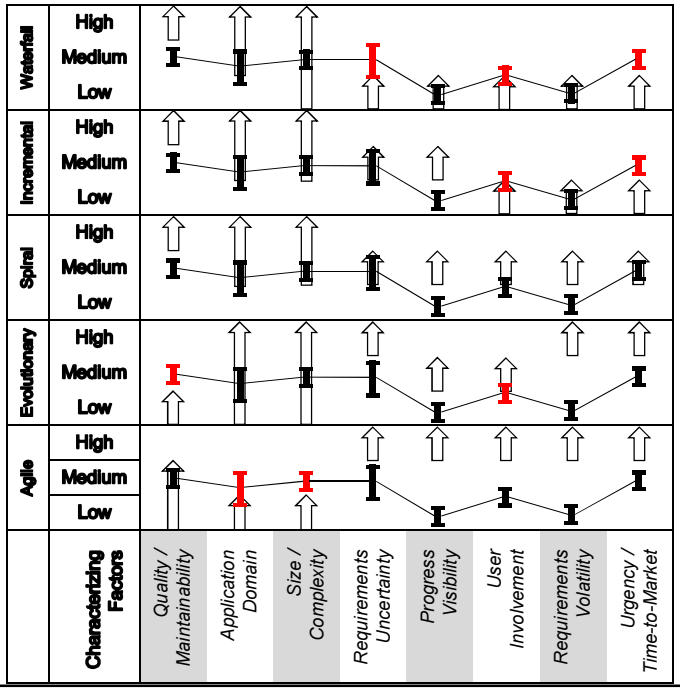
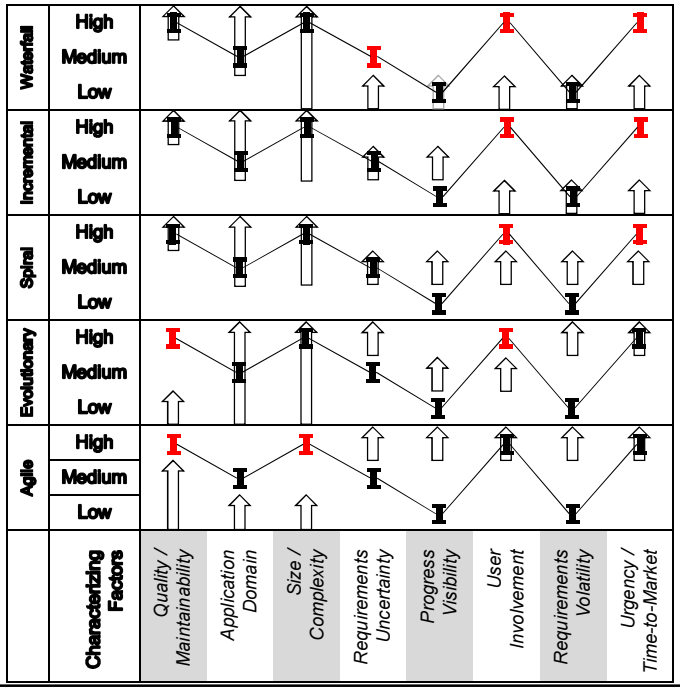


Figure 11: Illustrating Project A Matched to Lifecycles



PNSQC 2012 & 2013 Critical Factors Characterizing Projects and Lifecycle Models, presented by Kal Toth

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



Summary

Goal: explored possibilities – not reams of data

Suggested:

- 8 critical factors for characterizing lifecycles & projects
- Process for characterizing lifecycles & projects:

Likely the 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]

Questions?

Welcome constructive criticism and validation

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