Architecture Centric Design Method

Anthony J. Lattanze
Carnegie Mellon University
lattanze@cs.cmu.edu
412.268.4736

Agenda

- Specifically, we will discuss
  - Motivation for the ACDM and the current version of ACDM
  - industrial experiences with the method
  - changes to ACDM
  - the future

How "Buildings" are Built

- Architects are hired very early in the conceptual phase of construction.
  - They provide models of the thing they plan to build to potential stakeholders.
- From the architectural models
  - detailed designs are developed
  - estimations, work breakdown structures, and construction schedules are derived
  - work force allocation is determined
- *The ACDM provides guidance for practitioners to help broaden the role of architectural design artifacts.*

ACDM Philosophy

- As in the building industry, architectural designs for software intensive systems should be more than pretty pictures and models.
- The ACDM provides a framework to broaden the use architecture design to
  - help define and refine requirements
  - help set and manage product expectations
  - quickly identify and overcome unknowns
  - help to define/reallocate team structures
  - aid in project estimation, scheduling, project tracking and oversight
  - guide design, production, maintenance,… and others

ACDM Origins

- Began in 1999 within the Masters of Software Engineering software studio project course as a complete development process.
  - real world, 16 month projects, small teams (~5)
- Refined based on experiences in industry
  - by applying QAW, ATAM, and training practitioners as a member of the SAT Initiative
  - private consulting engagements to design and document architectures
  - personal professional experience as an architect
- Currently being piloted and used in industry.

How is ACDM Different?

- Today, the ACDM continues to evolve into an *architecture design method*.
  - not a complete development process
  - designed to fit with existing process frameworks not supplant or disrupt them
  - not designed to supersede or eliminate the need for detailed designs or design methods
- The ACDM is a uniquely positioned method designed to
  - help with the design of architectures
  - complement existing processes and augment them throughout the product lifecycle

Why Bother?

- Enterprise and system architecture communities don't explicitly address software architecture design.
  - software architectural design concerns are often lost in the "system" or "enterprise" abstraction
- Some architectural methods are *intervention oriented* rather than *holistic*.
  - stop, apply the method, and continue
  - are geared toward big organizations, and budgets
  - heavyweight and hard to tailor to fit existing organizations, lifecycles, and processes
  - not clear about what to do with the generated artifacts

What I Hear in Industry – 1

- How do I use these architecture methods in my organization (with our processes)?
- What do I do with a software architecture once I create it?
- So I have all these quality attribute scenarios – now what?
- How is software architecture different from EA, DoDAF, C4ISR, or system architecture? Do I need all of them? How (or can they) work together?
- When do I design my architecture?
- When am I done architecting?
- How/when do I evaluate my architecture?

What I Hear in Industry – 2

- How do I use the output of an architecture evaluation?
- How is architecture different from the UML designs I have now?
- Can I design an architecture and still be agile?
- Can I use architecture concepts, methods, and techniques in my small teams and small projects?
- Can I design an architecture if I develop my products iteratively?
- How much requirements work do I need to do before I begin architecture design?
- How and when do I get architecture requirements?

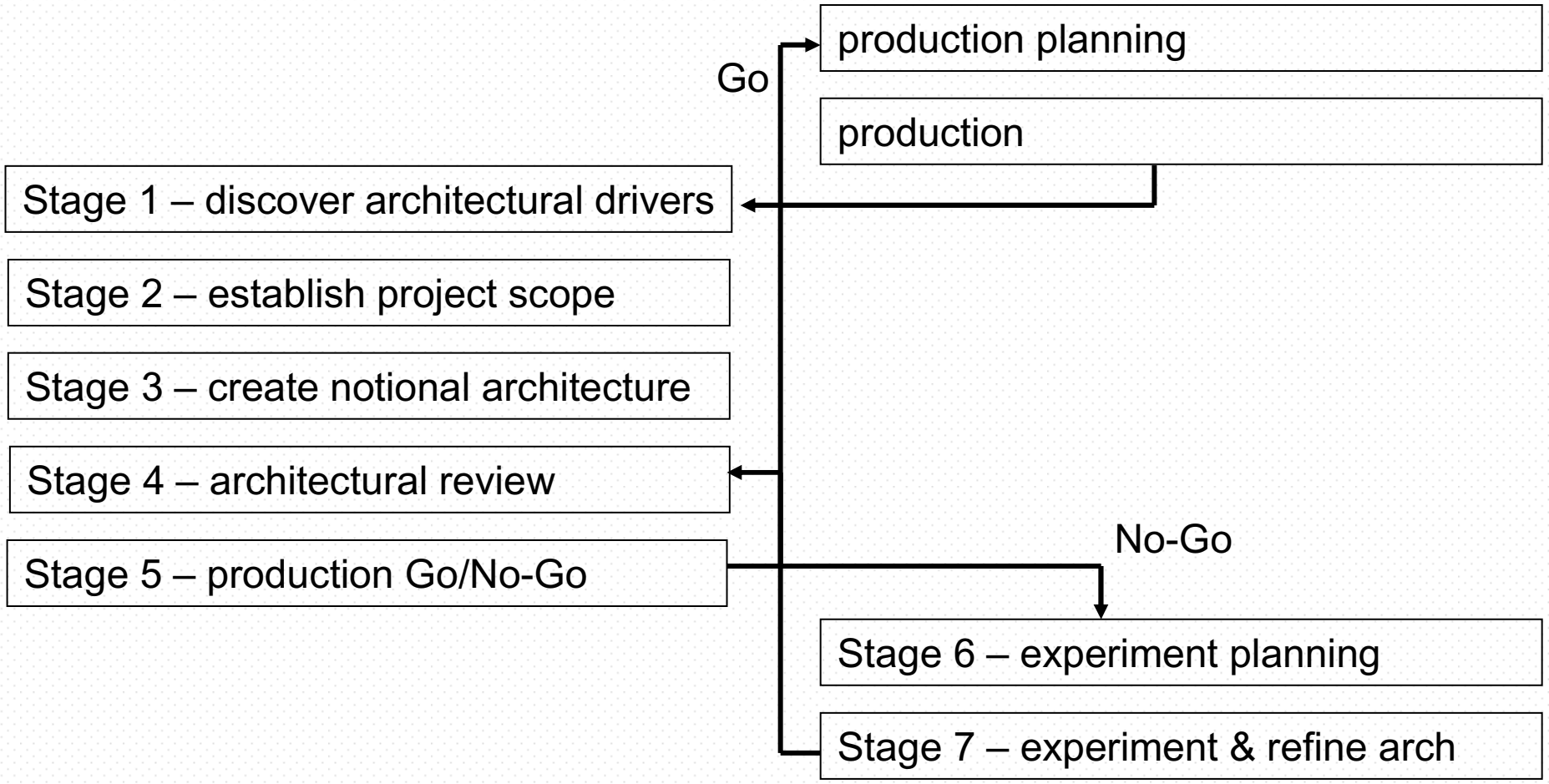
ACDM Defined

ACDM is a scaleable method for designing the architecture of a software intensive system with a product focus that uses the architecture to complement organizational processes and implementation activities.

ACDM Overview – 1

- ACDM is an iterative architecture design method
  - *prescribes* iteratively designing, evaluating, and refining the architecture until it is deemed fit for purpose
  - *supports* iteration in the production of the elements/systems/products
- ACDM has 7 fundamental stages.

ACDM Overview – 2



ACDM Overview – 3

- ACDM defines 6 roles and responsibilities.
  - The roles serve as "placeholders" for essential responsibilities for development team members.
  - Roles are strongly recommended but should serve as a starting position – please apply common sense.
  - Roles may be shared, filled by a single person, or by entire teams and/or separate organizations.

ACDM Roles – 1

- Chief Architect
  - Coordinates creation of the notional architecture and continual refinements of the architecture.
  - Captures and documents architectural risks and tradeoffs.
  - Responsible for architectural representations: Coordinates team creation and maintenance of architecture documentation.
  - Responsible for configuration control of the architecture (representations and structures).

ACDM Roles – 2

- Managing Engineer
  - Coordination of the overall development effort.
  - Establishing project estimates.
  - Establishing project budgets.
  - Coordinates the creation and documentation of the project's plans and schedules.
  - Conduct project tracking and oversight.
  - Primary customer/client interface.

ACDM Roles – 3

- Chief Scientist
  - Coordinates architectural reviews and questioning.
  - Coordinates the creation, execution, and documentation of experiments.
  - Documents the element and architectural framework test plans (if necessary).
  - Works with Requirements Engineer on test planning and execution.
  - Works with Managing Engineer to create construction estimates, plans, and schedules.
  - Oversees the work of downstream engineers (designers and/or developers).

ACDM Roles – 4

- Requirements Engineer
  - Coordinates and leads the gathering and documenting of the architectural drivers:
    - Functional requirements use cases.
    - Quality attribute discovery and documentation.
    - Programmatic and technical constraints.
  - Coordinates creation of the Statement of Work (SOW).
  - Serves as customer liaison.
  - Responsible for quality assurance plan
  - Responsible test planning and execution (e.g. documenting plan and results).

ACDM Roles – 5

- Support Engineer
  - Coordinates and maintains support tools and environments for.
    - software development and experiments
    - CM, defect tracking, and COTS products
    - license tracking and maintenance
    - Operating systems and web presence
  - Ensures that the ACDM is followed, records deviations, documents changes to the ACDM as required.

ACDM Roles – 6

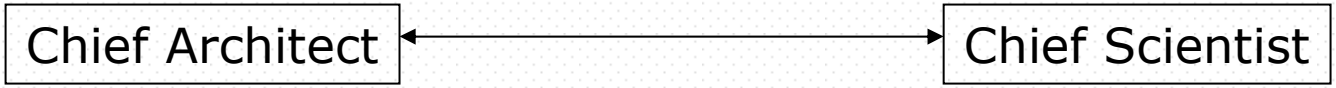
- Software Engineer
  - Focuses on detailed design and/or coding of the architectural elements of the system.
  - May conduct element or system integration tests per the established test plans.
  - In small teams, all team members will be software engineers.
  - In larger organizations, you may have separate software engineering teams, departments, and organizations.

Deliberate Separation of Powers and Concerns

- Examples:

- view of requirements is black-box
- what is needed by stakeholders

- view of requirements is clear-box
- implementation



- creates notional architecture
- refines notional architecture

- coordinates review of the architecture
- plans experiments

- programmatic concerns

- technical concerns

Discover The Architectural Drivers – 1

- Meet with client stakeholders to discover and document architectural drivers.
- The method *does NOT* prescribe using Quality Attribute Workshop (QAW).
  - While QAW could be used, this step blends some of the social techniques that are used in QAW and ATAM.
  - Not using QAW in totality allows ACDM to scale from small teams, to bigger organizations.
  - ACDM provides guidance for teams to elicit and capture ALL of the architectural drivers.

A Distinction from QAW

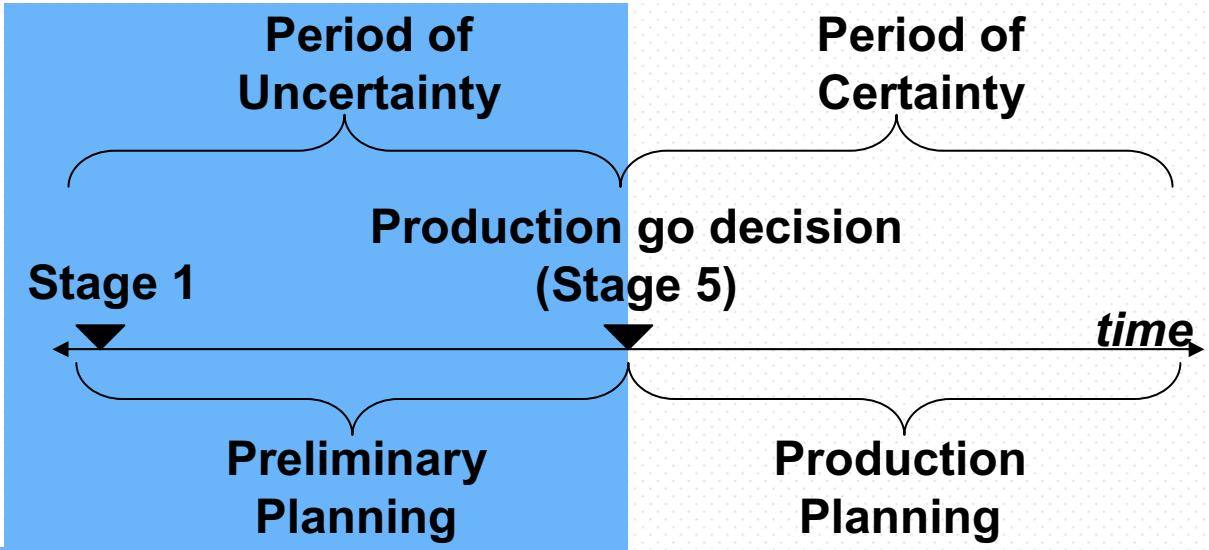
- The ACDM focuses more broadly on capturing all of the *architectural drivers* not just quality attributes.
  - Functional Requirements – *recommends* documenting the key functional requirements as use cases
  - Constraints – *prescribes* capturing and separating technical AND business constraints
  - Quality Attributes – *prescribes* capturing and building the utility tree immediately, but mapping from business driver, to quality attribute,... to quality attribute scenario

Establish Project Scope

- Team refines and distills architectural drivers into an *Architectural Drivers Specification*.
  - This is where the architectural drivers are consolidated and a utility-tree-like† artifact is created.
  - Create an initial Statement of Work.
  - Create a preliminary project plan.
    - Does not include complete production estimates and schedules.
    - Preliminary project plan estimates how long it will take the team to create a *stable* architecture.
  - ACDM provides guidance for performing early estimates during the *period of uncertainty*.

† I call this artifact a "quality attribute/business goal characterization table" but this may change…

ACDM and Preliminary Planning



Focuses on:

- how long it will take to discover the architectural drivers
- create the notional architecture
- how many experiments
- refining the architecture for production

Focuses on:

- mapping architectural elements to tasks, schedules, and personnel
- how long it will take to design, construct, and test each element
- how long it will take to integrate the elements of the architecture into a system

Create Notional Architecture – 1

- The development team uses the architectural drivers specification as a basis to create the *notional architecture.*
- The notional architecture represents the first attempt to design the system.
  - initial partitioning of the system
  - developed quickly
  - refined iteratively in subsequent stages

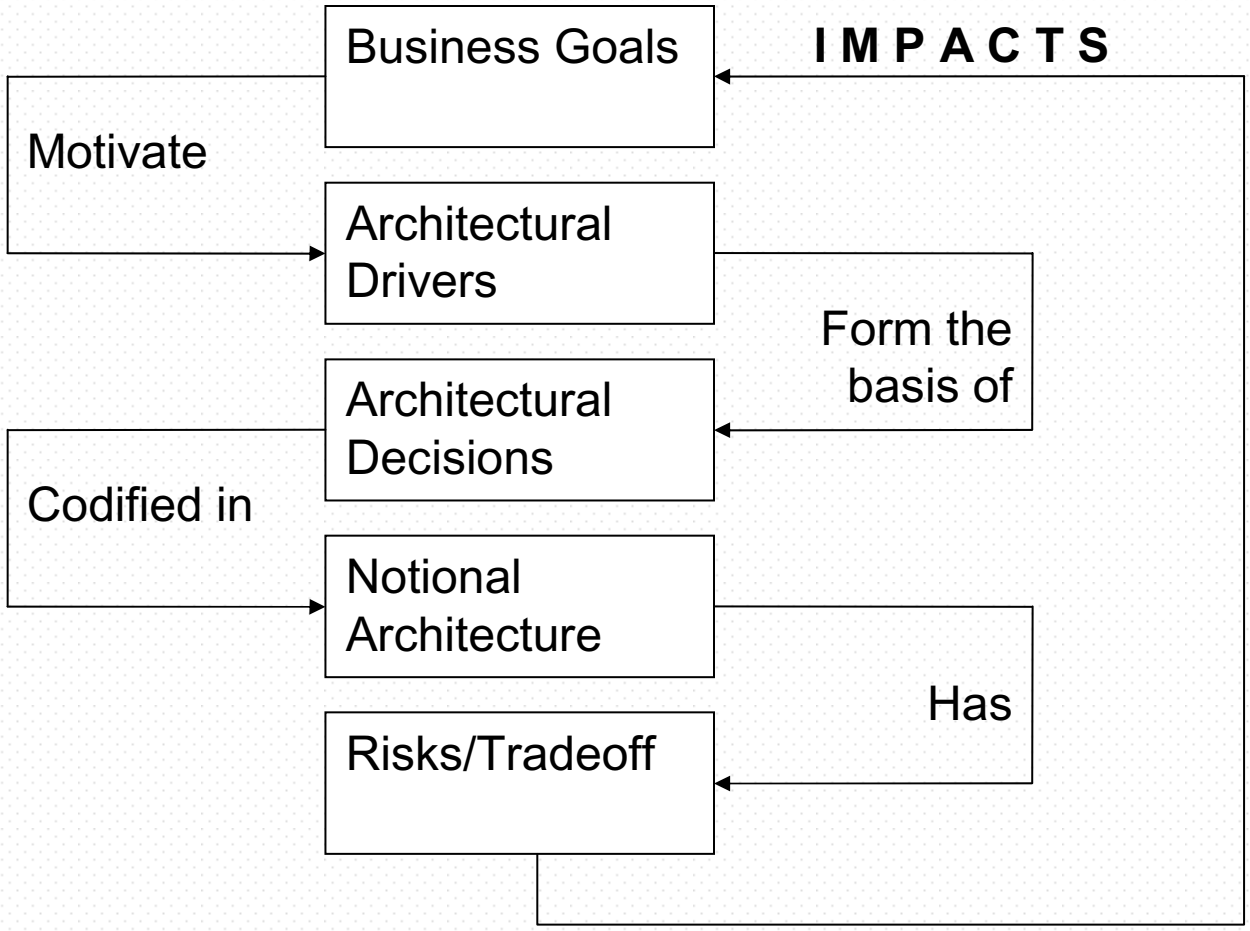
Create Notional Architecture – 2

- Create initial representations of the structures that will comprise the system.
  - context
  - physical perspectives
  - code or static perspectives (includes data models)
  - run-time perspectives
  - element interfaces
  - management perspective (important later)
- Not all representations are required for the notional architecture.
- ACDM provides guidance for how to create the architecture.

Review Architecture

- ATAM is NOT prescribed, but this stage is similar to ATAM step 6.
  - The utility tree already exists, no need to create it.
  - Initial reviews are done within the design team (phase 1ish); after refinement, subsequent reviews are done with broader stakeholder involvement (phase 2ish).
  - Only risks and tradeoffs are identified: I have found sensitivity-points to be too subtle to identify and quantify and not be as valuable.
  - Use cases also play a role in the review – I have found this to be an interesting exercise.

Risk/Tradeoff Pedigree†



† This is maintained in part vis-à-vis the "quality attribute/business goal characterization table"

Production Go/No-Go Decision

- The team decides whether the development team is ready to begin production or if they need to refine the architecture.
  - Discovered risks should be evaluated for severity and likelihood of coming to fruition.
  - This need not be an all-or-nothing decision.
    - Perhaps the overall structure is sound, but more refinement is needed on particular elements of the system.
- ACDM provides Go/No-Go guidelines.

Plan Experiments

- The term *experiment* is used deliberately.
  - In practice, prototypes are often unplanned, and ad hoc prototypes transmogrify into products.
  - To avoid this, ACDM prescribes that
    - teams design experiments specifically to mitigate risks that were discovered during the review (stage 4)
    - experiments are targeted, planned, technical prototypes that are for the purpose of refining the architecture exploring the architectural drivers
- ACDM provides an experiment template.

Example Experiment Template

Experiment Plan Title	
Element	**Content Description**
Experiment ID	This is the title or something that uniquely identifies this experiment.
Responsible Engineer	This is the development team member that is responsible for this experiment.
Purpose	Describe the reason for conducting the experiment. It is strongly advised that the author explain how the experiment will be used to refine the architecture.
Expected Outcomes	Describe what the responsible engineer expects the outcome or outcomes will be of the experiment.
Resources Required	List the resources required that include: compute resources (software/hardware), people, time, money, and so forth.
Artifacts	These are the artifacts that will be created as a result of executing the experiment such as software, documentation, and so forth.
Experiment Description	Describe the experiment. This includes software that will be written, research to be performed, studies to be carried out, information that will be collected and how it will be collected and so forth.
Duration	The amount of time that it is expected to complete the experiment. Must include an explicit start date, stop date, and milestones as applicable. This should be a mini-schedule of events that can be tracked by the Chief Scientist. The Managing Engineer can roll up the durations and dependencies for all the experiments.
Results and recommendations	The responsible engineer must document the results of the experiment. Describe deviations from the expected outcomes and reasons for the deviations. Discuss and deviations from the planned experiment description. Describe recommendations as a result of conducting the experiment.

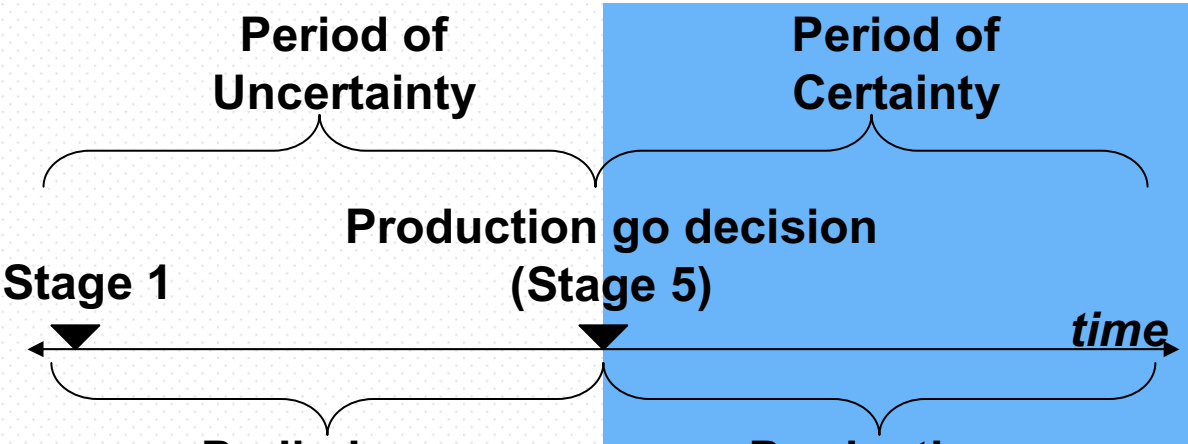
Experiment and Refine Architecture

- Update preliminary plan based on experiments – *experiments take time!*
- Team executes the experiments per the experiment plans.
- Results of experiments are recorded on the experiment template.
- Architecture is updated/refined per the results of the experiments.

Production Planning

- The ACDM provides guidelines for creating production plans.
  - production plans, test plans, and schedules are derived directly from the architecture
  - typically derived from the management perspective, but the exact viewtype used will vary
  - an estimation technique is provided based on the Wideband Delphi technique
    - ACDM uses a tailored version called ***Element-Wise Wideband Delphi Estimation***.
    - Production estimates are based on estimating individual architectural elements

ACDM and Production Planning



Focuses on:

- how long it will take to discover the architectural drivers
- create the notional architecture
- how many experiments
- refining the architecture for production

Focuses on:

- mapping architectural elements to tasks, schedules, and personnel
- how long it will take to design, construct, and test each element
- how long it will take to integrate the elements of the architecture into a system

Production Guidance

- In the ACDM context, production includes detailed element design, coding, integration, and testing.
  - production may be big-bang, or iterative
- ACDM provides guidance for
  - transitioning and mapping architecture design to detailed design elements
  - performing tracking and oversight using *earned value* derived from the elements of the architecture
  - testing using the architecture and a blueprint

Tracking and Oversight Example – 1

- Once the schedule has been derived, the total project time is calculated - add up the sum of all the task times derived from the elements that comprise the architecture.

The total project duration = Σ (task duration*T*) for all tasks *T*

The earned value for task *T* = (task duration*T)* **/** (total project duration)

Percentage Complete at time *t* = (summation of earned value at time *t*) **/** (total project duration)

Tracking and Oversight Example– 2

- Performance Index is a measure of how well the team is performing with respect to the schedule - A value close to 1 indicates that the team is performing very well with respect to the schedule.

Performance Index = (total project duration) **/** (summation of earned value at time *t*)

Tracking and Oversight Example – 3

- Schedule variance is the difference between the actual earned value at time *t* and the planned earned value at time *t*. If schedule variance is
  - zero the team is not deviating from the planned schedule.
  - negative the team is behind schedule
  - positive the team is ahead of schedule

Schedule Variance= (actual earned value at time t) - (planned earned value at time t)

Industry Experience So Far – 1

- What I am hearing so far…
  - "I always heard about architecture, I even claimed to have designed one, now I know what to do with it once I have one [an architecture]."
  - "Experiments are an extremely valuable concept for knowing what to prototype and focusing those efforts."
  - "ACDM may be the best description of what a software architect does in the course of doing their job."
  - "ACDM is the first plausible marriage of architecture and iterative development."
  - "We found the roles to be extremely beneficial."

Industry Experience So Far – 2

- Still some rough edges:
  - Requires software architectural expertise to fully exploit the method.
  - Architecture can be a hard sell in a product focused organization.
  - Need more information about how to
    - how to use ACDM with legacy systems.
    - how to scale up ACDM
    - how, where, and when the detailed requirements are written down.
  - More templates and guidance in specific techniques please.
  - The roles are helpful, more information please.

Status of the ACDM

- Version 3.0 is currently being piloted, although it has not been published yet.
  - version 2 is currently published
  - currently collecting and consolidating latest round of lessons learned
  - planning an update (version 3) to the existing technical report this summer based on lessons learned thus far
  - this will be the last technical report for the method
  - a book proposal is currently in the works

Planned Changes to the Method – 1

- More elaboration upon architecture drivers
  - techniques for gathering them
  - documentation strategies
  - taxonomies for interacting with stakeholders for elicitation
  - how and when to refine architecture drivers into a detailed requirements specification
- More elaboration upon architectural design
  - guidance for decomposition from system to element interface
  - guidance for what, how, and when to document

Planned Changes to the Method – 2

- More elaboration upon how to conduct useful architecture reviews
  - scenario base questioning taxonomy for guiding the review of the architecture based on quality attribute scenarios AND functional use cases
  - hints on decisions that are likely to result in common tradeoffs between quality attributes
  - guidance for conducting architectural reviews with or without stakeholders present

Planned Changes to the Method – 3

- Addressing non-green field development
  - guidance for performing architectural forensics(?)
  - guidance for element as-built (black-box) element specification(?)
  - guidance for avoiding, detecting, and resolving architectural mismatch
- More templates and guidance
  - architecture drivers specification
  - quality attribute scenarios
  - project planning, estimation, tracking and oversight using the architecture
  - architecture review

Planned Changes to the Method – 4

- The method will no longer addresses production concerns prescriptively, but will provide extensive guidance for
  - using ACDM to complement other development processes such as RUP, SCRUM, and XP.
  - what SW, SE, and SW/SE CM PAs of the CMMI are addressed by using ACDM.

References – 1

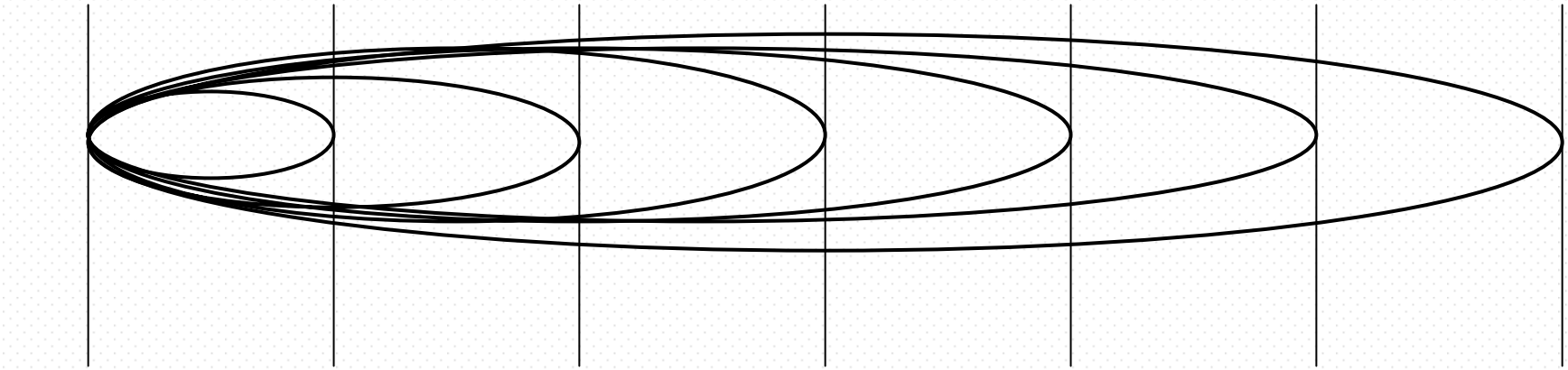
- Lattanze A., *"The Architecture Centric Development Method,"* CMU-ISRI-05-103, School of Computer Science, Carnegie Mellon University, 2005
  http://reports-archive.adm.cs.cmu.edu/anon/isri2005/CMU-ISRI-05-103.pdf
- Clements P., Kazman R., Klein M., *"Evaluating Software Architecture: Methods and Case Studies,"* Reading, MA: Addison-Wesley, 2002
- Bass L., Clements P., Kazman R.; "*Software Architecture in Practice - 2nd Edition*," Reading MA: Addison-Wesley, 2003
- Humphrey, W., *"Introduction to the Team Software Process,"* Reading MA: Addison-Wesley, 1999
- Humphrey, W., *"A Discipline for Software Engineering,"* Reading MA: Addison-Wesley, 1995
- Humphrey, W., *"Managing the Software Process,"* Reading MA: Addison-Wesley, 1989
- Boehm, B., "*Software Engineering Economics,*" Englewood Cliffs NJ. Prentice Hall, 1981
- Singpurwalla, N., Wilson, S., "*Statistical Methods in Software Engineering,*" New York. Springer, 1999

References – 2

- Beck, K., *"Extreme Programming Explained: Embrace Change,"* Addison-Wesley, 1999
- Pressman, R., "Software Engineering: A Practitioners Approach 6th ed.," Singapore. McGraw-Hill, 2005

Implicit Iteration

Period of Uncertainty: Iteration is fast and furious between each stage not just review and refine



Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7
Arch Drivers	Define Scope	Notional Arch	Review	Go/No-Go	Experiments	Refine Arch

Production, that-a-way

Carnegie Mellon

Questions/Comments?

Presenter: Anthony J. Lattanze