Outline

- Introduction to System Development Methodology
- Contrasting Plan-Driven & Agile Project Approaches
- Selection Considerations
System Development Methodology

- Approach to developing software-based systems
- Provides processes and, optionally, procedures
- May provide engineering tools and techniques
- A defined process
  - Not necessarily formal or documented
  - Should be for consistency and conformance!
SDM vs. PMBOK

- **SDM** = System Development Methodology
- **PMBOK®** = Project Management Book of Knowledge
- PMBOK focuses on managing a “generic” project
- SDM focuses on producing a software product
- PMBOK - who does what and when
- SDM - what to produce and how
The Agile Manifesto

- **Individuals and interactions** over processes and tools.
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan
The Plan-Driven “Manifesto”

- It is better to know than not
  - How much is this project going to cost?
  - How long will it take?
  - Who will need to be involved and when?
  - What can I expect and when can I expect it?
Plan-driven vs. Agile

- Selecting an approach is situational depending upon organizational needs, teams skills and abilities, problem complexity, etc.
  - Project approach is used to mitigate and control risk
- Plan-driven software development uses structure to control risk
- Agile software development uses flexibility to control risk
- Personal observation:
  - Management teams that work well with Plan-driven approaches also tend to work well with Agile approaches
  - However, management teams that lack the ability to work well with Plan-driven approaches may lack the discipline required of Agile
Plan-driven vs. Agile Situational Approach

- Division landing on the beach
- Brain surgery
- Major motion picture
- Round the world cruise

- Black ops behind enemy lines
- Emergency room
- Reality TV program
- Weekend in Las Vegas
Plan-driven vs. Agile SDMs

- There is a continuum between these approaches with “extreme” ends – very agile or very controlled.
- At the middle, there are many subtle differences allowing the “right sizing” of approaches.
- There are some generalities that can be made
  - Plan-driven emphasizes formal communications and control – it attempts to be more predictive in nature
  - Agile emphasizes continual informal communications and an ability to react to changes and uncertainty – it is more adaptive in nature
Plan-driven Approaches

- Artifact and milestone driven
- Documentation is formal and structured
- Emphasis on “up-front” planning
- Higher degree of project ceremony
  - Team meetings, formal project reviews, documented status reports with great deal of details
- Structured communications
- Heavy project governance and oversight
- Formal change control with Change Control Boards
- Project stage gates requiring formal approval in order to proceed
- Well defined project roles with separation of duties
Agile Approaches

- Code-based deliverable driven (vs. documents)
- People oriented with informal but constant communications
- Emphasis on “on-going” planning
- Lower project ceremony
  - Limited team meetings, brief project reviews, documented status reports with low details
  - Limited or no formal change control within iterations
- Limited project stage gates requiring formal approval in order to proceed
Plan-driven Approaches

- Do not confuse plan-driven approaches of today with those of the past.
  - They have learned to be adaptive in their own way incorporating ideas such as time-boxing, iterative development, emergent designs, and multiple releases.

- The “control” part is now mostly focused on communications, project governance, and resource management.
Plan-Driven Methodologies

- Traditional project management
  - PMBOK
  - Software development from the 1970’s and before
- Unified Process
  - IBM/Rational’s heavy implementation (vs. newer agile implementations)
Example - Unified Process

- A very robust methodology with well defined systems development lifecycle
- Provides specific artifacts
  - Use cases, scenario diagrams, architecture models
  - Not all required - situational
- Defines how each process and artifact evolves from previous items
  - Very iterative in nature
- Spans a wide section of the Agile/Plan-driven continuum allowing it to be very adaptable to different needs
Agile Methodologies

Examples include:

- Extreme Programming
- SCRUM
- Feature Driven Development
- Lean Development
- Crystal
Example - Extreme Programming

- Defined Rules and Practices including:
  - Documented stories (vs. use cases)
  - Design for no more than is needed
  - Pair programming
  - Write tests before coding; all code must have unit tests
  - Co-location of developers and subject matter experts
  - SME is always available
  - Daily stand-up meeting
  - Many small releases (iterations with re-factoring)
  - Continuous integration
  - No overtime
Feudin’ Methodologies

- There are those who like to argue that one type is better than the other in all cases.
- Agile bigots paint Plan-driven as inflexible, slow, and unable to respond to changing conditions.
- Plan-driven bigots paint Agile as chaotic, unmanageable, and un-controlled.
- When the methodologies are properly implemented, these arguments are wrong for both.
The Right Tool

- Selecting a project methodology should depend upon the situation
- A more predictive approach works better under some circumstances, being more adaptive works better under others
- There is no hard and fast rule as to which is better in each situation
- Selecting components of each, define the process that works best for your situation
A Word About Iterations

A single iteration includes:

- Analysis: what are we trying to accomplish
- Design: how will we accomplish
- Construction: building the solution
- Confirmation: test then deploy or repeat another cycle

Not a “waterfall” progression

- The steps may overlap and mix in a variety of ways however, all steps are addressed

A plan-driven approach will try to determine how many iterations are required in advance with a specified end-date

An adaptive approach may develop an end-date over time or elect to stop after any iteration
Factors to Consider

The following series of slides will present some of the factors which you may wish to consider when choosing between the different approaches.
Problem Domain Complexity

- Complex problem domains requiring input from many individuals tend to lend themselves to the formal communications of Plan-driven approaches.

- Problem domains understood by a limited number of individuals tend to lend themselves to the rapid communications capabilities of Agile.
Solution Complexity

- Situations in which the solution is complex may be better addressed with more “up-front” modeling and planning
  - e.g. an airline reservation system, NASA control systems
  - in environments with many technical disciplines

- Less complex solutions or where the solution is less understood may be better addressed with adaptive modeling processes
  - e.g. data entry system used by a handful of people, “green-field” development

- In both cases an architecture should emerge over time
  - planning timing and effort is relative
Experience – Problem Domain

- When team has experience with the problem domain, it may be easy to plan a solution
  - Updates or extensions to current systems
- When team is new to problem domain, use the iterative discovery process of Agile
  - R&D efforts
  - After the R&D effort, a switch to Plan-driven may be appropriate in certain circumstances
    - See “other factors” in this presentation!
Experience - Technology

- Using new technologies with which the team has little experience may benefit from an Agile approach
  - R&D
  - Iterative development while gaining experience with performance factors
  - Development using new technologies impacts a subset of the team – smaller groups ➔ Agile!
Team Process Maturity

- Teams with low process maturity may require the additional structure of a Plan-driven methodology.

- Teams with high process maturity may more easily take advantage of Agile.
  - Another benefit; project schedules can be far less detailed – targeted to deliverables instead of tasks!
Project Team Size / Location

- Large or geographically disperse teams may require the formal communication structure of Plan-driven

- Small, co-located teams may more easily utilize the informal communications style of Agile

  - Note: most Agile methodologies recommend that teams not exceed 6-8 members
Regulatory Requirements

- Heavily regulated industries requiring thorough documentation with formal approvals may need Plan-driven structure

- A note about Sarbanes-Oxley:
  - SOX does not preclude Agile approaches but some considerations or process modifications must be given to required artifacts
    - e.g.: written project request, approval to initiate project, requirement specifications, test results, post-implementation approvals, etc.
Organizational Culture

- Some organizational cultures and management teams may be more receptive of one approach over the other.

- Note: be wary of management teams who want to throw out the old “waterfall” process and jump completely onto the Agile bandwagon.
  - Potential for unrealistic expectations in ease of implementation, level of effort, etc.
  - Even with the benefits of Agile, it is not a “silver bullet” – software development is still hard work!
Concurrent of Releases

- Commercial products with multiple versions under concurrent development while performing maintenance patches require very complex controls and mature processes.
- Teams which are not extremely high in process maturity may require the structured constraints of a Plan-driven approach.
Availability of Resources

- Agile dictates easy and unfettered access to subject matters experts
- In cases where access to SME’s is limited in duration or frequency, the more formal communication style of Plan-driven may work better
Scalability & Portability

- Architecting a robust system for extreme scalability across a variety of architectures and infrastructures may be best addressed with the controls of a Plan-driven methodology
  - E.g.; a commercial transaction processing system
Creating a commercial product requiring coordination with Marketing and Sales departments, preparation of User Guides, development of training programs, and establishing production support teams may benefit by using a Plan-driven approach.
Mixin’ & Matchin’

Combining practices from both types of methodologies could be beneficial in certain situations. Possible approaches:

- Use Plan-driven for a major program or project and then use Agile for sub-projects or sub-components
- Use Agile for the project but use Plan-driven techniques for project governance and stage-gates
(Almost) Conclusion

- No single factor will dictate which type of methodology will be used

- Many factors must be considered and input gathered from many stakeholders including:
  - Business/Functional management
  - Subject matter experts
  - Development team members (analysts, developers, testers, infrastructure, etc.)
  - Customers (especially if internal to the organization; i.e. “users”)
So........

How do you know when your project is in trouble?
If you hear these comments on your Plan-driven project:

- “Oh goody! Another death march!”
- “The project schedule has my lunch break and my restroom break too close together”
- “We’ll discuss it at the Change Control Board meeting next quarter”
- “Our initial project estimate is 3,537.25 hours”
If you hear these comments on your Agile project:

- “When did we make that decision? Nobody told me!”
- “The pair programmers want a divorce.”
- “We lost our design documentation when someone used it as a Kleenex.”
Suggested Reading

Thank you!