

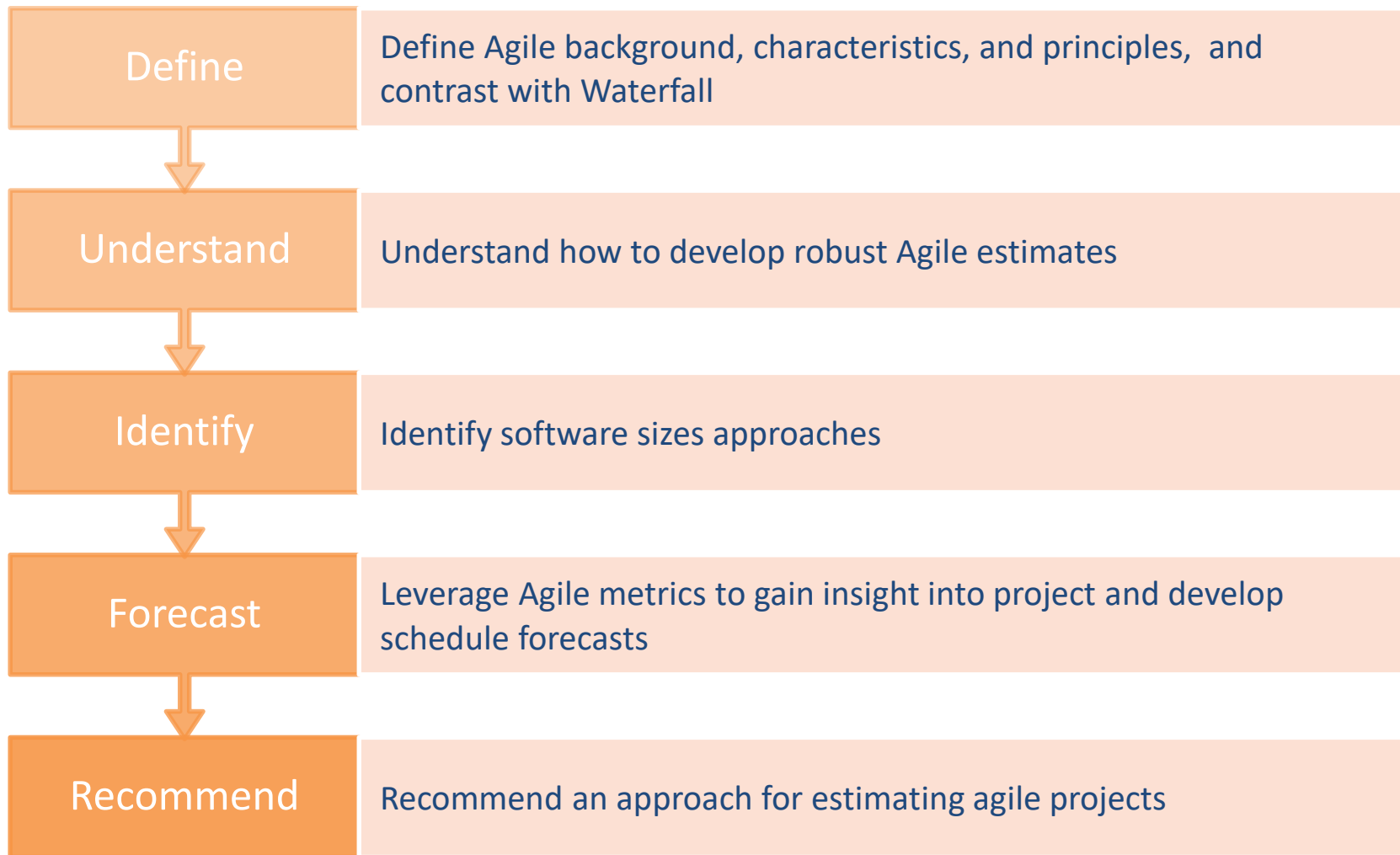
Accurate Agile Estimation

Prepared for:
2018 ICEAA Conference

Presented By:
Kevin McKeel



Agenda

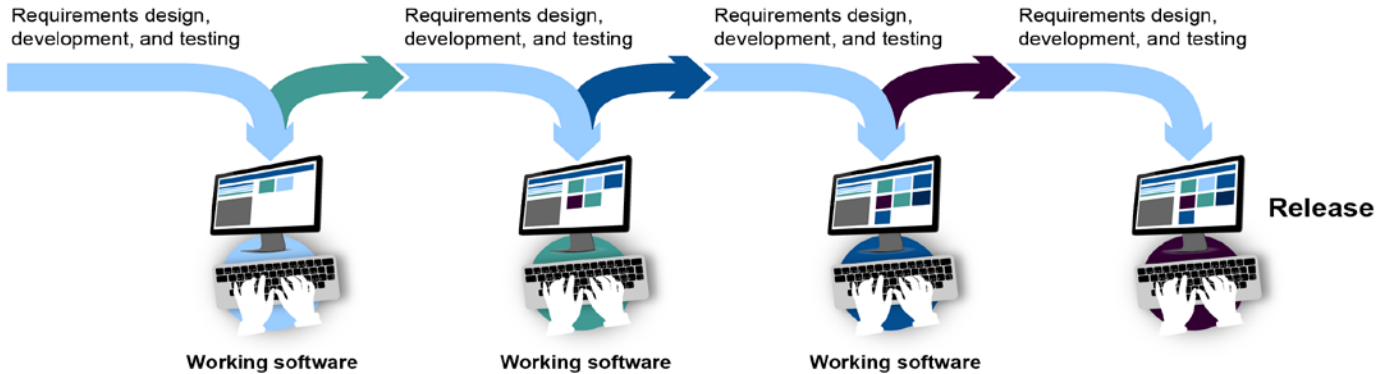


Agile Background

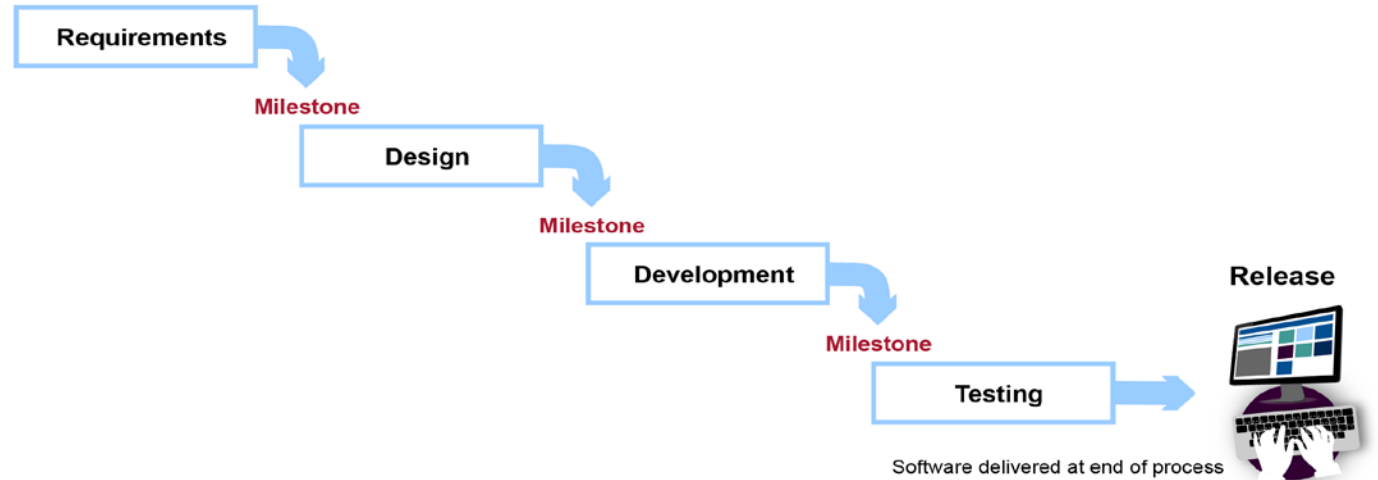
- ❁ Agile is a software development paradigm often involving time-boxed development, small multi-functional teams, evolving requirements, test-driven development, and early delivery of functional software.
- ❁ In contrast, Waterfall projects involve sequential phases, starting with system requirements and ending in a single release of software product.
- ❁ Agile Values were placed on
 - Working Software over Comprehensive documentation
 - Individuals and Interaction over Process and Tools
 - Client collaboration over Contract Negotiation
 - Responding to Change over Following a Plan
- ❁ Agile principles enable IT organizations to respond to changes in requirements and priorities through continuous collaboration between the IT project team and business experts
- ❁ Agile tenets, such as RUP, Scrum, and Extreme Programming date back to the 1980s

Agile vs Waterfall

A: Agile iterations



B: Waterfall phases



Characteristics of Agile

Vision and customer value driven

- ❖ User requirements change over time
- ❖ User requirements follow the cone of uncertainty
- ❖ Responding to change is critical

Iterative, feature driven development

- ❖ Delivery every cycle (1 – 4 weeks)
- ❖ Cycle are full lifecycle
- ❖ Cycle ends with a user review (demo)
- ❖ A release plan outlines product development

Collaborative

- ❖ Team intimacy (product owner and development staff)
- ❖ Short, shorter, shortest feedback loops
- ❖ Self organizing and self managed

12 Agile Principles

01 Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

02 Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

03 Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

04 Business people and developers must work together daily throughout the project.

05 Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

06 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

07 Working software is the primary measure of progress.

08 The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

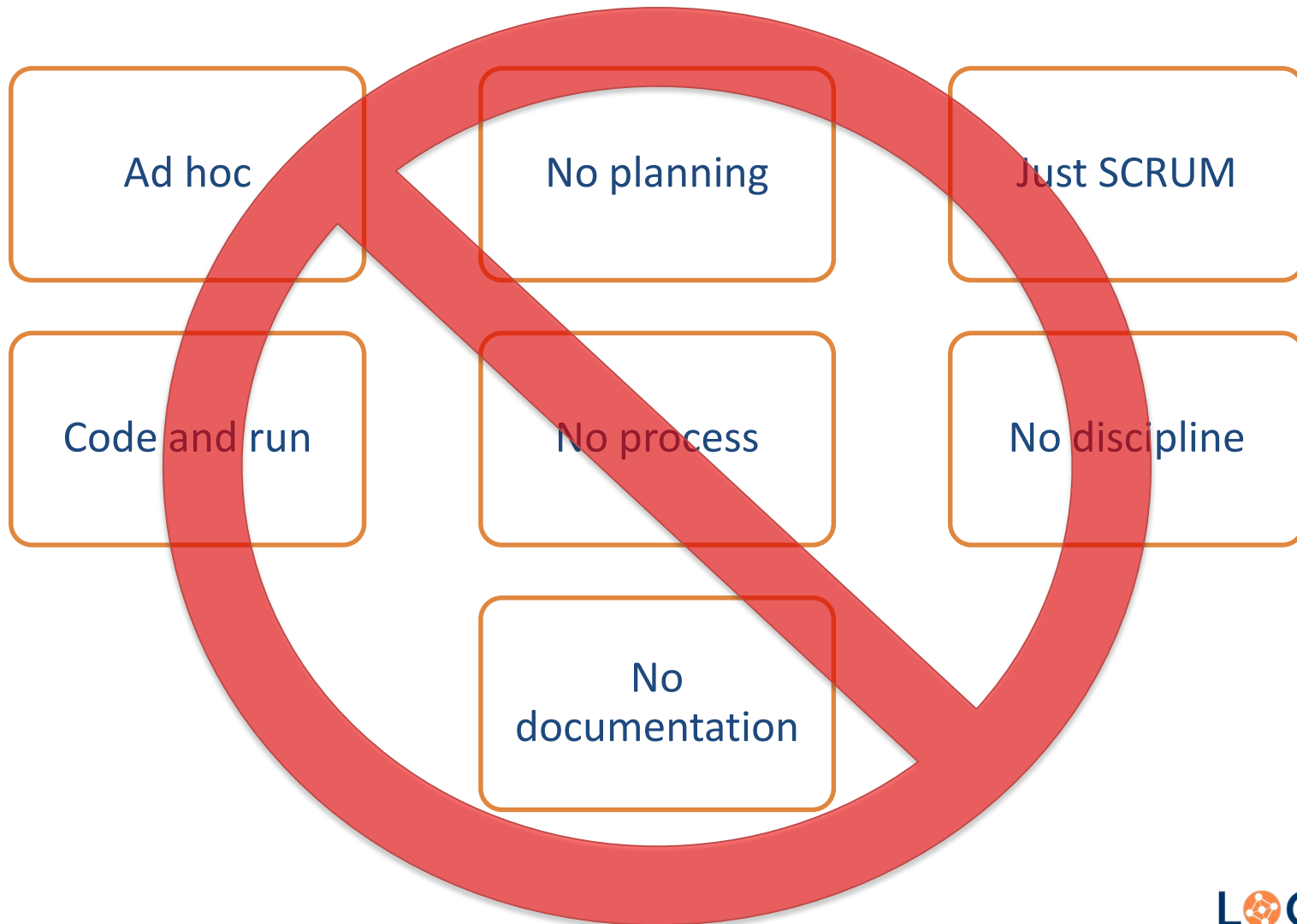
09 Continuous attention to technical excellence and good design enhances agility.

10 Simplicity – the art of maximizing the amount of work not done – is essential.

11 The best architectures, requirements, and designs emerge from self-organizing teams.

12 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

What Agile Is Not



GAO Identified Agile Federal Best Practices and Challenges

Effective Practices

- Continuously improve Agile adoption at both project and organizational level
- Enhance migration to Agile using Agile terms, such as User Stories
- Seek to identify and address impediments at project and organizational level
- Obtain stakeholder and customer feedback frequently
- Empower small, cross-functional teams
- Track progress daily and visibly

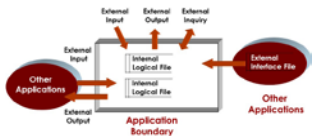
Challenges

- Teams had difficulty collaborating closely, transitioning to self-directed work and managing iterative requirements
- Staff had difficulty committing to more timely and frequent work
- Timely Adoption of new tools was difficult
- Agile Guidance was not clear
- Procurement practices may not support Agile projects
- Customers did not trust iterative solutions
- Compliance reviews were difficult to execute within an iterative time frame
- Federal reporting practices do not align with Agile
- Traditional Artifact Reviews and status tracking does not align with Agile

Basic Estimation Approach

1. Develop Functional Size

- ❖ Collect vision and scope documents
- ❖ Collect backlog data (epics, features, stories)
- ❖ Estimate via fast Function Point size
- ❖ Utilize Proxy Sizing
- ❖ Use analogy as last resort



2. Identify Process Maturity

- ❖ Identify Process Maturity
- ❖ Check alignment with Agile principles and best practices
- ❖ Confirm cross-functional team composition
- ❖ Collect and review Agile metrics



3. Evaluate Complexity

- ❖ Identify Cyber requirement
- ❖ Review System Architecture
- ❖ Determine software technology
- ❖ Review non-functional requirements and technical debt

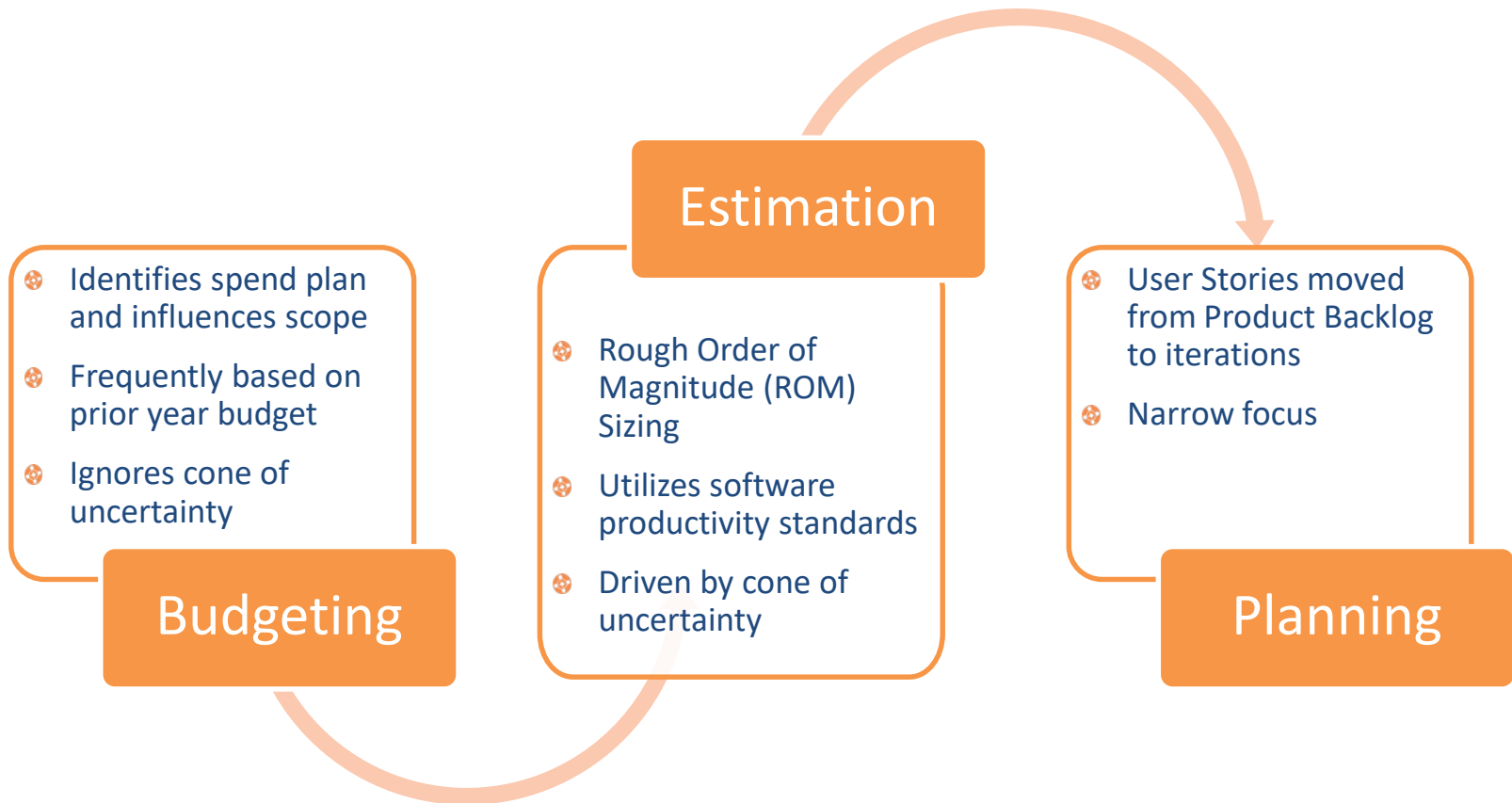


4. Apply Productivity Factors

- ❖ Determine productivity
- ❖ Develop traditional software productivity measures (FP/month)
- ❖ Evaluate schedule against capacity
- ❖ Develop crosschecks
- ❖ Apply risk



Budgeting, Estimation, and Planning



Software Sizing

🌀 Agile software sizing challenges

- User Story points are inherently difficult to utilize in an estimate due to lack of consistency across Agile teams, and poor metrics collection
- User stories are frequently unknown beyond next few iterations
- Not all Agile teams use Story Points

🌀 Sizing Approaches

- Apply function point techniques to develop size estimates for each user story, feature and epic
- Use proxy sizing when appropriate

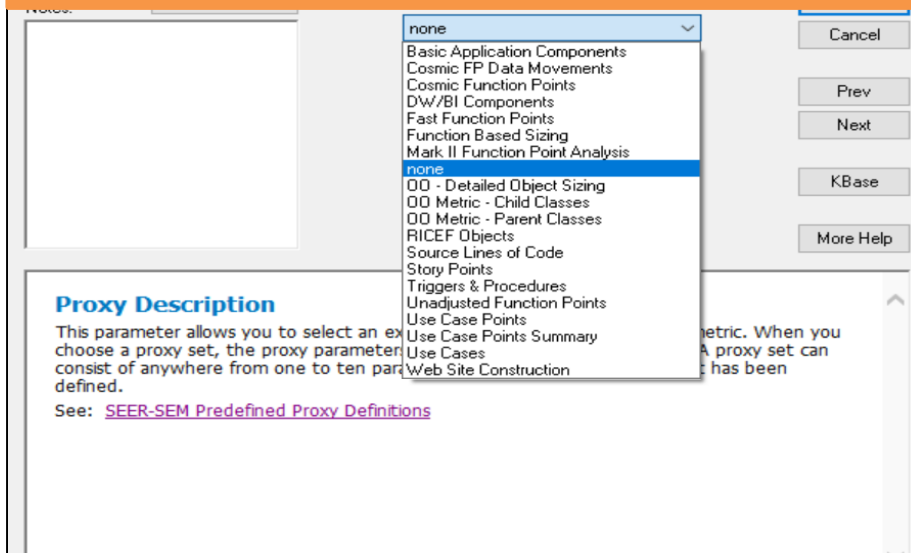
🌀 Best Practices

- Collect user stories, and any available artifacts describing functionality
- Remove non-functional requirements
- Include technical debt and refactoring user stories
- Utilize function point estimate where applicable, due to being a uniform standard
- When using proxy sizing (user stories, use cases), understand there is not a uniform standard, and make sure to calibrate local data
- Use Cumulative Flow charts to identify bottlenecks and also to develop schedule estimates

Parametric-Based Estimation

- ❁ Parametric Models provide proxy sizing, which allows sizing by User Stories and variety of other sizing approaches
- ❁ Tools also allow models to be aligned to particular flavor of Agile
- ❁ Caveat: be careful of using parametric models without training, and calibrate to your organization's data and environment

SEER for Software: Proxy Sizing



The screenshot shows the SEER for Software Proxy Sizing interface. A dropdown menu is open, listing various proxy sizing methods. The 'none' option is currently selected. Below the dropdown, there is a 'Proxy Description' section with text explaining the parameter and a link to 'SEER-SEM Predefined Proxy Definitions'. Navigation buttons like 'Cancel', 'Prev', 'Next', 'KBase', and 'More Help' are visible on the right side of the dialog.

Proxy Description
This parameter allows you to select an existing proxy set. When you choose a proxy set, the proxy parameter is automatically populated with the proxy set name. A proxy set can consist of anywhere from one to ten parameters. A proxy set can be defined.
See: [SEER-SEM Predefined Proxy Definitions](#)

TruePlanning: Agile Parameters

Development Process Details	
Development Process	Agile ▼
Planning Cycle	Iterations ▼
Process Formality	Tailored Process ▼
Number of Spirals or Increments	10.00
Software Requirements Stability	Fully Evolutionary ▼
Customer Involvement	Full ▼
Sequential Phasing	Some Overlap ▼
Level of Rework Expected	High ▼

Cost Estimation Process

Improvement related to Agile



Acknowledging the modern approach to software development, Cost Estimators need to update their deliverables and processes



Review Agile metrics to understand remaining scope and challenges



5 year development efforts are not realistic in an Agile environment.
Produce 1-2 year estimates



As Agile reduces comprehensive documentation, executive presentations are preferred to lengthy Basis of Estimates



Estimators need to produce products aligned to Agile cycles

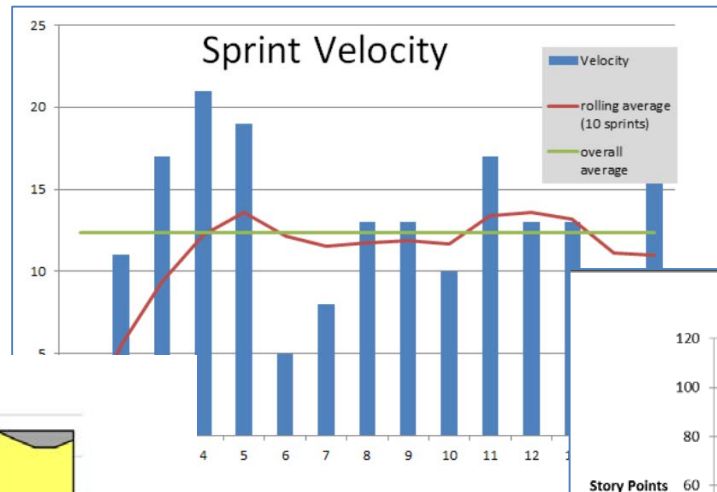


Reduce analysis cycle time from several months to weeks

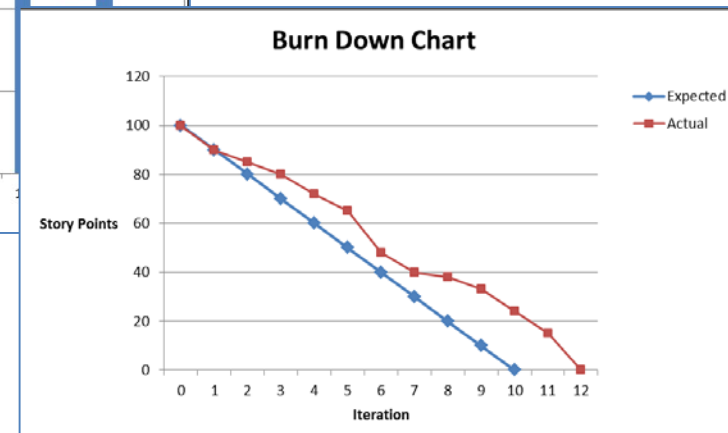
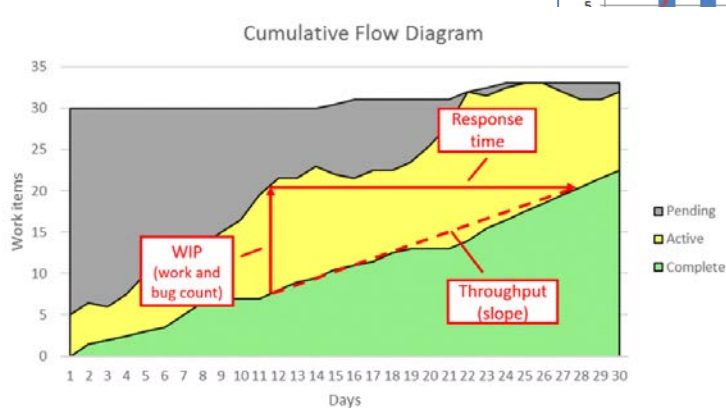
Agile Metrics

- Metrics help identify performance to date and insight into work remaining and potential bottlenecks
- Common metrics are cumulative flow, burn down/burn up charts, and velocity

Identifies WIP, throughput and cycle time



Identifies output per sprint and performance improvement



Displays plan versus actual

Schedule Crosschecks

Cumulative Flow

- 🌀 Work-in-Progress: User Stories in process
- 🌀 Cycle time: Duration per User Story
- 🌀 Throughput: Measure of stories that traverse process in given period
- 🌀 Little's Law: $\text{Throughput} = \text{WIP} / \text{Cycle Time}$
- 🌀 **Remaining Sprints = User Stories in Backlog/Throughput**
- 🌀 Note: User Stories are preferred to Story Points as measure due to story point inconsistency across teams

Velocity

- 🌀 Velocity: Story points completed per Sprint
- 🌀 **Remaining Sprints = Story Points in Backlog/Velocity**
- 🌀 This approach is not recommended due to story point variability across teams, and also that velocity can be impacted by story point inflation

IRS Case Study: Web Apps

Background

- Web Apps was the first IRS project to development within the Agile development framework
- The project was initiated in 2015, as a customer serving, public facing, mobile-friendly portal to provide payment and refund capabilities to individuals, businesses and tax practitioners

Approach

- Software size/effort based on level of effort for in-flight projects, function point analysis for projects referenced in VSA or design document/user stories, and analogy for out-year projects
- Analysis team developed both ICEs, and estimate for project Increments
- Analysis developed a streamlined Basis of Estimate tailored to Agile projects

Challenge

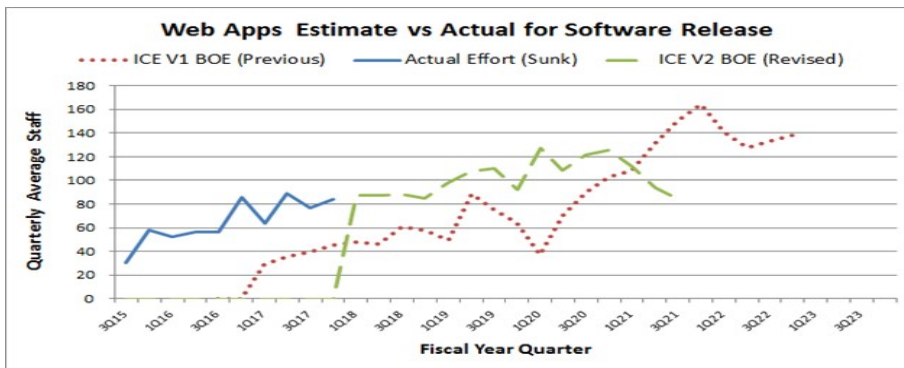
- Project Manager relied upon internal team estimates for software capabilities
- Lack of trust between Project Manager and analysis team
- Few details on capabilities beyond current fiscal year were documented

Impact

- Analysis Team has aligned its delivery tempo to provide estimation support when most needed, not based on fiscal year cycle
- Web Apps using Analysis team to deliver estimates for specific iterations, as well as for annual ICE to support budget
- Deliverables focus on comparison to program budget

Observations

- Analysis Team is using Web Apps as template for future Agile project estimates
- 3 year ICE produced in FY17, as few details available for subsequent years
- Ongoing communication between Analysis team and Web Apps will inform future ICE updates



Recommendations

Logapps recommends the following ground rules for Agile project estimation

- ❁ Create a library of Agile Project cost histories to inform future projects
- ❁ Share Agile analysis experience with estimation community
- ❁ Implement an Agile project metric collection program
- ❁ Speak in terms Agile Project Managers are familiar
- ❁ Develop estimates within reduced time frame
- ❁ Utilize different software estimation and sizing techniques
- ❁ Streamline cost estimation deliverables
- ❁ Provide cost estimates outside of annual cycle
- ❁ Deliver ad hoc analysis, and provide decision-support analysis
- ❁ Develop schedule estimates using cumulative flow data