# Reconciling CMMI®-DEV Processes With Agile Software Development

November 6, 2012

Richard Chipman
*VP Chief Systems Engineer*
*Airborne Systems Integration Operation*

Science Applications International Corporation

# Agenda

- What is Agile SW Development
- Agile Practitioners' Perspective
- CMMI® Practitioners' Perspective
- Mapping Agile to CMMI®-DEV Process Areas
- Estimating Agile SW Development
- Examples
- Conclusions

SAIC®

# What is Agile?

- SW development methods based on iterative & incremental development
- Requirements & solutions evolve through collaboration within self-organizing, cross-functional teams
- Time-boxed iterative approach
- Adaptive planning
- Rapid and flexible response to change

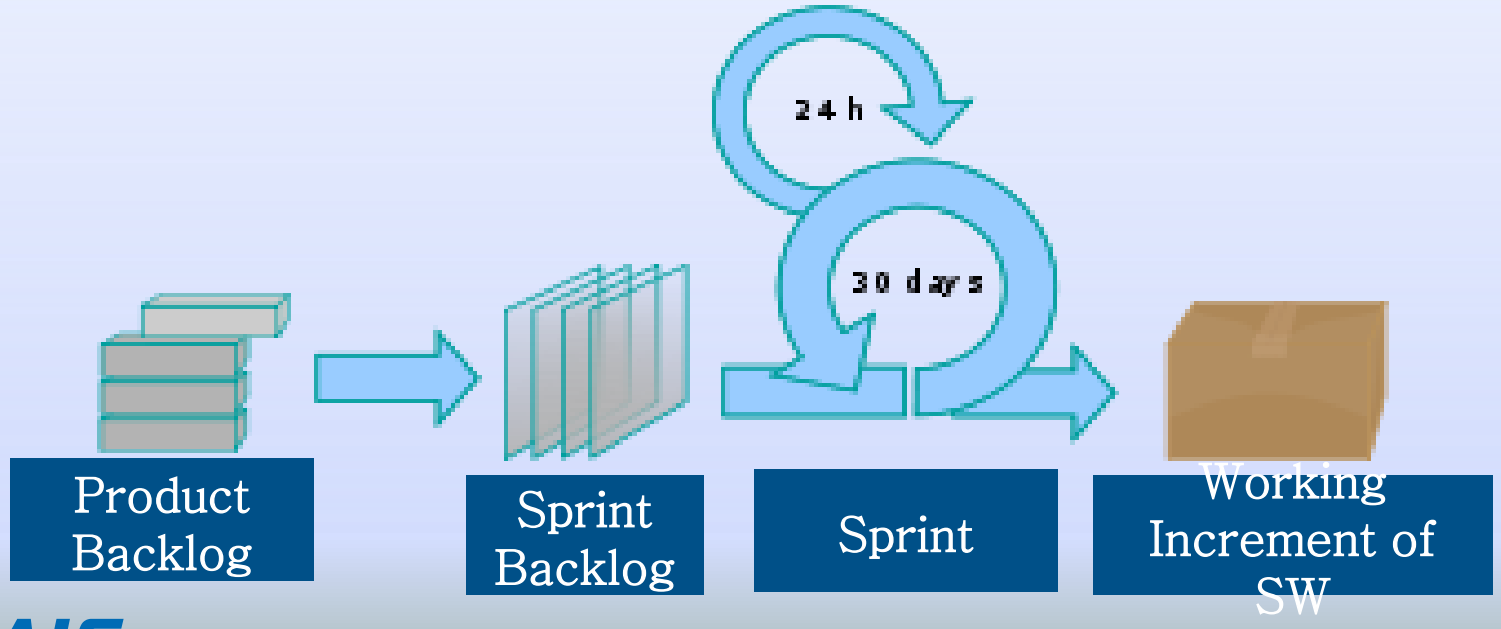
***Benefits:***

- ***Increased Productivity (25 - 50%)***
- ***Responsiveness to shifting customer needs & priorities***
- ***Frequent , concrete "products"***


SAIC

# Concept of a SCRUM

- SCRUM is aspecific Agile method -- almost become synonomous with Agile
- Organizes and prioritizes backlog into function sets
- Estimates development time/effort for each
- Selects top "N" to develop in a sprint (rapid, short development cycle)



# Estimation with SCRUM "Storyboards"

- Prioritize most important required functions on backlog
- Develop a simple "story" for each desired functionality (similar to scenarios and use cases)
- Each member of development team independently assigns a relative level of effort to story using the numbers in a Fibonacci series (reflects the uncertainty associated with larger projects)
- For each story, if estimates are close enough, it is assigned that relative rating
- If estimates are far apart, highest and lowest estimates explain their rationale, and team votes again
- Iterated until acceptable convergence is reached
- Complete rating for all functions/stories
- This is a rolling list, so number of un-rated is large only at beginning

SAIC

# Selecting Work Set for Sprint

- After all items have been weighted by effort, the effort associated with the smallest task is determined by “triangulating” to size/effort of similar known past tasks
  - Need to develop a set of reference data for “atomic” functionalities
  - Estimate reflects productivity of past projects
- Other tasks are estimated by scaling the effort by the ratio of the task’s weighting to rating of smallest task (usually =1)
- For next sprint (time box), determine available effort (resource  hrs)
- Determine top "N" items that can be done with this effort
- This process can be done at various granularities, e.g. sprints within releases

# SCRUM Development Rhythm

- Short daily team-only meetings to state prior and current days' tasks, and to discussion obstacles and how to remove them
- Team performs design, unit code & test of each distinct SW part needed to complete each function
- Integration & integrated test take place when required
- Sprint continues until end of pre-defined time box; anything uncompleted goes back on list as top priority
- Set of functions completed is product of the sprint
- This product can / should be demo'd to customer, but not necessarily released
- Release schedule is a programmatic decision

# Agile Practitioner's View of Process

**Manifesto for Agile Software Development**

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

http://agilemaifesto.org

**Observation:**

***Most SW developers hate processes, avoid documentation like the plague, don't like firm requirements and diverge from plan frequently.***

***The manifesto is almost a religious dogma; however, underlying the practice of Agile are some sound concepts with which we can agree.***

SAIC

# Agile Practitioner’s View of Estimation

- Most SW developers hate estimation, because it makes them define their product early in the life cycle.  They prefer to "see how far we get"; "try some things first"; or "get a release or two done first."
- Part of Agile's appeal to the SW developer is the false belief that agile is a "best effort" strategy.
- Agile practitioners have invented their own terminolgy for the parts of the traditional process they dislike  -- however, they still do them!
  - Use Case Scenario - Story
  - SW Productivity – Sprint Velocity
  - Requirements – Features
  - Function Point Estimation – Story Point Counting
  - Work Estimation – “Liar’s Poker”

SAIC

# Principles behind the Agile Manifesto

- Highest priority: satisfy customer by early & continuous delivery of SW
- Welcome changing requirements, even late in development
- Deliver working SW frequently, from a couple of weeks to months
- Working software is the primary measure of progress
- Business people & developers work together daily throughout project
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- Most efficient, effective way to convey information within a team is face-to-face conversation

# Principles behind the Agile Manifesto

- Agile processes promote sustainable development. Should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence & good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

SAIC

# CMMI® Practitioners' Perspective

- Defined, repeatable processes are essential to predictable, high-quality results in Software development and other types of projects
- Lack of defined processes is indicative of low productivity and poor quality
- CMMI is a model from which organizations can define the processes that work best for them
- Continual review and revision of processes result in improved quality and productivity
- Collection and analysis of metrics help organizations and projects know objectively how they are doing, facilitating problem resolution
- Sound estimation based on past productivity metrics and rigorous SW size estimation

SAIC®

# Mapping Agile to CMMI Process Areas

| CMMI Process Area | Agile Methodology |
|---|---|
| REQM: Requirements Management | Although Agile prefers to call them features, user requirements are a top priority. Agile is flexible about changes in requirements but if the change impacts scheduled deliverables, they put the change back into the backlog, and deliver it in the next sprint |
| PP: Project Planning | Each sprint is planned well; the difference is the involvement of the entire team in the planning process. Although the SW developers probably don't want to be pinned down, the Scrum master and PM are exacting about adherence to the plan. Peer pressure helps enforce the plan within the team. |
| PMC: Project Monitoring and Control | At a formal level, Agile doesn't report out during a Sprint. At a practical level, the team reports out to itself every day. This degree of monitoring is more than in a traditional process and is very effective. Issues still can arise at the end of a Sprint that fails to deliver the planned work. |
| MA: Measurement | Because the team relies on knowledge and |

# Mapping Agile to CMMI Process Areas

| CMMI Process Area | Agile Methodology |
|---|---|
| CM: Configuration Management | Agile is surprisingly silent about this process area; however, this is probably because they have no issue with following sound practices to avoid chaos. It is safe to say this is a process area that all endorse and practice. |
| TS: Technical Solution | As a process area, TS has always defied regimentation. Creative teams always seem to function better without being told how to create. The agile community at large places great value in fostering and encouraging creativity within the team. |
| OT: Organizational Training | Training has been an integral part of the Agile Process since its inception. Most organizations train entire teams as part of initial implementation, and continue with periodic updates. |
| QPM: Quantitative | Reality seems at odds with mythology on this topic. |

# Mapping Agile to CMMI Practices

- Establish an Organizational Policy
  - Adoption of Agile
- Establish a Defined Process
  - Agile process is well defined
  - Tailor it to the team
- Plan the Process
  - Agile stresses iterative planning
  - Each Sprint starts with Planning
- Provide the Resources
  - Staff the agile team and assign SCRUM Master
- Assign Responsibility
  - Empower team & SCRUM master
- Train People
  - Agile Training is part of process
- Manage Configurations
  - Same as traditional SW process
- Involve Relevant Stakeholders
  - Basic tenet of Agile
- Monitor and Control the Process
  - Daily SCRUM & SCRUM Master provide control
- Objectively Evaluate Adherence
  - Sprints end with self evaluation
- Collect Improvement Information
  - Agile measures velocity

SAIC

# Estimation in Agile (Story Points)

- What is a Story?
  - Brief top-level description of user functional requirement
  - Performance and other requirements are considered part of the story, unless they constitute a cross-cutting top level requirement
  - Story is smaller than a use case, larger than a function point type
- What are Story Points?
  - Arbitrary scale applied to user stories to define relative size and complexity of a each story as compared to other stories
  - Unit-less, meaning as a scale, they are only valuable to compare against each other within the same team
  - Used in conjunction with knowledge of team's past velocity as a guide to planning a sprint

# Estimating with Story Points (Planning Poker)

- Prioritize most important required functions on backlog
- Develop a simple “story” for each desired functionality
- Estimating team should be the development team including SME
- Each member of development team independently assigns a relative level of effort to each story using the numbers in a Fibonacci series (1,2,3,5,8,13,…)
  - Fibonacci series reflect the uncertainty associated with larger projects
- For each story, if estimates are close enough, it is assigned that relative rating
- If estimates are far apart, highest and lowest estimates explain their rationale, and team votes again
- Iterate until acceptable convergence is reached

SAIC

# Estimating Scope of a Sprint

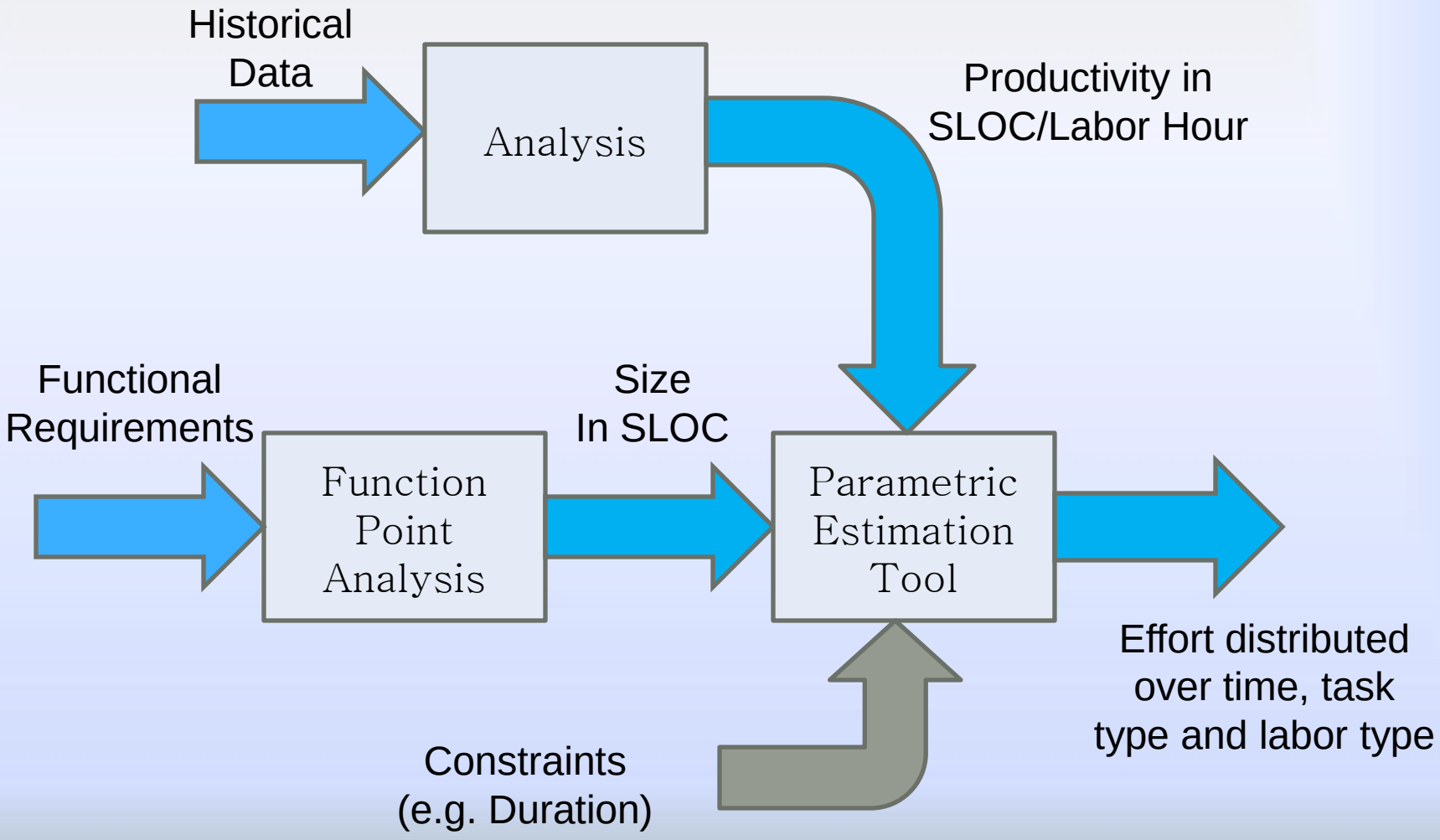
- Determining How Much Fits into a Sprint
  - Team’s past velocity measures the number of story points that can be implemented in a sprint: V= SP/sprint
  - After all items on backlog have been assigned a story point count, determine the top "N" items that can be done during the next sprint
  - Helpful to develop a set of reference data for “atomic” functionalities
  - Estimate reflects productivity/velocity of past projects; but the team’s velocity improves as team’s experience with Agile increases
  - Hence, velocity should be updated as team accumulates completed sprints

# Estimating an Entire Project / Responding to Requests for Proposal

- Agile does not provide software size or cost estimates
  - Velocity and story points are relative measures and team/project specific
  - Teams have  vastly different interpretations of Agile; hence, productivity is very much a function of the local practice
  - Difficult to build a supportable BOE for costing and proposal purposes
- Nonetheless, DoD and corporate management demand upfront estimates of SW effort, duration and cost
  - Costs and effort, based on known and understood methods and models
  - Requires that available data and tools be used to generate an estimate that is acceptable to customers and management
- What about all of the metrics you've compiled under CMMI?

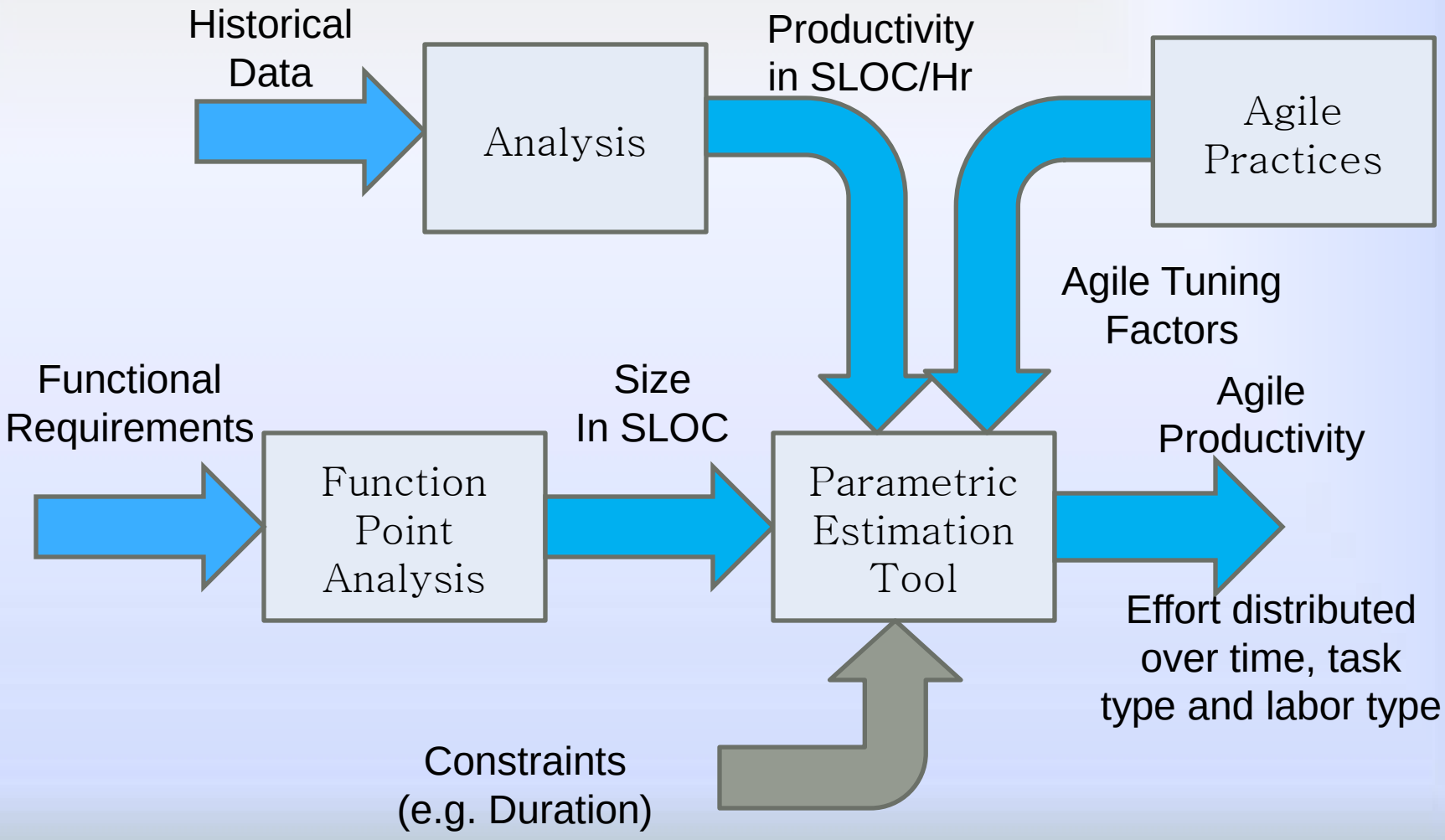
SAIC

# Traditional Estimation Process




SAIC

# Estimation Process

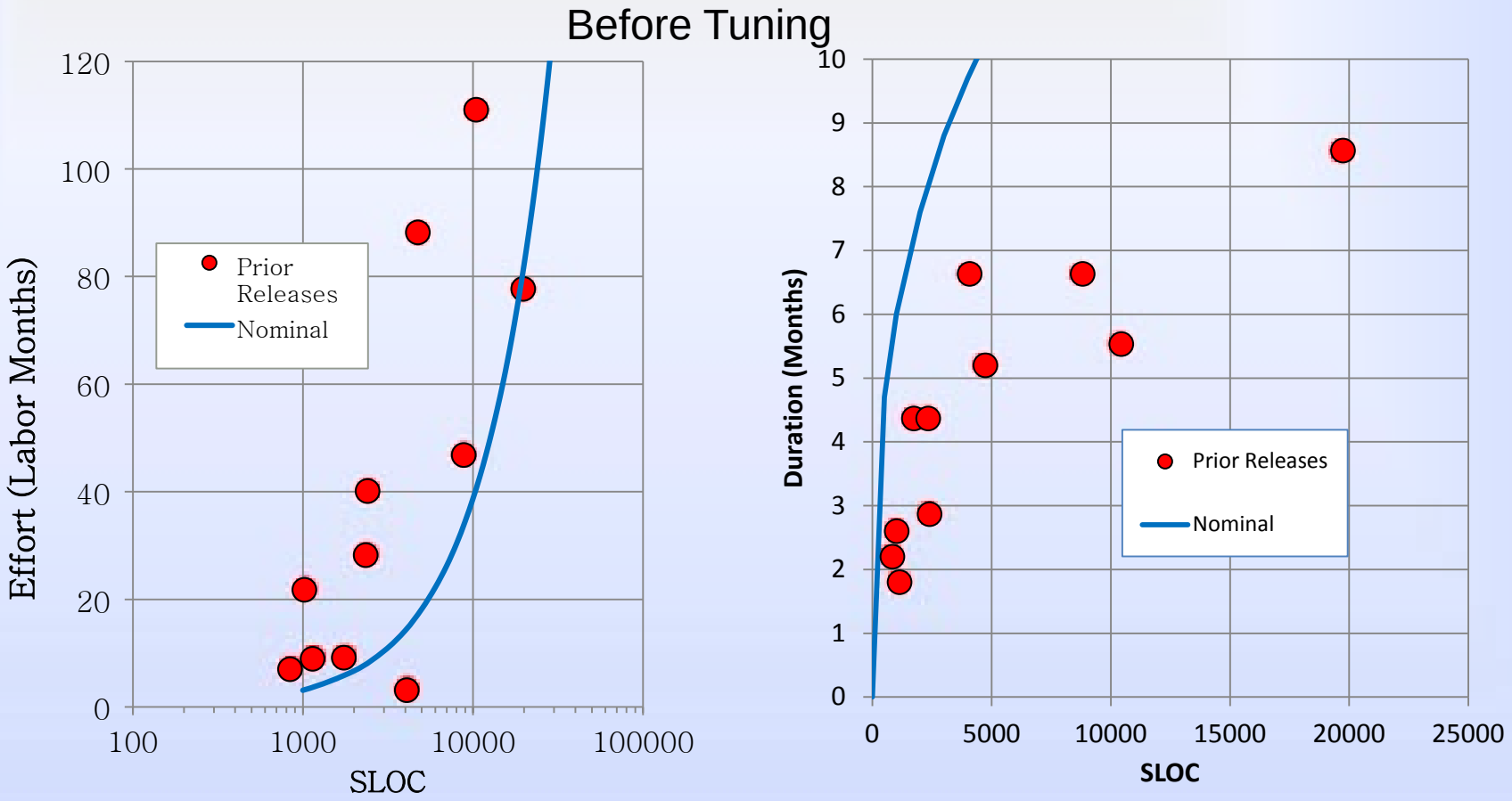



SAIC

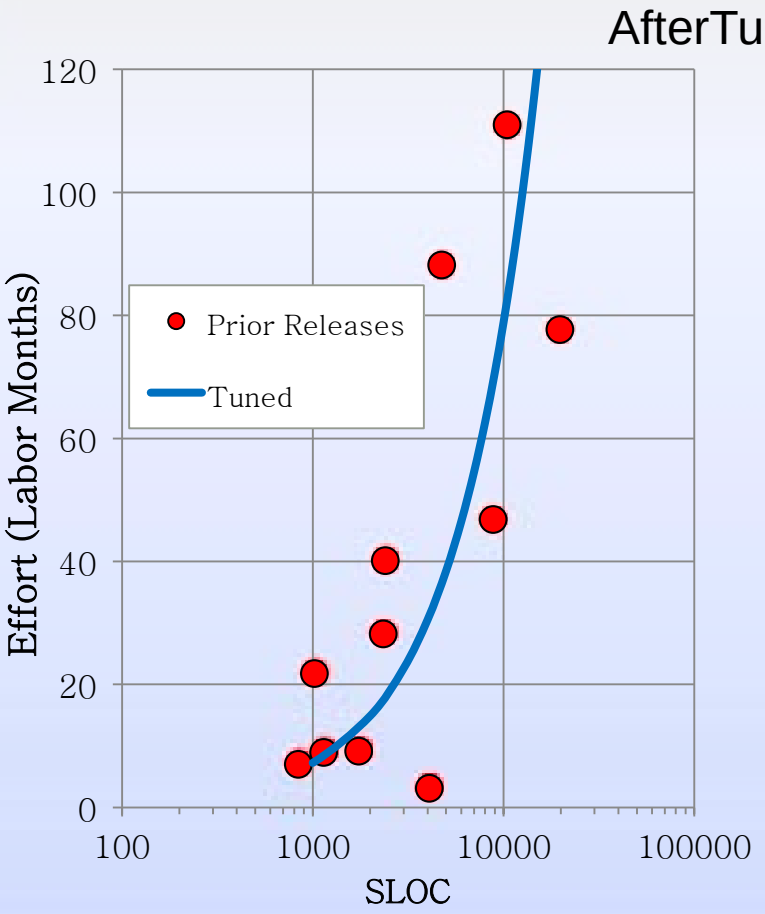
# Approach to Estimation of Agile SW Development

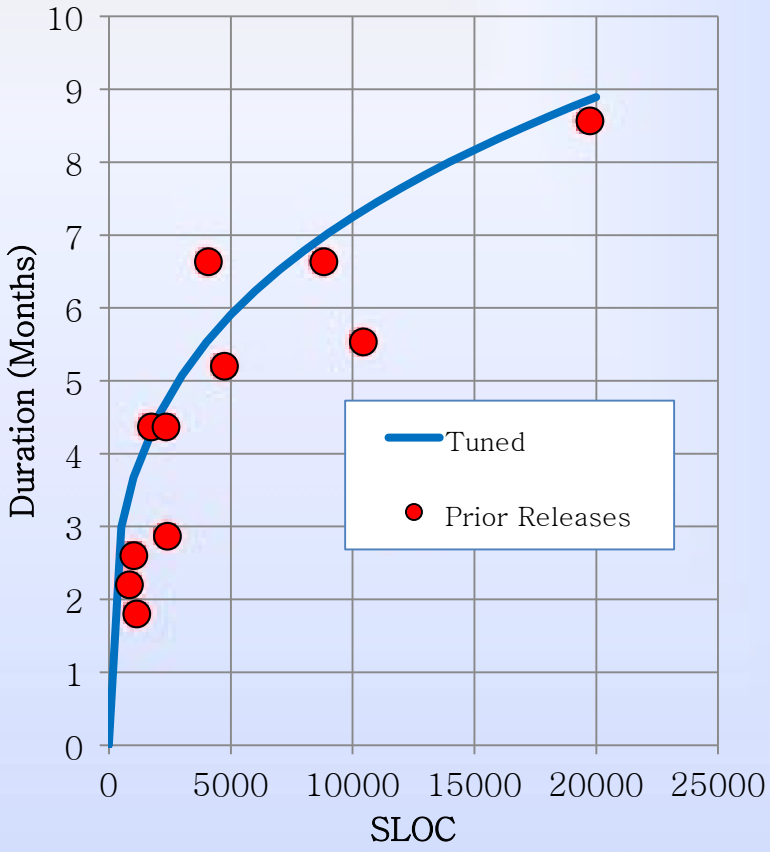
- Build parametric model in COCOMO, Price-S, SEER-SIM, etc.
- Tune model's factors to match your historical, non-agile, productivity
- Identity factors that Agile will impact
- Adjust those factors incrementally to determine the effects of adoption of agile over time; i.e. modest improvements at first growing to their maximum as team becomes fully agile
- Measure productivity of each sprint; accummulate and analyze over time to update your productivity for use in estimating future work

# Example Use Of Historical Data To Tune COCOMO Model



# Example Use Of Historical Data To Tune COCOMO Model




SAIC

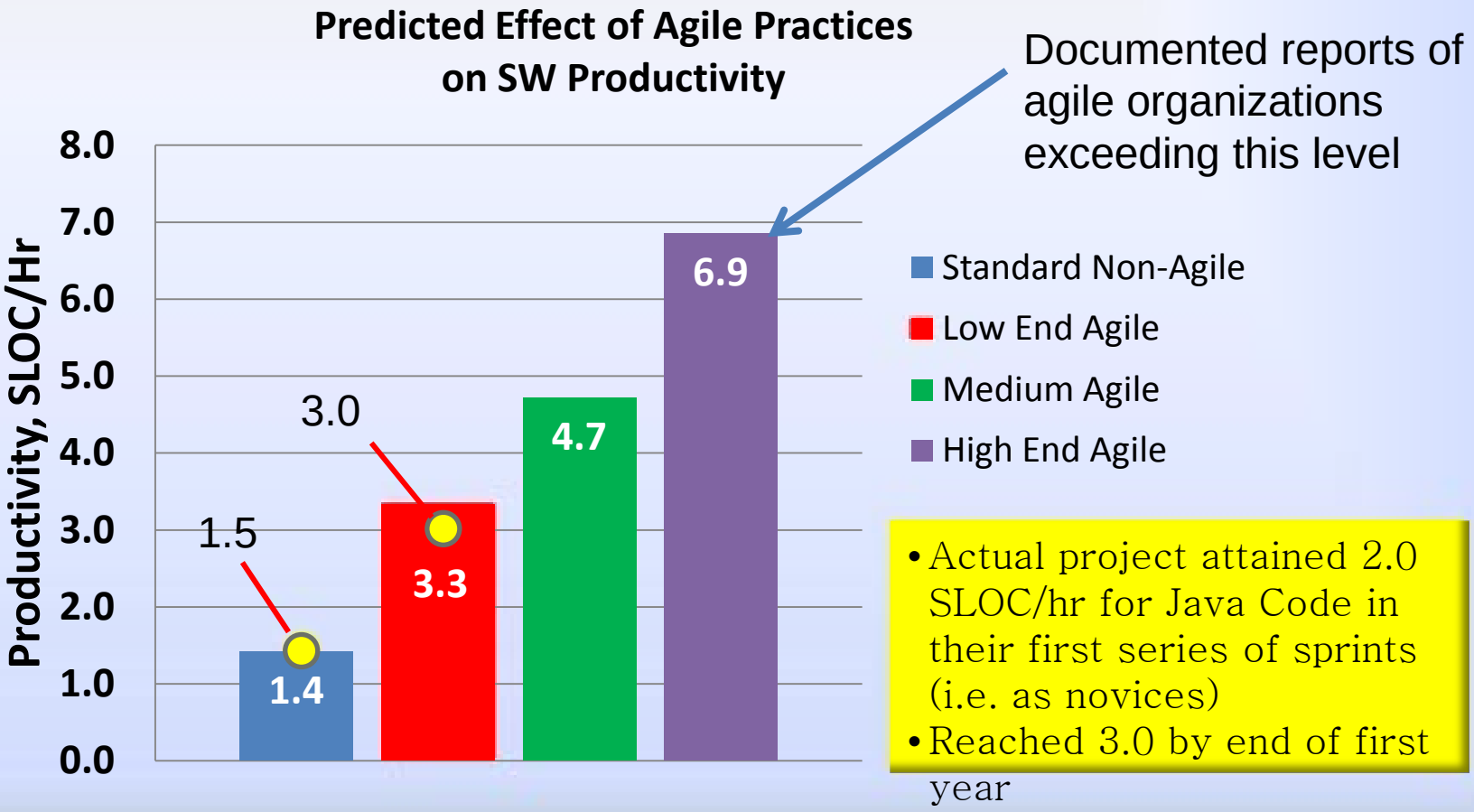
# COCOMO Factors for Agile

- Development Flexibility (FLEX)
  - Need for software conformance with pre-established requirements
  - Adjust to be more flexible
- Platform Volatility (PVOL)
  - Sprint is so rapid that platform is very unlikely to change
  - Reduce setting
- Personnel Continuity (PCON)
  - Sprint is so rapid that personnel are very unlikely to change
  - Increase setting
- Multisite Development (SITE)
  - Agile practice relies on daily Scrums, staff tend to be collocated
  - Increase setting

# COCOMO Factors for Agile

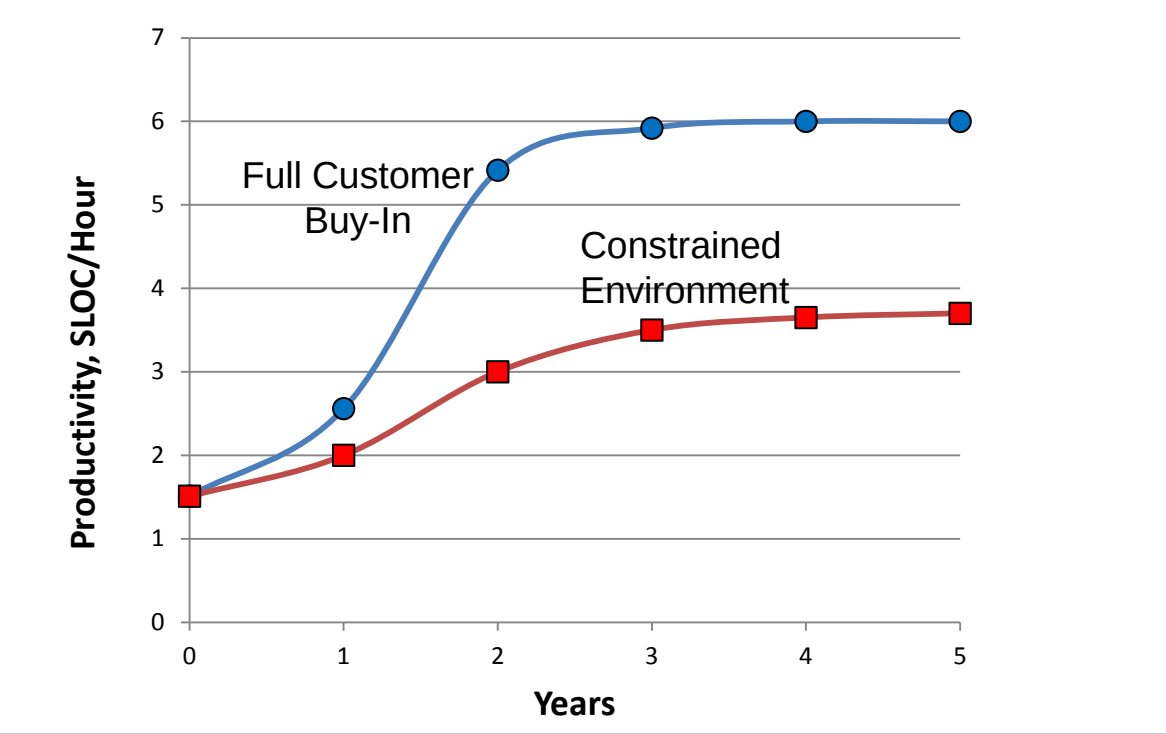
- Documentation Match to Life-Cycle Needs (DOCU)
  - Agile philosophy is to emphasize product, not documentation
  - Reduce setting
- Use of Software Tools (TOOL)
  - Agile typically uses strong, mature lifecycle tools, moderately integrated
  - Increase setting
- Analyst Capability (ACAP)
  - Agile tends to attract high performers and reinforces their productivity
  - Increase setting
- Programmer Capability (PCAP)
  - Agile tends to attract high performers and reinforces their productivity
  - Increase setting

# Example of Agile Tuning in COCOMO



# Effect of Adoption of Agile over Time

- Experience shows that adoption of Agile does not produce immediate results; however, the impact is noticeable within the first year.
- The impact appears much greater in year two and levels off thereafter.



# Concluding Remarks

- Adoption of Agile can lead to improved productivity if used for the right type of project:
  - Continual backlog of work that differs little in type from its predecessors
  - Little to no dependence on other evolving SW
  - Team continuity assured by project stability
- CMMI practices are still valid, albeit the Agile community has renamed them
- Estimating process for future work is totally different from the week to week estimation within the Sprint / SCRUM; and follows traditional approaches
  - Understanding how to apply standard techniques such as COCOMO, PRICE-S and SEER-SEM is essential to obtaining accurate estimates
  - Experience to date indicates the techniques are completely valid and estimates can be reliably developed.

# Author Contact Information


- Richard Chipman
- (703) 966-4808
- Science Applications International Corporation (SAIC)
- chipmanr@saic.com


SAIC