Adventures in Estimating

Open Source, Component Systems, Agile, and SOA Projects

Terry Vogt
Lead Associate
Booz Allen Hamilton

Sept 13, 2011

Adventures in Estimating

Agenda

- Background
- Open Source
- Component Systems
- Agile
- SOA
- Future Capabilities
- Wrap Up

Adventures in Estimating

Background – many different types of work

- Our Clients –a wide variety of user and situations
  - US Dept of Defense
  - US Civil Agencies
  - US National Security Agencies
  - Commercial and Non-Profit Entities
- Our Work – different objectives for different needs
  - Estimates
  - Performance Assessment
  - Process Improvement
- Our Tools – recognized standard processes and industry tools
  - Function Point Analysis
  - Parametric Software Estimation Tools
  - Parametric Combination Software/Hardware Estimation Tools

Adventures in Estimating

Agenda

- Background
- Open Source
- Component Systems
- Agile
- SOA
- Future Capabilities
- Wrap Up

Adventures in Estimating

Open Source

- What It Is
  - Open source software is software whose source code is published and made available to the public, enabling anyone to copy, modify and redistribute the source code without paying royalties or fees.
  - Essentially free software from the public domain provided under a software license that permits users to study, change, and improve the software
  - Supported by groups of users who contribute to maintenance
  - Those who use the software are motivated to support and maintain it
  - Many diverse general and specialized applications available e.g. Linux, Firefox

Adventures in Estimating

Open Source

- How It's Done
  - Anyone can contribute software to public domain
  - A general non-restrictive use license is granted to anyone using it
  - Anyone wanting a specialized version can create one and support it
  - New versions of software become widely adopted if they meet the needs of a wider group of users than just the originator
  - If new (or modified) software is adopted for use by others, they will typically help to maintain it, otherwise maintenance is the responsibility of the originator

Adventures in Estimating

Open Source

- How Do We Measure It
  - Measured same as COTS or reused software
  - User requirements define needs
  - Open source contains some or all required user functionality – typically includes additional functionality not needed – must weigh the impact of accepting the unneeded functionality along with that which is needed
  - Comparison of user needs and existing application functionality identifies extent of modifications needed – becomes enhancement/development
  - Total cost, delivery schedule and other characteristics of each alternative (including non-Open Source) are compared to choose preferred solution
  - Maintenance responsibility will be resolved by the response of the broader user community to adopt and support the software or not – for variants of specialized applications assume maintenance will be an ongoing cost

Adventures in Estimating

Open Source

- How We Work It with Estimation Tools
  - Platform is typically internet based (but not always)
  - Primary acquisition type is integration with or without change to software depending on whether existing software fully meets user needs
  - Software modifications and new development are modeled and estimated as usual – separate acquisition types
  - All software involved is modeled and estimated for integration and test
  - Integrated cost and schedule effort estimates for each complete alternative are compared against alternatives which may include traditional development or COTS
  - Additional comparative costs outside often apply to comparative evaluation of alternative solutions: installation, training, O&M

Adventures in Estimating

Agenda

- Background
- Open Source
- Component Systems
- Agile
- SOA
- Future Capabilities
- Wrap Up

Adventures in Estimating

Component Systems

- What It Is
  - Component Systems involve software that is built or acquired in pieces and then assembled to provide a complete set of functionality
  - Software may come from a variety of sources and usually includes embedded software in some components
  - Each of these software components has its own functionality and interfaces with one or more other components
  - Examples of component systems include communications software, satellite systems, process controls, robotics

Adventures in Estimating

Component Systems

- How It's Done
  - Each component is designed to be sourced separately but integrated into a complete system
  - Different sources make it more complex to design, test and ensure it performs as intended
  - Frequently there are security features in the embedded components that make the systems a collection of "black boxes"
  - In an Object Oriented framework the components do not need to be designed with knowledge of their internal characteristics
  - Integration is a major part of project effort
  - Mixed platforms, performance requirements, quality standards and other differences can complicate the design and the estimate

Adventures in Estimating

Component Systems

- How Do We Measure It
  - User requirements define needs. Design identifies technical platform differences. Technical and sourcing differences affect FPA sizing approach to properly estimate effort.
  - Each component is sized according to standard FPA rules, but sizes must be segregated to be used to model effort for each component
  - Component qualitative characteristics such as performance can vary a great deal
  - Extent of functionality provided by each existing candidate component identifies the extent of any modifications needed – becomes a question of enhancement versus development and source A versus source B
  - Total effort, cost, delivery schedule and other characteristics of each alternative combination of components are compared to choose a preferred complete solution

Adventures in Estimating

Component Systems

- How We Work It with Estimation Tools
  - Almost always involves multiple platforms due to the requirements and availability of component solutions. Each component is separately characterized.
  - Acquisition type varies by component – new development or integration with (or without) change to software depending on extent to which candidate software fully meets user needs
  - Software modifications and new development are modeled and estimated as usual – separate acquisition types, platforms, sources, etc.
  - The complete set of components in the solution scenario is modeled and estimated for integration and test
  - Integrated cost and schedule effort estimates for each complete alternative are compared against alternatives.
  - Additional comparative costs may include installation, training, O&M

Adventures in Estimating

Agenda

- Background
- Open Source
- Component Systems
- Agile
- SOA
- Future Capabilities
- Wrap Up

Adventures in Estimating

Agile

- What It Is
  - Agile is a software development approach or methodology
  - Agile is not a specific toolset
  - Emphasis is on delivering software, not documentation
  - Intense interaction with client/user throughout development process is a touchstone of this approach – trade-off for documentation and contract renegotiation
  - Short cycles to deliver working software are a driving factor in development
  - Requirements changes are welcomed at any point in SDLC
  - Rigor is a relative thing – "trust dedicated people"
  - While it appears very informal, Agile development can be highly disciplined

Adventures in Estimating

Agile

- How It's Done
  - Small teams iterate development in many short cycles
  - User participation in development activity is critical
  - Requirements volatility tends to be high
  - Requirements management and control tends to be informal
  - Skills and experience of the team in Agile development govern productivity
  - Productivity is expected to accelerate significantly with experience
  - Team size is small: typically 7 +/- 2 This small size and short project cycle dictate the small size of incremental deliveries
  - Multiple small teams can develop concurrently, however such parallel project scenarios are difficult to manage effectively – integration effort
  - Reuse of software from various sources can be part of the strategy

Adventures in Estimating

Agile

- How Do We Measure It
  - Each functional requirement is identified and measured for functional size
  - Requirements to express functional size are initially very high level – must be inflated to come closer to reality of eventual size of delivery
  - Requirements become specific as the project proceeds and the development team engages the user in the process – assumptions are decreased and estimates become sharper
  - Short development cycles – "scrums" – define specific requirements in scope for current delivery and then develop and deliver that functionality – so each scrum can be estimated with high precision
  - Each scrum is modeled separately – different platforms, acquisition methods etc. may apply to each; experience levels will likely evolve
  - Requirements for future delivery accumulate in "backlog"

Adventures in Estimating

Agile

- How Do We Work It with Estimation Tools
  - Agile team performance characteristics – important to distinguish between Novice and Experienced team experience level– different performance
  - Team size and experience and short delivery cycle govern delivery capacity
  - Projection is made of effect on delivery productivity of increased experience
  - First project is modeled based on industry performance experiences
  - Subsequent projects are modeled with performance calibrated to local norms – this is essential to get estimates in line with local performance
  - Estimates for future delivery cycles are made against existing backlog
  - Backlog can be further amended to factor new requirements growth rate, team experience and effect of maintenance effort required to support accumulated software delivered by successive delivery cycles
  - Estimation is a matter of delivery capacity versus size of scrum and backlog

Adventures in Estimating

Agenda

- Background
- Open Source
- Component Systems
- Agile
- SOA
- Future Capabilities
- Wrap Up

Adventures in Estimating

SOA

- What It Is
  - SOA = Service Oriented Architecture
  - SOA is a concept of building solutions from independent parts
  - SOA aims to allow users to form ad hoc applications that are built almost entirely from existing software services
  - A way of designing systems composed of services that are invoked as packages in a standard way – actually more of a pattern or conceptual approach rather than a specific architecture
  - SOA defines the interface in terms of protocols and functionality

Adventures in Estimating

SOA

- What It Is - continued
  - Services are reusable components that represent business or mission tasks: customer lookup, or transaction processing, etc.
  - Interoperability is the key characteristic - loose coupling of services
  - Reuse is the strategy to achieve productivity – the greater the occurrence of reuse, the greater the savings versus new development
  - Identification and suitability of available components is an issue
  - Responsibility for maintenance is an issue
  - Performance consideration in choosing the granularity of services: Trade off between larger software chunks with fewer interface points versus smaller chunks that are easier to use but require more processing overhead

Adventures in Estimating

SOA

- How It's Done
  - Aspects of application requirements are matched against components
  - Components are selected for use in a complete solution (-> inventory)
  - "Orchestration" - integration of components - as the full extent of required effort is the ideal
  - Component enhancement or new development is used where necessary
  - Trade-offs between use of an existing component, modification of an existing component, or development of a new component, must be considered and weighed against alternatives – this can be complicated
  - Governance board may be involved in trade-off decision-making process regarding component modification– this can constrain schedule
  - SOA efforts require well-designed information architectures in order to base their activities on trusted data -- or else the business value of the SOA project may evaporate - SOA will compound unresolved data problems

Adventures in Estimating

SOA

- How Do We Measure It
  - Functional requirements are compared to available component functionality (-> inventory)
  - SOA fundamental characteristics are Data Complexity, Service Complexity, Process Complexity, and Enabling Technology
  - Each component involved in the solution must be identified, characterized and scoped
  - Each component is identified as being an integration-only item or one requiring additional effort to provide required functionality
  - Proper scoping of components is critical – each component has its own boundary in terms of correctly measuring the effort required to achieve a solution – boundaries define points of data flow between components and extent of integration effort required
  - Infrastructure modeling issue – how to account for performance issues

Adventures in Estimating

SOA

- How Do We Work It with Estimation Tools
  - Every component is measured for integration effort
  - Effort and schedule are modeled for each component independent of other components
  - Components identified as needing enhancement are treated as enhancement projects
  - Components identified as needing development are identified as development projects
  - Recognize that testing can be significantly more complex than in traditional development due to complex interactive relationships under actual usage
  - Integrated cost and schedule estimate is modeled for combination of all components
  - Integration testing, usage performance requirements and validation and application complexity will be dominant factors in estimation/comparison

Adventures in Estimating

Agenda

- Background
- Open Source
- Component Systems
- Agile
- SOA
- Future Capabilities
- Wrap Up

Adventures in Estimating

Future Capabilities

- Capability Wanted
  - Estimation capability for Cloud Computing
- How It Will Be Provided
  - Acquire data from completed Cloud projects:
    Costs & effort to migrate application to cloud, configuration costs to use alternative application software, vendor service fees, etc.
  - Develop new model building approach in existing estimation tools for Cloud Computing: Different blends of delivery/deployment
- What We CanDo with Additional Capability
  - Demonstrate potential cost savings: Data aggregation, potential power savings, effects on development teams
  - Evaluate advantages of flexibility & scalability of this solution approach
  - Efficiently explore trade offs of alternative Cloud solutions
  - Model traditional solutions and compare against Cloud solutions

Adventures in Estimating

Future Capabilities

- Capability Wanted
  - Easier way to compare different solution outcomes
- How It Will Be Provided
  - Add capability in current estimation tools to open multiple project files
  - Allow some or all inputs to be shared/linked across projects
  - Provide output comparison data (tables & charts) across projects
  - Provide an estimation tool repository showing relationships among models
- What We Can Do with Additional Capability
  - Demonstrate contrasts between different solution approaches
  - Efficiently explore trade-offs of alternative solutions
  - Guarantee version control of estimation models

Adventures in Estimating

Future Capabilities

- Capability Wanted
  - Stratified view of processes, projects, programs, organizations, etc.
- How It Will Be Provided
  - Develop capability in estimation tools to link multiple model files to record and project data in multiple perspectives: different organizational levels, development roles, decision-making scope, etc.
  - Allow estimation models to be linked hierarchically
  - Allow different scopes to apply to different participant roles
- What We Can Do with Additional Capability
  - Illustrate impact of changes in data, assumptions, etc. through a scope of control greater than a single project
  - Illustrate effects of changes at any point to guide decision-making

Adventures in Estimating

Agenda

- Background
- Open Source
- Component Systems
- Agile
- SOA
- Future Capabilities
- Wrap Up

Adventures in Estimating

Wrap Up

- What We Have Learned
  - With a little training and demonstration users can understand FPA
    - Skepticism is overcome with familiarity and patience
    - Ultimately they understand the superiority of the functional sizing concept
  - A fool with a tool is still a fool – users need training and mentoring
    - Monkeys can push buttons and get output – it takes some intelligence and experience to produce useful answers
  - Calibration is better than good to have, it's essential
    - Every organization, team, approach, toolset, user and situation is different
    - Calibrated results are specific to the issue at hand and reflect local reality
  - Change in all things is inevitable – embrace it, plan for it, deal with it
    - Build flexibility into the estimation process – it will pay off
    - The next new paradigm in software development is always on its way

Adventures in Estimating

Wrap Up

- Things We Need to Improve
  - How to most effectively model trade-offs between potential solutions
  - How to show value (absolute and relative) in non-dollar outcomes
- Where Will We Go from Here
  - Continue to experiment and to devise extensions to existing tools
  - Work with vendors to enhance tool capabilities
  - Educate clients to inform them on capabilities, guide them on expectations, and listen to them to identify their needs

Adventures in Estimating

Contact info:

Terry Vogt

Booz Allen Hamilton

Email: vogt_terry@bah.com

Office: (703) 377-4567