



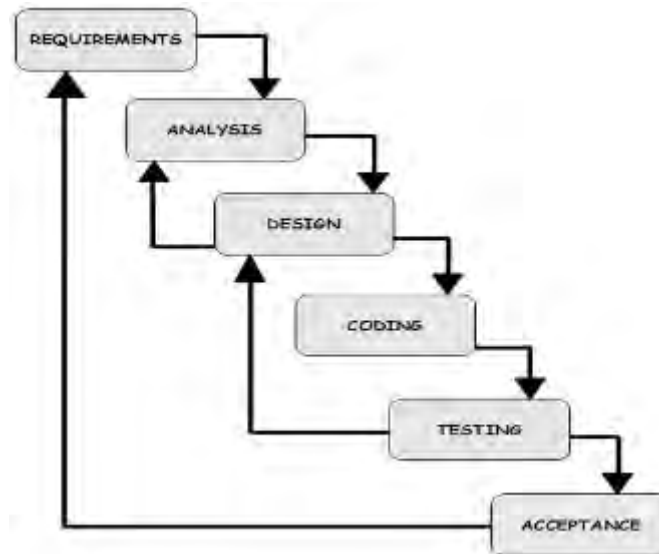






## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce



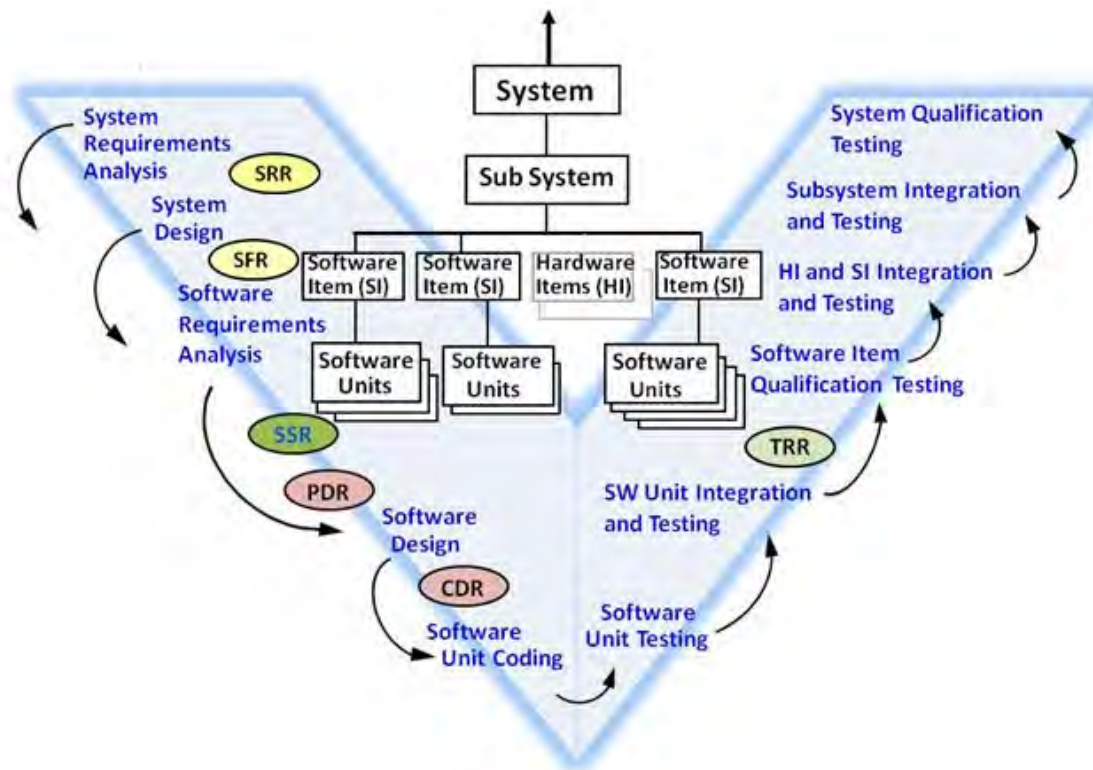
**Figure 1.0 Waterfall Software Development Concept**

In 1985, the United States Department of Defense captured this approach in DoD-STD-2167A, the standard for working with software development contractors, which stated that "the contractor shall implement a software development cycle that includes the following six phases: Preliminary Design, Detailed Design, Coding and Unit Testing, Integration, and Testing."<sup>8</sup> This methodology and associated processes is depicted in **Figure 2.0**.<sup>9</sup>

These overlapping phases of requirements analysis, architecture and preliminary design, detailed design, coding, and a series of tests beginning with software unit testing followed by integration and associated qualification testing were established in the 1970's.<sup>10</sup> Qualified Software Configuration Items (SCIs) are integrated into subsystems and finally into the complete system.

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce



**Figure 2.0** Traditional Waterfall Software Development Process and Activities

This traditional waterfall approach discourages revisiting and revising any prior phase once it is complete. This "inflexibility" has been a source of criticism by supporters of other more "flexible" models. It has been widely blamed for several large-scale government projects running over budget, over time, and sometimes failing to deliver on requirements due to the Big Design Up Front (BDUF) approach. The main criticism is this approach does not accommodate situations where the client does not know exactly what their requirements are before they see working software and so change their requirements, leading to redesign, redevelopment, and retesting, and increased costs.<sup>11</sup>

Designers may not be aware of future difficulties when designing a new software product or feature, in which case it is better to revise the design than continue with a design that does not account for newly discovered constraints, requirements, or problems. Except when contractually required, the waterfall model has been superseded by more flexible and versatile methodologies developed specifically for software development.<sup>5</sup> The DoD has

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

a stated preference against waterfall-type methodologies, starting with MIL-STD-498 (superseded by IEEE 12207).<sup>12</sup>

The weakness of the waterfall model led to the development of other software development approaches to reduce client uncertainty and risk and unfavorable cost and durations. Some of these approaches are

- Incremental development
- Evolutionary development
- Spiral development
- Rapid application development
- Agile development

### Incremental Development Model

The incremental life cycle model was one of the first variations to be derived from the waterfall model. The assumption behind the model is that the requirements can be segmented into an incremental series of the products, each of which is developed somewhat independently. The series of releases is referred to as “increments”, with each increment providing more functionality to the customers. After the first increment, a core product is delivered, which can already be used by the customer. Based on customer feedback, a plan is developed for the next increments, and modifications are made accordingly.

This process continues, with increments being delivered until the complete product is delivered. The incremental model has the following advantages: (1) Less cost and time is required to make the first delivery. (2) Less risk is incurred to develop the smaller systems represented by the increments. (3) User requirement changes may decrease because of the quicker time to first release. (4) Incremental funding is allowed; that is, only one or two increments might be funded when the program starts.<sup>13</sup>

From the estimators point of view, under this model (like the waterfall model), the client must define all requirements upfront and the estimator can use any number of the above parametric models to obtain reasonable estimates of the person months involved. But what if the client does not know all his specific requirements up front?

### The Evolutionary Development Model

The next step in life cycle development is the evolutionary model, which explicitly extends the incremental model to the requirements phase. In this model, the customer defines the requirements for “Build 1” and after delivery, he uses his experience with the first build to define the requirements for the next build. This process continues until all the

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

requirements are satisfied. The advantage of this approach is that there is greater user involvement and feedback than in the Incremental or classic waterfall models.<sup>14</sup>

From the cost estimators point of view, these parametric models can be used to estimate “Build 1” based on the stated requirements (and derived sizing factors), but the estimator has little information for estimating “Build 2” due to changes that the client may make after reviewing “Build 1.” This is a significant challenge for the estimating community. In many cases, the total project is incrementally funded and the cost estimator may obtain requirements for each build and thus be able to produce credible estimates for each build.

### Rapid Application Development Model

The Rapid Application Development (RAD) software development methodology favors iterative development and the rapid construction of prototypes instead of large amounts of up-front planning. The “planning” of software developed using RAD is interleaved with writing the software itself. The lack of extensive pre-planning generally allows software to be written much faster, and makes it easier to change requirements. The rapid development process starts with preliminary data models and business process models using structured development techniques. In the next stage of RAD, requirements are verified using prototyping, refining the data and process models. These stages are repeated iteratively; further development results in “a combined business requirements and technical design statement to be used for constructing new systems.

The basic principles of Rapid Application Development are:

- Fast development and delivery of a high quality system at a relatively low investment cost.
- Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
- Produce of high quality systems quickly, primarily via iterative Prototyping (at any stage of development), active user involvement, and computerized development tools. These tools may include Graphical User Interface (GUI) builders, Computer Aided Software Engineering (CASE) tools, Database Management Systems (DBMS), fourth-generation programming languages, code generators, and object-oriented techniques.
- Emphasis is on fulfilling the business need, while technological or engineering excellence is of lesser importance.
- Project controls for prioritizing development and defining delivery deadlines or “time-boxes”. If the project starts to slip, emphasis is on reducing requirements to fit the time-box, not in increasing the deadline.



## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

- Generally includes joint application design (JAD), where users are intensely involved in system design, via consensus building in either structured workshops, or electronically facilitated interaction.
- Active user involvement is imperative.
- Iteratively producing production software, as opposed to a throwaway prototype.
- Produces documentation necessary to facilitate future development and maintenance.
- Standard systems analysis and design methods can be fitted into this framework.<sup>12</sup>

It is difficult from the cost estimator's point of view to develop credible estimates for these efforts because there is no set of stable requirements to use as the sizing for the estimating model. A key attribute of this approach is the client and developer are permitted to reduce requirements to fit a time-box, while not increasing the deadline. Many cost estimators will estimate the effort associated with the prototype based on the stated requirements and then extent the personnel resources over the total projected development duration. If requirements are not known, the estimator is left to bounding the amount of effort by using the results of prior analogous efforts.

### The Spiral Model

A current development approach is the Spiral model that combines some key aspects of the waterfall model and with rapid prototyping methodologies in a top-down and bottom-up concept.<sup>12</sup> The Spiral model is a risk-driven controlled prototyping approach that develops early deliverables to specifically address risk areas followed by assessment of prototyping results and further determination of risk areas to prototype. Prototyped areas frequently include user requirements and algorithm performance. Prototyping continues until high risk areas are resolved and mitigated to an acceptable level.

This model is appropriate for exploratory projects that are working in an unfamiliar domain or with unproven technical approaches. The iterative nature allows the knowledge gained during early stages to inform subsequent stages. During each iteration or loop, the system is explored at greater depth and more detail is added.<sup>14</sup>

The spiral model starts with an initial pass through a standard waterfall life cycle, using a subset of the total requirements to develop a prototype. The set of requirements is hierarchical in nature, with additional functionality building on the first efforts.<sup>15</sup> With the assumption that functionality can be dropped, the estimator can use the parametric estimating tools. Use of similar historical results also lends credibility to the estimate.

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

### The Agile Development Model

The Agile software paradigm is a collection of software development methods where solutions evolve through collaboration between self-organizing, cross-functional teams. Agile promotes adaptive planning, evolutionary development, continuous delivery, continuous improvement, and rapid and flexible response to change.<sup>12</sup>

Development methods exist on a continuum from *adaptive* to *predictive*. Predictive methods, such as the classical waterfall model, focus is on analyzing and planning the future in detail and cater for known risks. In the extremes, a predictive team reports what features and tasks are planned for the entire length of the development process.

Predictive methods rely on effective early phase analysis and if this goes very wrong, the project may have difficulty changing direction. Agile methods lie on the *adaptive* side of this continuum where the focus is on adapting quickly to changing realities. When the needs of a project change, an adaptive team changes as well.<sup>5</sup>

While the use of the Agile model has been applied in the commercial environment where scope is often open ended, the DoD, and other federal agencies have been encouraging the use of the Agile model when the system being developed needs to rapidly adapt to emerging user needs.<sup>16</sup> Federal government and DoD have emphasized the necessity to shorten acquisition timelines to be responsive to increasing operating tempo and warfighter need with more rapid capability development. In 2009, the Defense Science Board wrote that “The fundamental problem DoD faces is that the deliberate process through which weapon systems and information technology are acquired does not match the speed at which new IT capabilities are being introduced in today’s information age”.<sup>17</sup>

Additionally, requirements for any given system are likely to evolve between the development of a system concept and the time at which the system is operationally deployed as new threats, vulnerabilities, technologies, and conditions emerge, and users adapt their understanding of their needs as system development progresses. With budgets constrained, ops tempos increasing, and requirements perpetually evolving, software development and acquisition practices must evolve in a way that facilitates faster capability deployment and flexibility in approaching system requirements.<sup>18</sup>

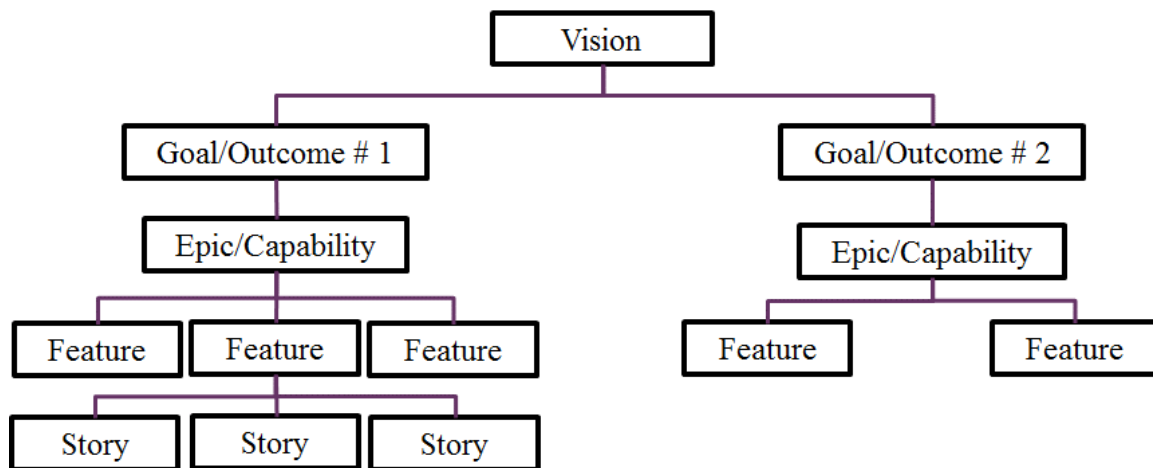
Iterative, incremental software development methodologies commonly referred to as “Agile” methods have been gaining ground in efforts throughout the DoD and federal agencies as a means to achieving these objectives for the acquisition of software-intensive systems and improving visibility into development execution to enable early detection of problems that can derail programs.<sup>18</sup>

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

Under Agile, an exhaustive set of requirements is not locked down at the start of the program. Rather, Agile development assumes that system and software requirements will evolve over time, rather than be definitized prior to system development. With Agile, high-level vision for the system is defined up front, but specific requirements are fixed at the iteration (or “Sprint”) level according to an established cadence. The development team and user representative (generally referred to as a “Product Owner”) agree to a set of requirements to accomplish during the defined time interval associated with the iteration. This serves to time-box the delivery of software: increments are completed on a regular, predictable basis. At the end of a sprint or increment, prioritized software requirements are agreed to for the next development iteration. The understanding of the user requirements evolves, guided by the high-level vision (or roadmap), as the software product continues to be developed. [Wrubel, p. 8]

While there are multiple ways for Agile project to be implemented, and there is disagreement in the terms used, the DoD’s Performance Assessment and Root Cause Analyses (PARCA) office recently issued a *Program Manger’s Guide on Agile* which clarified processes and terms.<sup>19</sup> The guide states that a program’s “High Level Vision” should be communicated to the implementing contractor as “Capabilities” and these in turn and decomposed into features. The guide states that “capabilities” refers to a group of Features that are traceable to the technical and operational requirements of the product being delivered.[McGregor, p. 6] One way to view how the high level vision statements are decomposed into capabilities, features and user “stories” is depicted in **Figure 3.0**.



**Figure 3.0.** Decomposing Customer’s Vision into Capabilities, Features and Stories

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

PARCA's PM Guidebook on Agile and Earned Value Management (EVM) provides useful term definitions and they are reproduced in Table 1.0 below. [McGregor, p. 14]

|                   |  |
|-------------------|--|
| <b>Capability</b> | Term frequently used interchangeably with <b>Epic</b> to describe a high level system functionality defined by the government to meet a specific required need. All Capabilities should have clearly defined objective technical completion criteria. Capabilities are typically found at or above the Control Account level of the WBS and are usually composed of multiple <b>Features</b> . |
| <b>Epic</b>       | See <b>Capability</b> .  |
| <b>Feature</b>    | Term used to describe a discrete system functionality defined by the government to help meet the specific completion criteria of a <b>Capability</b> . All Features should have clearly defined objective technical completion criteria. Features are typically found at the Work Package level of the WBS and can typically be completed in a single <b>Release</b> .                         |
| <b>Story</b>      | Term used to describe activities that contribute to the completion of a <b>Feature</b> and can be completed within a single <b>Sprint</b> .  |
| <b>Release</b>    | Term used to describe a concrete time box or cadence used to complete <b>Features</b> . Release duration can vary, but is typically three to six months. Many practitioners use the Release cadence as their rolling wave planning period.   |
| <b>Sprint</b>     | Term frequently used interchangeably with <b>Iteration</b> to describe a concrete time box or cadence used to complete <b>Stories</b> . Sprint duration can vary, but is typically two to four weeks.  |
| <b>Iteration</b>  | See <b>Sprint</b> .  |

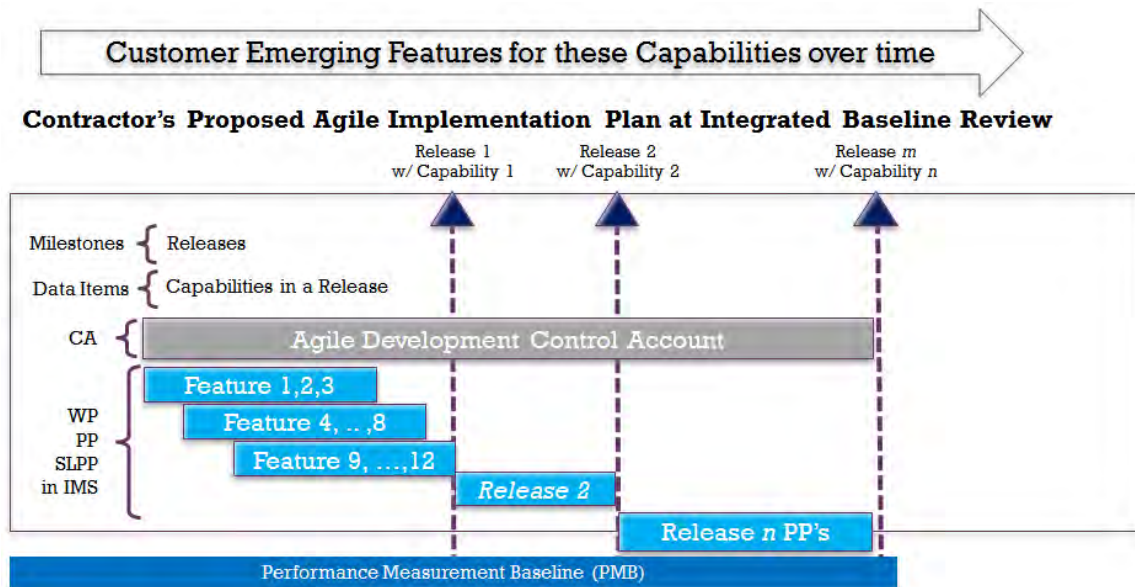
**Table 1.0.** PARCA's Agile Term Definitions

The Software Engineering Institute recommends that "...when a program pursues incremental delivery approaches, a clear high-level vision (e.g., a concept of operations, or CONOPS) is placed on contract—one that describes Agile concepts and principles and desired outcomes but does not specifically mandate Agile." [Wrubel, p. 27]

**Figure 4.0** illustrates how customers' desired capabilities are integrated into the contractors' oversight of Agile efforts after the contract is awarded. The program office either contracts for the full set of Capabilities stated in the CONOPS, or incrementally funds (and baselines) each release of capabilities or epics.

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce



**Figure 4.0** How Customer Capabilities Are Developed Using the Agile Model

### A Proposed Way to Estimate Agile Software Development Efforts

So how does DoD prepare an estimate for a major software effort that only has a high level vision statement and desired capabilities and features? To answer this question, we first must understand what we are given and what drives the amount of effort and duration of an agile software project.

Before contract award, all we know is the government CONOPS and set of desired features, but we also know from the lessons learned from the data used to drive the parametric models, that effort and duration outcomes are highly dependent on the experience and “productivity” of the team which can vary significantly from one developer to another and is dependent on the extent of common software libraries and automated development tools used. Experienced developers in the application domain may have made extensive use of libraries and modern automated tools are far more productive than teams who are not experienced or make use of existing code and automated tools.

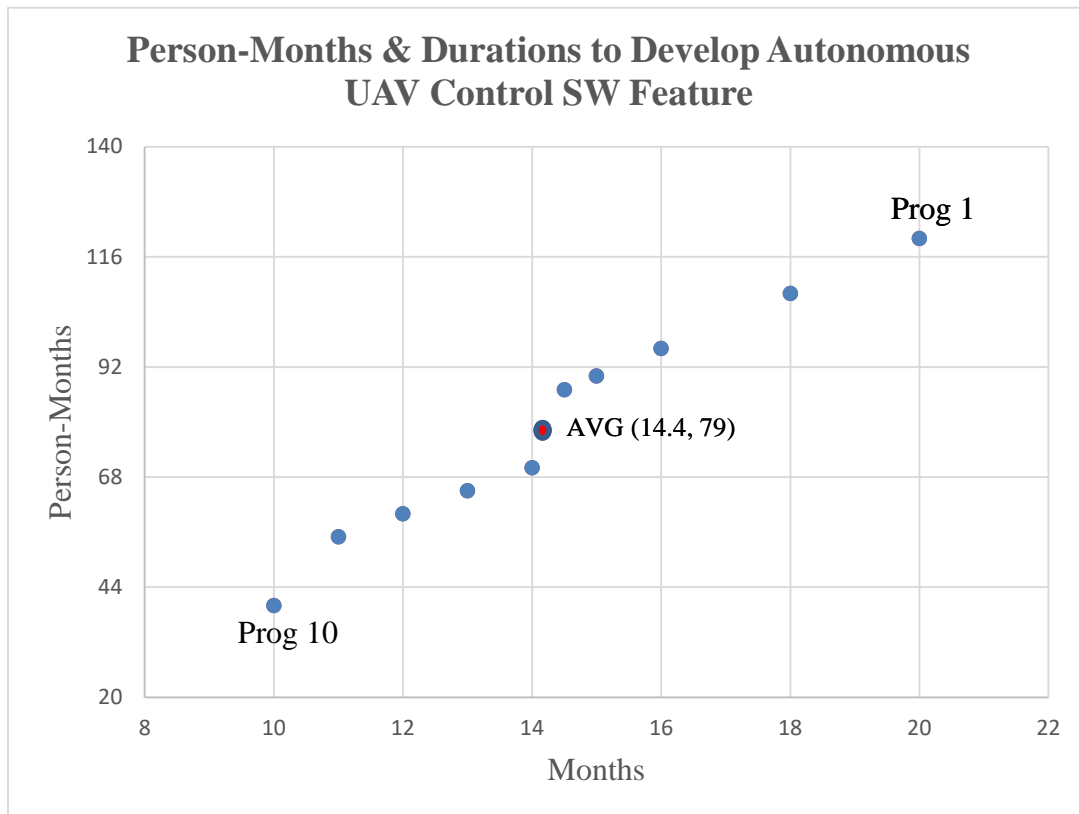
The authors propose that Agile software project can be estimated by using the historical experience of the features of historical analogies, in short, a nearest neighbor analogy method. Such historical development actuals would include feature descriptions and indications about the amount of common library use and extent of automated development tools. With the desired features described in the customer’s CONOPS, the cost estimator could find the “nearest neighbor” within the historical actuals. If the cost estimator knew more about the developer’s use of libraries and automated tools, the effort

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

and duration actuals can be selected that comes closest to matching the one in the historical database. If there is no information about the developer and their use of libraries and tools, the estimate of effort and duration can be calculated from historical means.

A notional Feature development example is shown in **Figure 5.0** for an Unmanned Air Vehicle to autonomously execute a mission received from its ground station.



**Figure 5.0. Notional Nearest Neighbor Estimating Example**

Each data point reflects the historical outcome of similar development efforts for this Feature and each contains information about the extent of library and tool automation use. **Table 2.0** reflect these data point attributes. The cost estimator would select the historical outcome pair nearest to the specifics for the project (if known) or use the means (14.4 months and 79 person-months) if not known.

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

| Prog ID | Feature                   | DevProc | Lib Use | Tool Use | Durr | PM  |
|---------|---------------------------|---------|---------|----------|------|-----|
| Prog 1  | Auto Exc<br>Pilot Mission | Agile   | None    | None     | 20   | 120 |
| Prog 2  | Auto Exc<br>Pilot Mission | Agile   | Min     | Min      | 18   | 108 |
| Prog 3  | Auto Exc<br>Pilot Mission | Agile   | Min     | Mod      | 16   | 96  |
| Prog 4  | Auto Exc<br>Pilot Mission | Agile   | Min     | Heavy    | 15   | 90  |
| Prog 5  | Auto Exc<br>Pilot Mission | Agile   | Mod     | Min      | 14.5 | 87  |
| Prog 6  | Auto Exc<br>Pilot Mission | Agile   | Mod     | Mod      | 14   | 70  |
| Prog 7  | Auto Exc<br>Pilot Mission | Agile   | Mod     | Heavy    | 13   | 65  |
| Prog 8  | Auto Exc<br>Pilot Mission | Agile   | Heavy   | Min      | 12   | 60  |
| Prog 9  | Auto Exc<br>Pilot Mission | Agile   | Heavy   | Mod      | 11   | 55  |
| Prog 10 | Auto Exc<br>Pilot Mission | Agile   | Heavy   | Heavy    | 10   | 40  |

**Table 2.0.** Data Attributes of Notional Historical Feature Data in **Figure 5.0****Data Needed to Estimate the Using the Nearest Neighbor Approach**

Historical features, their attributes, and actual effort and durations are needed to use this approach. There are many hundreds of software features that have been (and are being) developed that have different outcomes depending on the operating platform and application domains. So, some form of grouping or standardization would be needed. Fortunately, the DoD's CAPE has recently modified its Software Resource Data Report (SRDR) to capture Agile actuals and attributes on future programs. The draft SRDR contains Agile planned and actual hours by the contractor's feature identifier. Most significantly, the draft SRDR requests the development contractor to categorize the effort by operating platform and primary application domain.<sup>20</sup> These applicable platforms and primary applications are shown in **Table 3.0**.



## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

| <b>Operating Environments</b> | <b>Primary Application Domain</b> |
|-------------------------------|-----------------------------------|
| Air Vehicle-Manned            | Command and Control               |
| Air Vehicle-Unmanned          | Communication                     |
| Missile Systems               | Custom AIS Software               |
| Ordnance Systems              | Enterprise Information Systems    |
| Other                         | Enterprise Service Systems        |
| Sea Systems                   | Microcode and Firmware            |
| Space Systems                 | Mission Planning                  |
| Surface Fixed                 | Other Real Time Embedded          |
| Surface Mobile                | Process Control                   |
| Surface Portable              | Scientific and Simulation         |
| Surface Vehicle               | Signal Processing                 |
|                               | Software Tools                    |
|                               | System Software                   |
|                               | TMDE                              |
|                               | Training                          |
|                               | Vehicle Control                   |
|                               | Vehicle Payload                   |

**Table 3.0. CAPE SRDR Operating Environments and Primary Application Domains**

Within the Primary Application Domain, the CAPE has promulgated about 200 subdomains examples that will go a long way to organizing a feature-rich historical repository. **Table 4.0** depicts the possible set of data fields that would be necessary to use the nearest neighbor analogy approach to estimate features contained within a UAV illustrated in **Figure 5.0** and **Table 2.0**. Note the attribute fields of Operating Environment, Application Domain/Sub-domain are in accordance with the SRDR requirements. Also note the field called Dev Process. This is also per the SRDR and valid fields include: Agile, Hybrid-Agile, Spiral, RAD, and Waterfall. This helps qualify the historical record.

The Library and Tool Use levels are suggested additions to the SRDR that seem importation productivity differentiators. Personnel experience level would also be useful, but was omitted from the table for clarity of the concept. Also note the inclusion of the attributes called Feature Description and Feature Category. The Feature Description appears to be missing from the draft SRDR and would need to be added. The Feature Category constitutes a Feature Breakdown Structure (FBS) and would need to be



## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

developed by the estimating community over time similar to the standardized Work Breakdown Structure. This would ease the estimator effort of finding suitable analogies.

| Program Name | Contract Number | Operating Environment | Application Domain/SubDomain              | Dev Process | Library Use Level | Tool Use Level | Feature Category  | Feature Description   | Person Months | Duration |
|--------------|-----------------|-----------------------|---|-------------|-------------------|----------------|-------------------|---|---------------|----------|
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VP-Pointing, Command, & Control Interface | Agile       | Heavy             | Moderate       | Flight Control    | Accept pilot missions via web client                            |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Flight Control                         | Agile       | Heavy             | Moderate       | Flight Control    | Transmit pilot mission to UAV                                   |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Attitude Control System                | Agile       | Heavy             | Moderate       | GN&C              | Record aircraft orientation, altitude and vector data           |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Flight Control                         | Agile       | Heavy             | Moderate       | Flight Control    | Transmit aircraft orientation, altitude and heading data to GCS |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VP-Pointing, Command, & Control Interface | Agile       |                   |                |                   | Allow GCS pilot to input control commands                       |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Flight Control                         | Agile       | Heavy             | Moderate       | Flight Control    | Transmit real time pilot flight control commands to UAV         |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Flight Control                         | Agile       | Heavy             | Moderate       | Flight Control    | Execute GCS pilot control commands real time                    |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Flight Control                         | Agile       | Heavy             | Moderate       | Flight Control    | Collect & store real time camera video                          |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Flight Control                         | Agile       | Heavy             | Moderate       | Flight Control    | Transmit real time video to GCS pilot                           |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Flight Control                         | Agile       | Heavy             | Moderate       | Flight Control    | Allow pilot to view UAV real time camera                        |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Flight Control                         | Agile       | Heavy             | Moderate       | Flight Control    | Accept Pilot GCS mission commands                               |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Flight Control                         | Agile       | Heavy             | Moderate       | Flight Control    | Autonomously execute pilot mission                              |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VP-Reconnaissance Payload                 | Agile       | Heavy             | Moderate       | ISR               | Allow GCS personnel to request SAR images                       |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VP-Reconnaissance Payload                 | Agile       | Heavy             | Moderate       | ISR               | Record & store SAR image data                                   |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VP-Reconnaissance Payload                 | Agile       | Heavy             | Moderate       | ISR               | Allow GCS personnel to request SAR images                       |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VP-Reconnaissance Payload                 | Agile       | Heavy             | Moderate       | ISR               | Transmit GCS-requested SAR image data to GCS                    |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VP-Reconnaissance Payload                 | Agile       | Heavy             | Moderate       | ISR               | Allow GCS Personnel to view SAR data                            |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VP-Reconnaissance Payload                 | Agile       | Heavy             | Moderate       | ISR               | Record & store EOIR data  |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VP-Reconnaissance Payload                 | Agile       | Heavy             | Moderate       | ISR               | Transmit EOIR data to GCS                                       |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VP-Reconnaissance Payload                 | Agile       | Heavy             | Moderate       | ISR               | Allow GCS personnel to view EOIR data                           |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Executive                              | Agile       | Heavy             | Moderate       | Status Monitoring | Monitor Status of on-board systems                              |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Executive                              | Agile       | Heavy             | Moderate       | Status Monitoring | Transmit Status of on-board systems to GCS                      |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Executive                              | Agile       | Heavy             | Moderate       | Status Monitoring | Log & store mission statistics                                  |               |          |
| System Y     | N000-C-2017-yyy | Air Vehicle-Unmanned  | VC-Executive                              | Agile       | Heavy             | Moderate       | Status Monitoring | Transmit mission statistics to GCS                              |               |          |

**Table 4.0.** Sample Attributes Needed in Feature Repository for UAVs

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

### Conclusions and Recommendations

The traditional parametric estimating tools are not effective in helping a DoD analyst estimate the effort and duration of an evolutionary software project because detailed requirements are not available to generate software sizing inputs. The authors have shown that estimating a planned agile developed project can be done by using a nearest neighbor analogy technique built upon the actual development effort and durations and attributes of completed similar software features. To do this requires the following actions:

1. The DoD customers must produce a clear set of capabilities and desired features in the CONOPS.
2. Development contractors must submit the following data about each feature developed on CAPE's SRDR
  - a. Brief Feature Description;
  - b. An indication of the level of standard library and automated tool use;
  - c. Categorization of standardized Operating Environment and Domain and Subdomains;
  - d. The type of development method (Agile, Hybrid Agile, Spiral, RAD or Waterfall); and
  - e. Actual development hours and duration
3. The CAPE must verify the submitted data and make it available to the authorized estimating community, and
4. The estimating community must develop a standardized Feature Breakdown Structure from the collected software feature repository.

It is the authors' understanding that all projects are required to develop a CONOPS that describe the end-state vision, capabilities and features so item 1 above is already being done. The CAPE's SRDR already contains most of the data elements needed and CAPE is currently vetting the submitted data and making the data available to authorized estimators. Only items 2 (a) and (b) would need to be added to the SRDR. Once the data is collected, the estimating community would surely rise to the occasion to create a standardized feature breakdown structure.

## How Should We Estimate Agile Software Development Projects and What Data Do We Need?

Glen Alleman and Thomas Coonce

## References

- 
- <sup>1</sup> <https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=8e8f5bf3-f517-4e98-8c07-80bb8670a830>
  - <sup>2</sup> Jorgensen, Michel J., and Molokken, Kjetil. 2003. "A review of Studies on Expert Estimation of Software Development Effort". *International Symposium on Empirical Software Engineering*, 2003
  - <sup>3</sup> Nelson, E. A. 1966. "Management Handbook for the Estimation of Computer Programming Costs", AD-648750, Systems Development Corp
  - <sup>4</sup> Clark, Brad, and Madachy, Raymond, 2015. "Software Cost Estimation Metrics Manual for Defense Systems", Software Metrics, Inc.
  - <sup>5</sup> [https://en.wikipedia.org/wiki/Software\\_development\\_process](https://en.wikipedia.org/wiki/Software_development_process)
  - <sup>6</sup> Elliott, Geoffrey, 2004. "Global Business Information Technology: an integrated systems approach". *Pearson Education* .87
  - <sup>7</sup> <http://www.acqnotes.com/acqnote/careerfields/software-development-approaches>
  - <sup>8</sup> DoD-STD-2167A, Defense System Software Development, 4 June 1985
  - <sup>9</sup> <https://acc.dau.mil/CommunityBrowser.aspx?id=518290>
  - <sup>10</sup> Royce, Winston. August 1970 "Managing the Development of Large Software Systems," *Proceedings IEEE WESCON*, 1-9.
  - <sup>11</sup> [https://en.wikipedia.org/wiki/Waterfall\\_model](https://en.wikipedia.org/wiki/Waterfall_model)
  - <sup>12</sup> Larman, Craig and Basili, Victor. 2003. "Iterative and Incremental Development: A Brief History". *IEEE Computer (June ed.)*.
  - <sup>13</sup> [http://myprojects.kostigoff.net/methodology/development\\_models/development\\_models.htm](http://myprojects.kostigoff.net/methodology/development_models/development_models.htm)
  - <sup>14</sup> <http://www.acqnotes.com/acqnote/careerfields/software-development-approaches>
  - <sup>15</sup> <http://www.testingexcellence.com/spiral-model-sdlc/>
  - <sup>16</sup> Welby, Stephen, November 21, 2013. Deputy Assistant Secretary of Defense for Systems Engineering, "Thinking About Agile in DoD", *AFEI Agile for Government Summit*
  - <sup>17</sup> <http://www.marcorsyscom.marines.mil/Portals/105/FRO/The%20Point%20Archive/The%20Point%20January%202013.pdf.pdf>
  - <sup>18</sup> Wrubel, E., Gross, J., August 2015. "Contracting for Agile Software Development in the Department of Defense: An Introduction", *Software Engineering Institute, TECHNICAL NOTE CMU/SEI-2015-TN-006*,
  - <sup>19</sup> McGregor, John. 03 March 2016. "Agile and Earned Value Management: A Program Manager's Desk Guide", *OUSD AT&L (PARCA)*
  - <sup>20</sup> ERP SRDR Form 3026-3 dated 2017January19