A Framework and Metrics for Addressing an Agile Enterprise
(Extending the Observe-Orient-Decide-Act paradigm into Acquisition)

Pritchett Harris
The SI Organization, Inc.

Abstract. When attempting to design or specify agile solutions for an enterprise with long acquisition timeframes, both the solutions and the enterprise itself must be considered in the same framework to understand the value of time and money spent at one time and performance measured at another. A broad overview of techniques, artifacts, and measurements developed for, and used to, address Enterprise Agility is presented and discussed. Emphasis is placed on treating the distance between both ends of the traditional engineering ‘V’ in a common framework, and on ‘virtual’ feedback between operations and design or portfolio management spaces.
Introduction
Over the last several years, corporate explorations of agility, gaming, evolution, and complexity have led to our organization developing a framework for addressing and dealing with agility at the enterprise level, particularly under constrained conditions of time and resources to support multiple goals in the face of a chaotic environment. From a customer’s point of view, the problem is characterized by a slow acquisition and deployment timescale and a rapidly changing environment which tends to partially invalidate solutions by obsolescence while the acquisition process is still going on. To be effective, portfolio management approaches therefore need to focus on understanding the value returned when deploying the program outputs into an environment that does not match the developmental assumptions and is changing faster than the next acquisition program can react, not just on the immediate affordability (fitting within a budget profile) of a collection of development and improvement programs.
These explorations included identifying some key enterprise level metrics and developing some example requirements to drive agility or resiliency into systems under design. Beyond the basic INCOSE Resilient Systems Working Group definition, ‘resiliency’ in this context is ‘agility in the face of adversity’. This approach represents an essentially hybrid of agile SYSTEMS ENGINEERING and engineering AGILE SYSTEMS in order to apply Boyd’s Observe-Orient-Decide-Act cycle in a combined acquisition, development, and operations (an enterprise) environment.
This paper provides a high level overview of the framework and metrics with the intention of exposing or touching on several topics which can be further explored at later times. These are:

- **Timescales & Motivations** - Define ‘Enterprise’ and the need for ‘agility’
- **Framework** – Identify the conceptual relationships between various artifacts, techniques and metrics
- **Gaming** – Provide tomorrow’s feedback on today’s decisions
- **Portfolio Management (PfM)** – Link future value to current investment
- **Connected models** – Link models which represent different points in time
- **Systems and System Requirements** – Understand the units or quanta of change
- **Enterprise Requirements and Key Metrics example** – Provide example inherited top level requirement and its measurement
- **Defining Enterprise level Requirements** – Provide general classes of inherited top (enterprise) level requirements
- **Observables and Success Metric** – Define an enterprise level formula for Agility
- **Example results** – Apply the framework to a specific challenge
- **Governance** – Enable the decision making process

**Timescales & Motivations**
An enterprise has aspects of continuous operations cross multiple physical and temporal scales; i.e., development and operations, years and days. At all levels, the enterprise competes with its environment, performing Boyd’s Observe, Orient, Decide, Act (O-O-D-A) loop to maximize the satisfaction of enterprise goals by gathering and analyzing information and reconfiguring assets to respond to an ever changing environment. This results in the definition of Enterprise given:
**Enterprise:** An entity comprised of one or more organizations, engaged in a mission requiring the **development, sustainment, and projection** of supporting capabilities in a changing environment.

This definition deals with the persistence of organizational imperatives (the collective) and the span of the ‘Lifecycle V’, starting with conceptual design, through development, and ending with operations plus lessons learned for the next cycle (the individual). It recognizes that current enterprise model approaches generally focus either on developmental-space decision making (portfolio trades between cost and schedule, with limited impact on future performance) or operational behavior (trades between corrections and improvements to current performance), either through modeling and simulation, or ‘exercise’ type activities. However, there are still enterprise scale questions that cannot be addressed by either of these two approaches alone, questions that are typically answered by the 20/20 hindsight of operations viewing past decisions in light of the present environment because individuals at each end of the ‘Developmental V’ tend to view the process from paradoxically different points of view.

Agile development addresses part of this ‘Requirements Paradox’ – the need to develop in a static environment, yet operate in a dynamic one. Current agile software approaches create short-term artificial static developmental environments inside the dynamic rate of change.

A broader enterprise approach that ‘looks into the future’ is desired. Such an approach would augment a static ‘specified environment’ that does not necessarily reflect the actual future operational environment, with a ‘pseudo’ or ‘virtual’ environment in which the system(s) under development are expected to operate and succeed. Systems within such an enterprise might have to be designed on principals to operate across a range of future cases, rather than tightly specified design rules. Figure 1 presents the Framework based on the definition of Enterprise presented above. The left side is focused on the ‘organizational (slower) side’ of agility, while the right side addresses the ‘operational reaction space’ where operations personnel and systems are ‘doing what they can with what they have’.

The Framework is intended to represent multiple concepts that connect the two sides of the ‘Life-cycle V’, so that the agility of the enterprise itself can be addressed via both operational responsiveness, and developmental responsiveness (speed), including injection of features that improve agility or resilience. There are multiple places within the Framework where an ‘artifact’ or approach spans both sides of the ‘time based’ point of view and/or is reflected in both frames.

The ‘development side’ evaluates enterprise performance and procures and/or delivers assets (IT equipment, capitol assets, systems, staff, etc.) that support daily or continuous operations. In United States Department of Defense (DoD) vernacular, the ‘development side’ could modify or create anything in the doctrine, organization, training, materiel, leadership and education, personnel and facilities (DOTMLPF) spectrum. Operations has to deal with executing the ‘mission’ in the face of a changing environment (weather, technology, economics, markets, combat, etc.) using the assets and mindset they have on hand. The Framework is intended to deal with the ‘lag’ between acquisition decisions and operations in a later (unanticipated) environment.
Elements of the Framework

The Framework is a collection of adaptable corporate processes, reusable artifacts, and mindset cues that can be used individually or collectively depending on the customer’s needs and scope of the engagement. In almost all cases the underlying assumption is that regardless of the scope of the engagement, the customer has an enterprise (per the definition above) encompassing multiple projects which can benefit from an agile enterprise approach. The elements are described from top to bottom referencing Figure 1.

**Gaming.** *Provide tomorrow’s feedback on today’s decisions.* The ‘Gaming & Simulation’ block at the top of Figure 1 is enlarged in Figure 2 to show a conceptual loop between ‘virtual’
versions of development (representing a programmatic decision-making space) and operations (represented by an immersive 3D first person shooter type environment).

![Figure 2 - Connection of cost/schedule/performance trade space with 'virtual battle-space' to determine utility of options traded](image)

Traditional modeling and simulation has been augmented by immersive and interactive simulations designed to close the decision loop in development space while adding experiential data and understanding to the ‘orient’ phase of the O-O-D-A loop, which can only be gained by exercising the systems in an ‘operational’ environment. This ‘feedback’ of experience into the ‘orient-decide’ activity was demonstrated by a proof-of-concept internal research and development (IRAD) task (executed in the summer/fall of 2009) that combined three distinct IRAD capabilities into a decision making feedback loop. The approach used a value model to rank proposed solution alternatives for using unmanned aerial vehicles (UAVs) to monitor a battle space and report enemy positions (with errors) to a blue force mission commander. The UAV options each included cost, schedule, and predicted technical performance values (i.e. resolution and latency) and the options were ranked by a stakeholder (developers and operators) derived value structure. The experiment was designed to determine if the performance assumed for various choices provided sufficient operational utility to accomplish some hostage rescue and direct combat scenarios. This approach allows correlation or calibration between the stakeholder value model and the actual measures of effectiveness (MOE) as measured by the game statistics. The approach also allows identification of key parameters that improved statistical mission success to be identified in the UAV architecture and tracked in development. It also provided the opportunity for the ‘opposing team’ to learn, to surprise, or to respond to innovations on the ‘friendly’ side, indicating how long military or competitive advantage might be maintained and influencing deployment and operational doctrine.

This is inherently an approach to create an immersive environment for design collaboration at
the very beginning of the SE lifecycle. The intent is to perform multiple iterations on the objective behavior of a system and identify ‘evolutionary updates’ to a concept before the system actually starts development so the first development spiral might be considered ‘version 4.0’ instead of ‘version 1.0’. This design point could then be released for development while testing spirals continue in the background and as multiple systems are added to the virtual environment to understand interoperability of future systems-of-systems.

**Portfolio Management.** *Link future value to current investment.* Moving down the framework, (see Figure 3) the programmatic decision-making space embodied in ‘gaming’ is fully represented by a Portfolio Management (PfM) approach that uses a genetic algorithm to adjust developmental tasks and schedules to search out and identify maximum value solutions in a very large Pareto space of options, represented by the left-hand ‘faster’ and ‘cheaper’ ovals (in dark blue). When used alone, this approach assumes that future performance (the right-hand dark blue ‘better’ oval) is fixed, but in reality it is not, and it is measured at a different time. Gaming and Simulation are used to pre-estimate that performance and feed it back into the design process, but it is actually measured in operations.

![Figure 3: Portfolio Management (PfM) in context](image)

The concept of “better” in development space represents the ability to better estimate cost, schedule and performance. Cost and schedule are dealt with in programmatic space, while the domain and range of performance involves gaming and/or analysis. The inverse faster-better-cheaper triangle (shown in the figure in light blue) represents, on the operations side, the ability of the operations user to get ‘better’ quickly and cheaply (agilely), and as mentioned earlier, to better estimate outcomes during development.
**Connected Models.** *Link models that represent different points in time.* The diagonally structured icon centered in the Framework is shown in Figure 4. It represents an enterprise following Boyd’s O-O-D-A model, bounded by a guiding value model (upper left) and the external environment (lower right). The black dots in the model represent ‘is-related-to’ connections between differing models or data-bases. Any intersection can have a connection. This model promotes the conceptual and actual linking of sub-models (cost, schedule, performance, etc.) within an enterprise level framework, including the construction of a predictive performance capability that supports both traditional performance modeling and more free-form gaming. This is the construct used to support the ‘gaming feedback’ described earlier in Figure 2 and applies the O-O-D-A loop against both operations and SE, effectively applying it at the enterprise level. When used with portfolio management, the approach attempts to optimize the enterprise with respect to the value model’s goals and objectives. This may or may not maximize performance or any other parameter but rather allow them to reach an equilibrium based on optimal ‘value’. The $N^2$ matrix is extensible for additional domains such as risk, and for recursive levels of detail.

**Systems and System Requirements.** *Understand the units or quanta of change.* Within the structure above (particularly performance modeling and architecture) the construct used to model individual systems, their driving activities and their potential re-arrangement or improvement, is shown in Figure 5. This figure is nominally referred to as a ‘kill chain’ within the DoD, but can be thought of as general ‘mission chain’ of activities (across the top) and the systems and networks which enable them. This construct allows for visualizing, rearranging, improving or re-sequencing systems and/or activities which are used to create capabilities and
accomplish missions. This is the level at which requirements are mapped to systems or networks, and modification via systems engineering, gaming, genetic algorithms or other means occurs. Figure 5 shows a framework that has been used to perform gap analysis of an enterprise where there were multiple (somewhat fungible) systems available to perform each step in an activity chain (e.g. Find, Fix, Track, Target...) representing a capability; the color dots represent how well each system performed its step of the mission under various specified conditions (e.g. daytime or night-time).

This element of the Framework presents the opportunity to identify not only performance, but **time-to-reconfigure** the individual systems (including changes to performance and/or networks) or the overall sequence. Reconfiguration can occur in operations (or a performance/gaming model of operations) if it has been designed in, or it must be done in acquisition/development, which implies a different time scale. Use of the same framework in both development and operations promotes an enterprise level approach and a common understanding from the point of view of users on both side of the ‘life-cycle V’.

“Time-to-reconfigure” and “time-to-recover” were identified as critical success factors in the O-O-D-A response process, where the environment is not static but dynamic and possibly adaptive and hostile. For example, Find, Fix, Track, **Target** could be updated to Find, Fix, Track, **Rescue** if the information from the track activity could be forwarded to any system associated with the rescue activity capable of interpreting the publicly defined data the activities have in common *and* it could be done in a relevant time frame. This implies that adaptive/flexible features must be injected during the development process in order to decrease the likelihood of needing additional development cycles for each new challenge. The frequency of need for development cycles is a Key Performance Parameter (KPP) or metric of enterprise agility.

**Enterprise Requirements and Key Metrics.** A *key example inherited top level requirement and its measurement*. Using the “time-to-reconfigure” and “time-to-recover” concept above spawns a small but significant class of Enterprise Requirements that deal with the rate-of-change or rate-of-adaptation to the problem environment. This includes metrics for success (number and span of potential problems the enterprise can handle), time to reconfigure, etc., which can be identified as metrics and used as KPPs where the value of improving the KPP at the enterprise level can be decomposed down to individual or collective improvements to systems or components. This is illustrated in Figures 6 through 8. Figures 6 and 7 show an enterprise (or system) response to an unanticipated change in mission need (system failure or environmental change) where performance is recovered incrementally over time, and the best and worst possible recovery times. Although developed independently, this is similar to Alberts’ description of manifest agility except the best recovery profile (in time and performance) shows the maximum agility via ‘recovered area under the curve’ while manifest

---

1 Enterprise components include developmental activity, so a Time-to-Reconfigure KPP may be measured across Operations and Development. Development is also where the ‘time to reconfigure’ must be designed into both Development and Operations.
agility represents unrecoverable loss or ‘area above the curve’. It is also understood that ‘anticipation’ can initiate responses before that problem actually arrives, but when the problem is not known a priori the only agile feature which can be identified is ‘reaction time’ (time to change or recover).

Figure 6 – Conceptual Response to Change in Mission Need

Figure 7 – Best and Worst Case Response

Figure 8 shows response as a function of time for two differing architectures (possibly comprised of one stovepipe system each, many systems or the same systems with differing reconfiguration approaches). Architecture A in the figure is the only one of the two that achieves any response in the time frame needed to address the problem (which is not known a priori). Architecture B achieves a higher performance, but at a later time. Current architecture evaluation approaches do not measure success in this axis, while an enterprise must. If the example in Figure 8 were two systems within a single architecture then system A responds to the stated problem, while system B might have the better response to a different problem. The outcome of this approach is to select performance parameters that can be both measured and then improved across an aggregated (time varying) problem space using the enterprise’s portfolio of systems. Developmental choices must be made with an eye towards improving these metrics (See Figure 9) while constrained by a traditional cost/schedule/performance space. These metrics are measured by continually comparing a virtual or model enterprise to a set of known and unknown problems to estimate reaction against possible futures.

In the real world the ‘problem space’ (known and unknown) is created by the ‘bad guys’ and the enterprise designers should be cautious about relying on their own world view to make up examples. Use of intelligent, hostile, challenge designers is recommended. Gaming and virtual worlds offer significant opportunities to harvest ‘new challenges’.
Changing the frame of reference to include ‘response time’ is based on John Boyd’s seminal paper *Destruction and Creation*. The Agile Enterprise concept extends Boyd’s Observe, Orient, Decide, Act (O-O-D-A) loop to include both development and operations in reacting to a changing environment. An Enterprise:

- Observes itself and its environment – it’s problem space;
- Orientates itself to identify how the environment is stressing the enterprise;
- Decides how to deal with changes in the overall environment, and;
- Acts to implement systems that can deal with the environment and be adapted as the environment changes.

This is particularly important because the development time is typically longer than the rate-of-change in the problem space (the environment). The key feature is to consider the development and operations aspects of an Enterprise in the same framework, so that the detection of changes in the environment is understood to be part of both the operations and the development aspects of the enterprise and its management. This implies a ‘biological’ and ‘evolutionary’ approach to continuous improvement of the enterprise’s O-O-D-A loop.

**Defining Enterprise level Requirements.** *General classes of inherited top (enterprise) level requirements.* From a corporate requirements perspective the Framework treats the enterprise level as a tier across or above system level, although there are some enterprise level definitions that drive individual operations systems (and business systems for that matter). All systems and processes would inherit the higher level enterprise requirements. Some examples are shown below:

What is the definition of System and System Requirement?
- Associated with a defined Enterprise Activity or internal support function (fits the enterprise defined framework – see Figure 5).
- Conforms to published data I/O Standards (including fitting an Envisioned State Reference Architecture – or similar structure with multiple ‘OSI-like-layers’ of functionality)
- Task-able (though not necessarily in multiple instantiations if a hardware system)

What is the definition of Enterprise Requirement or KPP?
- Measurement (and improvement) of overall enterprise reaction time. Maximize ‘area-under-the-curve’ of performance against dynamic challenges – not just single system performance against a static challenge
- Being self-aware (pre-Observing) what the enterprise is capable of (in terms of the ability to react to problems (known and unknown) before the need arises. [modeling, simulation, gaming]
- Interoperability based on a Concept of Operations (CONOPS) for reconfiguration and adaptation.
- Defines time-persistent Activities, Info exchanges, Data exchanges. System info boundaries (to be used by individual systems) unique to the enterprise span of control. There may be ‘inherited’ definitions from adjacent/higher tiers.
- Defines the ‘skeleton’ of the enterprise including definition of ‘low level layers’ that support all capabilities/activities/systems. (The infrastructure that ‘systems’ will plug into.) This may include infrastructure definitions shared with partner agencies.
Observables and Success Metric. Defining an enterprise level formula for Agility. The primary observable for an enterprise interacting with its environment is performance against the environment and rate of change of both performance and environment. This can be thought of as the ‘impulse response’ of the enterprise to changes in environment. As in real life, it is unlikely that the enterprise will ever be able to come to equilibrium with its environment long enough for a proper measurement of its response to a change in forcing function. But, since some of the goals of enterprise level systems engineering should be improved response time and ‘predictability’ of behavior, constructing a ‘model’ or ‘virtual’ enterprise that accurately emulates (or predicts) the behavior of the real enterprise is a good beginning, though it is not always a client priority. Such a model should be tuned to match observed behavior as a starting point to modifying that behavior. The reader is referred to earlier work on creating a value driven Cost/Schedule/Performance model that interacts with both real and virtual inputs.

The network diagram shown in Figure 4 attempts to address the ‘impulse response’ problem by creating a ‘virtual operations’ space that enables estimating performance against possible future environmental challenges. Challenges presented in the ‘virtual environment’ are continuously addressed in the ‘design cycle’ prior (hopefully) to their actual occurrence, with solution choices being driven by a value model controlled by the enterprise owner. The frequency of these changes in input and the adaptation time (the red queen time) of both the enterprise and environment should be approximated by the feedback of the ‘enterprise network’ shown in the figure. The Value Model (VM) box ‘is related to’ (it measures) the environment in the generic case to determine the value of forcing functions that the enterprise responds to. The VM is assumed to be able to measure values within any other network node so in this minimum case the VM measures the difference between inputs to the enterprise and the enterprise’s reactions. The VM is assumed to be ‘input-only’, measuring points or parameters across the enterprise and helping decision makers give direction. The actions and directions associated with a decision are not shown flowing back out across the enterprise. Other nodes can contain relationships that flow ‘both ways’ (e.g. enterprise and environment interact in both directions).

This approach implies high value in the ability to measure enterprise change with respect to its environment to gauge its success. Therefore a successful enterprise must not just have capability greater than environment, but meet this criterion for rate of change (Eqn 1):

\[
\frac{\delta \text{ ENTERPRISE CAPABILITY}}{\delta t} > \frac{\delta \text{ ENVIRONMENT}}{\delta t}
\]

This gives us the ability to create models and measurable metrics that can be used to drive agility requirements into the customer’s enterprise. Enterprise capability is defined using the structure in Figure 5 and all supporting layers. How quickly can it be changed or re-sequenced? In some cases the time includes the ‘developmental decision and acquisition process.’ How fast does it need to be? (How many enterprises actually measure the environmental forcing function?)
Example results. Applying the framework to a specific challenge. Applying Equation 1 and/or maximizing the area under the curve in Figure 9 as a driving imperative yields an example cloud resiliency requirement:

“Be recursively self-similar (fractal), regardless of scale”

This requirement was developed for a customer architecting their cloud and is intended to promote the fastest recovery time from disruption (resiliency) by allowing isolated instances of a cloud to continue performing their core cloud-like functions when the some portion of the cloud or the network has been attacked or damaged. The requirement has an associated “Perform auto-aggregation and auto-recovery from disaggregation” requirement to maximize the ‘value’ produced by the ‘undamaged’ portion of the cloud (thus maximizing the area under the curve in Figure 9). In this case the cloud would be performing its own O-O-D-A loop to track and understand the state of its assets and if divided into two or more pieces, each instance would perform the same behavior.

Governance. Enabling the decision making process. Finally, to enable many of the processes or approaches outlined by the framework, an enterprise level governance process (the primary decide function with some Observe and Orient features) must not throttle the response time of the enterprise. The structure and ‘business rhythm’ of the governance process must be tuned to enable the act phase of the decision cycle for both operational responses and developmental ones. Within governance, Systems Engineering is also the throttle for designing in agile features based on the rest of the framework described. Governance considers the change options developed by SE (the rate of evolutionary change or mutation) and considers the cost of the changes today versus the value returned tomorrow.

Conclusions

The Framework (Figure 1 - reshown here as Figure 11) grew organically based on dealing with customer systems and issues which grew over time to become ‘Enterprises’, where the enterprise response time was interfering (or beginning to interfere) with the customer’s ability to project capabilities that were relevant to their stated mission and the environment. The rapid change in environmental threats and conditions (military, political, intelligence, business, etc.) exposed problems with the complexity, rigidity, and governance of large enterprises. In essence [Equation 1] had been violated. The Framework is our corporate approach to helping ourselves and our customers deal with these concepts.

Future effort to improve the framework may include:

- Using genetic algorithms which now only adjust cost and schedule to adjust capabilities (system features or entire architectures) while still fitting in cost and schedule constraints (i.e. capability level portfolio management).
- Introducing budget and schedule uncertainty into the above process to drive genetic selection of ‘budget agile’ enterprises which can sustain capability deployment in funding risk environment.
Figure 11. Corporate Framework for Agility
References


iv) The author, and the SI Organization’s Enterprise Integration Community of Practice


Biography

Pritchett ‘Chett’ Harris is a principal Systems Engineer with The SI Organization, Inc. with 28 years of experience starting with satellite and payload integration, verification, test and operations. He has been lead or chief architect on multiple programs and proposals. He graduated from Lehigh University with a BS in Engineering Physics and serves as the SI’s chief architect, providing guidance and direction for multiple customers on architecture, agility and clouds.