American International Journal of Computer Science and Information Technology
Volume.7, Number 3;July-September, 2022;
ISSN: 2837-388X| Impact Factor:
https://zapjournals.com/Journals/index.php/aijcsit
Published By: Zendo Academic Publishing

SWIFT SOLUTIONS: URGENT SOFTWARE DEVELOPMENT STRATEGIES

Dr. Charles W. Butler[1]and Scott Wilkinsin[2]

Article Info

Keywords: Urgent Software Development, Resilience, Agility, Innovation, Operation Warp SpeedIntroduction

Abstract

In the ever-evolving landscape of top-performing organizations, urgency has emerged as a transformative force, reshaping responses to dynamic business needs. The crucible of the COVID-19 pandemic has underscored the criticality of resilience, agility, and innovation. Resilience stands as a linchpin, enabling organizations to uphold acceptable service levels amidst severe disruptions to vital business services and processes. Agility, a complementary trait, is imperative for swift and adept responses to both opportunities and disruptions. Meanwhile, innovation entails leveraging Information Technology (IT) in novel ways to foster greater organizational efficiency and enhance alignment between business objectives and technological ventures.

However, a fourth indispensable characteristic has crystallized in the realm of software development – urgency. It embodies the swiftness with which an unforeseen trigger exerts a substantial influence on the business, promptly necessitating action to address the trigger and modulate its repercussions and outcomes. The crucible of COVID-19 birthed the imperative of urgent software development out of sheer necessity.

In the current landscape, organizations are confronted with an amplified urgency to vie in an economic terrain profoundly shaped by a global pandemic. The Operation Warp Speed (OWS) initiative stands as a testament to the potency of urgency, leveraging the collective resources of the U.S. federal government and private sector to expedite the testing, production, development, and distribution of safe and efficacious vaccines, therapeutics, and diagnostics to combat COVID-19, achieving substantial milestones by January 2021. Akin urgency is palpable among companies operating and competing in today's economy.

[1] Colorado State University
[2] FXF Reliability and Environments, FedEx

Introduction

In top-performing organizations, urgency is impactingthe historical response to dynamic business requirements. If COVID-19 has taught us anything, it is the importance of resilience, agility, and innovation. Resilience is required for an organization to maintain acceptable service levels through, and beyond, severe disruptions to its critical business services and processes. Agility is essential torespond efficiently to opportunities and disruptions. Innovationmeans using information technology (IT) in new ways to create a more efficient organization and improve alignment between business goals and technology initiatives. As important as these concepts are, a fourth characteristic has emerged in software development – urgency.Urgency is the time it takes an unexpected trigger to have a significant impact on business andestablish a priority forcing action to respond to the trigger and affect the impact and outcomes. During COVID-19, urgent software development was founded out of "the mother of necessity."

Organizations today have a renewed imperative to compete in an economic environment driven by a global pandemic. Operation Warp Speed (OWS) used the resources of the federal government and the U.S. private sector to accelerate the testing, supply, development, and distribution of safe and effective vaccines, therapeutics, and diagnostics to counter COVID-19 by January 2021.(Operation Warp Speed, 2021)A similar urgency is realized by companies operating and competing in today's economy.

Objectives

Urgent software development (USD) focuses on the delivery of software solutions in a continuous manner outside of a traditional cyclical approach for software projects. USD addresses projects requiring immediate deployment but not a crisis response to an emergency state.

USD relies solely on agile discipline to implement continuous timelines and is distinguished from traditional development that is scheduled in line with business cycles such as planning, accounting, financial, manufacturing, and service cycles.

This article will:

1. Describe the ecosystem and components of urgent software development.
2. Define a maturity model for companies to follow.
3. Outline architectural implementation for ecosystem nodes.

Urgent Software Development Comes to the Forefront

Urgent software development is an ecosystem that enables the development, implementation, and operation of business solutions in response to unknown events and triggers within unprecedented, short duration deployment (timing) windows. In urgent-capable organizations, USD replaces methodical planning and reactionary, putting-outfires, panic culture with leadership, governance, methodologies, and tools and practices enabling effective response to urgent requirements.

What role does USD play in top-performing, urgent organizations? To begin with, it lowers or eliminates constraints and barriers associated with traditional software development. With the right management, methodologies, tools and techniques, way of working, IT services, and test-driven environments, organizations move rapidly, instead of pausing, to overcome imperfections in software development. Ultimately, an organization aims to realize the "urgent" level of USD maturity, where software development merges with a continuous time context. This level of software development requires a succession of changes in how the organization executes software development and how it applies resources to achieve its goals.

To achieve urgent software development capability, several characteristics define the organizational environment needed for continuous development. Some of these characteristics are fundamental prerequisites, and others drive urgency. USD organizations master four critical characteristics:

1. Priority: able to differentiate among a portfolio of installed applications and software development initiatives. Not all applications and software initiatives are urgent ones.
2. Awareness: able to assimilate the most relevant status of software assets. Software development status can vary from non-urgent to urgent.
3. Correctability: able to synthesize and establish critical aspects of software development information to know accurate software status.
4. Integration: able to synthesize software development information within an enterprise architecture and delineate cross functional linkages and acknowledge the architectural dependencies.

As important as the fundamental characteristics are, they alone are insufficient for urgency. Urgent-performing organizations excel at several additional characteristics:

1. Changing: able to integrate time as the critical characteristic of operational execution. Time drives when and how the state of business operation ebbs and flows.
2. Creating: able to alter the status quo creating an environment receptive to innovation and change.
3. Predicting: able to prepare in advance of anticipated events and triggers by analyzing market, business, and technical outcomes. This analysis yields proactive redefinition of new goals, strategies, and tactics.
4. Empowering: able to give employees authority to make decisions and act in the best interest of the organization.

Urgent software development is constructed from new and emerging methodologies, policies, procedures, and practices. To effectively execute urgent software development, a complete ecosystem must first be in place so that an individual USD node does not become a roadblock. Another aspect of urgent software development is to ensure all the nodes are in place prior to it being a necessity because it is not something an organization wants to piece together on the fly. This premise introduces a fifth critical factor to USD – preparation. Preparation addresses nonproduction software versus production software because production software needs a “superhighway” pathway to ensure compliance standards are met. A pipeline should be implemented to ensure the integrity, performance and accountability of work conducted at light speed with high levels of risks to the business. Urgent software development is a reimagination of development and testing via test-driven environments which finally shed waterfall concepts to support urgency.

How Urgent Is Urgent?

There are different types of software development projects, and projects can be classified according to software development urgency. Many software development projects are elective. Some planned software development is scheduled within an operational window that is cyclical with the organization’s financial and operational business cycle. These projects occur within a planned time that suits the organization, business units, stakeholders, and customers.

- Routine: performed as part of a regular operation rather than for a special reason or trigger; little or no time imperative.
- Expedited: when a project targets early intervention and implementation for a condition that is not an immediate threat to business operation.
- Urgent: project Intervention addresses acute impact on business operation and is targeted to relieve the risk of distressed organization operation.
- Immediate: when an operation-saving project is required, and software development is performed simultaneous with intervention. This classification of urgency normally occurs within hoursof a decision to respond to a trigger.

In today's business environment, the trigger for a software development project can originate totally from internal and external sources such as the COVID-19 pandemic. In these cases, urgency classification is immediate because projects are business and operation-saving.Usingurgency classification, the urgency of software development intervention can be defined, and urgent software development nodes can be used by managers to:

- Manage software development projects in daily or hourly dimensions.
- Analyze software development performance (key process indicators) by checking if projects are being completed within the appropriate time frame for its urgency classification.
- Ensure that actual application performance meets or exceeds goals.
- Verify that governance is being adhered to and that leadership is conducted "in continuous" time units only when appropriate.
- Organize and develop the USD nodes by taking appropriate corrective actions according to urgent classifications for development projects.

Urgent Software Development Ecosystem

As shown in Figure 1, urgent software development is a mine field of organizational and software development concepts. There are six nodes to an urgent software development ecosystem. The ecosystem is composed of organizationaland technical nodes and is a hybrid architecture of on and off premise resources. Urgent software development finds a home in an ecosystem of interconnected nodes, formed by the interaction of a community of organizational, managerial, and software development concepts.

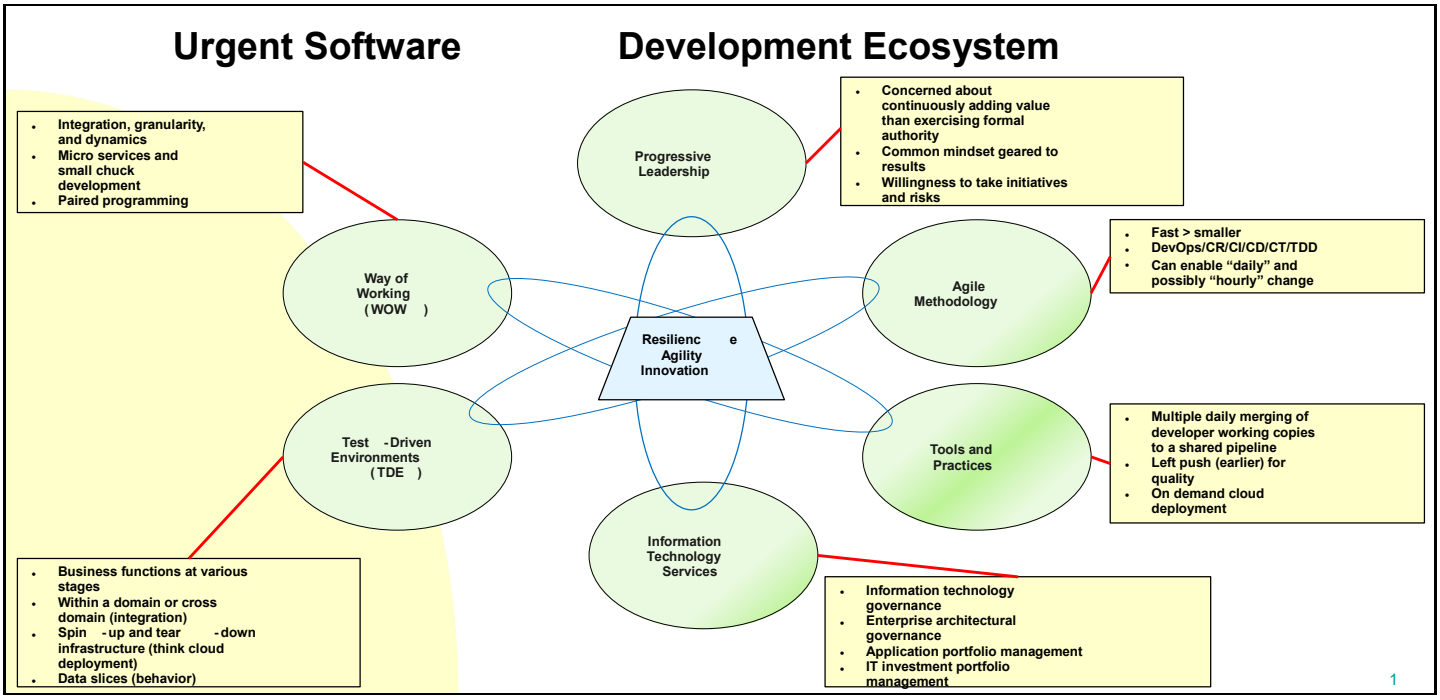

Figure 1 Urgent Software Development Ecosystem

USD Nodes

The requirements of the Urgent Software Development Ecosystem (USDE) are served by the technologies, methodologies, processes, and practices represented by the following nodes:

- Progressive leadership
- Iterative methodology
- Tools and techniques
- Information technology services

- Test-driven environments (TDE)
- Way of working (WOW)

Progressive Leadership

At every level in the organization, managers must be "all in" when it comes to Urgent Software Development and must establish a progressive culture within both the business and ITcommunities. This requirement is especially true for senior leadership, as they drive the organization to think and act differently. Management levels must embrace each of the nodes completely and leave behind traditional, legacy ways of leading (and thinking). It takes discipline and courage to adopt USD practices since doing so requires ending comfortable, familiar processes in favor of radically new ways of leading. Senior leadership adapts and supports dramatically different ways of working and, at the same time, drive radically different expectations across the organization. Management must adopt a mindset geared to results that continuously adds value. It must foster higher risk taking and a climate of innovation and thinking outside the box. (Goncalves, 2015)

Methodology, Tools, and Practices

Traditional software development methodologies are not conducive to "urgency." An agile approach is a critical success factor to establish cadence, flexibility, and response to continuously evolving requirements. Most organizations today have adopted and are comfortable with a "reactive reliability model", one in which reliability is defined as a series of reactions to certain trigger events. Alerts, monitors, or system crashes trigger reactive responses by teams of technically astute resources to uncover the source of the issue, diagnose the underlying problem, suggest redemptive responses, and implement a recommended response.

Urgent Software Development requires a "proactive and predictive" model to complement the reactive nature of traditional reliability models. Continuous reliability (CR) acts as a foundation that injects quality into the monitoring process, mitigates risk, and provides the structure necessary to respond quickly to operational-level triggers. Continuous reliability promotes quality solutions by using traditional support functions including problem diagnostics, efficiency tuning, real-time monitoring, and root cause analysis by implementing reactive monitoring.

Proactive discovery creates and deploys multiple strategies and tools to gauge the health and viability of application-based sets and functional supersets of applications. Critical application sets identified through internal analysis (applications that have affinity with each other and function as a cohesive group) serve as the focus for this approach, and health dashboards will indicate a relative health score for each of the defined focus areas. The purpose of this approach is to identify issues, bottlenecks, and failure points prior to them becoming production problems. The key tools for this practice are:

- Superset identification: All test scenarios (from traditional unit-based tests to performance tests) are identified. The test scenarios include all the applications executed across the identified testing scenarios (single applications, multiple applications within the same domain, multiple applications that cross domains), the necessary supporting infrastructure that these tests require for execution, and finally, the data that will be consumed by these tests (both positive, negative, and initial states of being).
- Reliability bots: A reliability bot is a proactive toolset created by the reliability team. Proactive, continuous reliability models depend upon tools sets that are customized to measure, model, and alert according to logical business scenarios that are tied to many of the superset definitions described previously. These "bots" are proactively probing groups of applications that have an affinity with each other and provide a grouped, logical health measurement used to monitor, diagnose, and predict the operating metrics associated with the superset.

- Health dashboards: Health dashboards further display the health of bots, measurements, and other diagnostic readings into "at-a-glance" decision metrics that can incorporate thresholds of acuity, trigger response systems, and alert technical teams of both current and pending issues.
- Site reliability engineers (SRE):SRE's are application subject matter experts that meet regularly to gauge not only the health of their respective applications within the aggregate superset designation but also diagnose and suggest fixes, upgrades, and architectural modifications (along with the project reliability engineers). This process occurs on a regular cadence as opposed to ad hoc response to immediate issues.

Continuous reliability employs a process of "predictive actions" designed to place load, stress and controlled trauma on application sets identified as critical to the overall reliability of production applications. Reliability teams use "rely-controlled environments "that exist behind the firewall (both static and virtual (spin-ups)) to predict potential trouble spots and create solutions that are much more resistant to future problems. This approach employs the following practices:

- Superset classification of high-risk, high-volume sets: The combined team of site reliability engineers (SRE) and reliability engineers work with their development counterparts to identify "potential and identifiable problem areas"in both application architecture and application execution. These areas may span across domains and form their own unique focus domains, each worthy of concentrated focus. Specific supersets that encompass these focus areas are identified and treated as an aggregate in scrutiny, testing and measurement.
- Break connections, expectations, stability levels, pre-conceived notions: Continuous reliability relies on the process of predictive execution in that special supersets may become candidates for reliability targeting to push levels of expectation, performance, and legacy inertia past accepted extremes. This practice allows technical engineers the opportunity to apply their expertise to push existing superset aggregates past breaking points and beyond performance acceptance levels to suggest business process and architectural modifications.
- Spin-up/spin-down environments: Test-driven environments incorporate cloud (specifically public cloud) infrastructure flexibility, essentially eliminating the need for continual involvement by (development teams) in ongoing hardware, software-support and other infrastructure-related activities. Environments are "spun-up" with each software build according to predefined environment variables and process, and then "spun-down" again when execution concludes.
- Continuous integration threshold gates during the build process: Quality standards and thresholds are used as a gate to "fail builds" during the CI/CT process. Sometimes, these gates are bypassed by well-meaning professionals that are pressured to meet deadlines or other politically-based directives. The reliability team functions as a final gateway into production for the most critical superset aggregates to form a quality barrier against political pressures.

It is the final wall for the most critical urgent software development and may ensure that quality measures are in place and that CI/CT thresholds are met. This practice is supported by testing toolsets, code-quality reviews, and proactive, predictive processes.

The continuous reliability team must leverage its overall production-level system and application knowledge, its breadth of system and application knowledge, and its own development skillset to create proactive and predictive toolsets. These toolsets respond not only to existing issues but also to proactively attacking known vulnerabilities. As a result, test scenarios are created that stress, break, and ultimately predict system weaknesses to fortify the production baseline. Some of the tools used in this approach include:

- AppDynamics: provides insight into application components across a cloud-based network. AppDynamics focuses on availability and performance.(Create Your Center of Business Observability, 2021)

- Splunk: provides a repository for machine-based data. Its web-based interface can be used to correlate, graph, and analyze data for trends, diagnostics, and mediation purposes.(Splunk Enterprise 8.2.0, 2021)
- Grafana: provides numerous visual-based data aggregations across multiple sources. This includes charts, graphs, dashboards, and other web-based indicators.(Create Your Center of Business Observability, 2021)

Information Technology Services

Four IT services drive USD. Each of these services grease development to move at an accelerated rate. By name, these services are not new. By operation, these services breathe life into urgency.

Information Technology Governance

IT governance is a long-standing service for providing a structure for aligning IT strategy with business strategy. It considers business unit, stakeholder, and employee interest to work toward the organization's goals and processes to be followed.(Lindros, 2017)In USD, governance embraces a dynamic priority process. Decisions are made in tight timing windows to respond to triggers. Based upon a daily or hourly triage, IT governance sets the priority of work to be completed and which applications will be deployed.

Enterprise Architectural Governance

Enterprise architectural governance (EAG) involves review, standards, and roadmap processes to generate policies, processes, procedures, and standards for the enterprise architecture. The timing windows of USD erode architecture. Continuous development can result in large or incremental changes to enterprise architecture. Enterprise architectural governance functions as oversight to the lighting fast software development initiatives. To govern USD effectively, the three components of EAG are enhanced:(Enterprise Architecture Governance, 2021)

- Architecture review process: This process is no longer periodic. It is operationally focused with the objective of ensuring conformance to strategy and standards. Reviews are exception-based triggers only with an issue classified with "high" enterprise impacts.
- Standards management process: In USD, the standards management process establishes change control priority over architecture standards. It requires that change control independent of a traditional change control board. Greater change control authority is delegated to the project team.
- Roadmap management process: A final component of effective USD EA governance is managing change against the enterprise roadmap. The roadmap provides lower volatility and more stability resulting from continuous time periods. Tolerances are established for acceptable and unacceptable architectural deviations.

Application Portfolio Management

Urgent software development targets important applications within the application inventory. How does USD know what applications are important? The application portfolio includes attributes for referencing "importance". Applications are classified as mission critical, mission essential, and mission foundational. Other attributes detail interaction between and among applications.

Who calls whom? What data is public, passed, and private? With these attributes, USD identifies which projects contain or interact with high-priority, high-risk applications.(The Definitive Framework for Application Rationalization, 2021)

IT Investment Portfolio Management

The resources allocated to software development are identified, approved, and encumbered in the organization's IT investment portfolio. Investment portfolio management utilizes diversification to reduce risk.Diversification across different software development projects with urgency classes of have risk return characteristics. Portfolio management evolves from a cyclical-based portfolio to an event-based portfolio reviewed and evaluated continuously throughout a fiscal cycle.(Ragin & Garibaldi, 2021)

Test-Driven Environments

A test-driven environmentprovides the testing approach needed in Urgent Software Development. TDE uses the concept of"supersets." A superset is multiple predefined test sets that consist of the following components:

1. Testing within and across domains based upon critical business operational scenarios.
2. Identifying and defining the applicable applications within each of the supersets.
3. Mapping and planning the necessary infrastructure to support each superset test.
4. Defining and creating "data slices" that correspond each business scenario.Data slices are spun up when the infrastructure is spun up and can be accessed as relevant data on demand.

Way of Working

Way of working is how all the ecosystem nodes operate in an integrated, unified manner. Leadership, agile methodology, tools and practices, IT services, and TDE collectively performance as an operating organism, mutually interdependent parts functioning together. USD evolves and changes its composition. WOW operates as the "glue" that ties the other ecosystem nodes together into a cohesive, evolving environment. Way of Working not only captures and defines the operating nature of each individual ecosystem node but also acts as a diagnostic response mechanism and proactive instigator for favorable change across the entire IT organization. The WOW node accepts input (both favorable and unfavorable) from each USDE node, analyzes the input from the focus point of the whole, and instigates and promotes modifications across the entire domain of ecosystems as necessary. WOW is ever evolving, always probing, and continually updating its "Way of Working" definition. This process effectively drives the operating blueprint for what is current and for what is future state. It acts as the control mechanism for all the USDE nodes as they operate in cooperation with each other as well as an accelerator to embrace necessary change. A healthy WOW node lowers the barriers that exist among USDE nodes and acts the arbiter to ensure a smooth catalyst for change. Finally, the WOW node provides both valuable feedback and future state opportunities to the leadership node to help maintain a vibrant IT evolutionary process.

WOW requires managed granularity in the context of today's large software solutions. It relies upon generating and maintaining small chunks of code. To response urgently to a trigger, more granular code is utilized. Software engineers use a microservice architecture reference model to understand the breadth and depth of code chunk operation and logic. These small chunks of code are designed with strong encapsulation and separation of concerns, are largely independent of other code, and generally stand alone in both application testing and functional execution. Code issues are logically isolated to the code's microservice. In general, this isolation allows the overall application to operate while the microservice is altered. Monolithic code-based architecture, by contrast, are often affected by functional code problems and are either up or down based on its weakest link.

When generating code, thresholds are set high to ensure that all urgent software deployment progresses through a series of high quality assurance processes at both the unit and integration levels. Code is controlled within a code repository (such as Gitlab)(DevOps Platform, 2021), moved within a pipeline structure (such as Jenkins) (Pipeline, 2021), and iteratively driven through quality tools (including Junit(Why Use JUnit for Testing?, 2021)), Jacoco(Jacoco (Java Code Coverage) - Tutorial, 2021)), SonarQube (SonarQube 8.8, 2021) and Fortif(Micro Focus Fortify Software v20.2.0, 2020). In addition, more advanced testing should be embedded that goes beyond a unitfocus and more toward integration and end-to-end (E2E) (using scripts created in toolsets like Cucumber). (Tools & Techniques that Elevate Teams to Greatness, 2021)This coding practice proves its worth by having two programmers involved with writing and testing code and reviewing it for functionality and quality at the same time. In USD, combining these efforts into the initial code building process serves to ensure both functional understanding and high-quality builds since the process does not rely on a single perspective. Each participant can then take their combined knowledge to other pairs providing for a broader understanding of functioning code.

Architecting an Urgent Software Development Maturity Model

The Urgent Software Development maturity model depicts a succession of changes related to how software development is managed and conducted. As shown in Figure 2, maturity is defined as the organization-wide ability for managing software development and maintenance processes. With a mature USDE, a disciplined evolution is consistently followed as an organization moves to be an urgent-enabled status:

- Level 1 Initial: no or missing ecosystem nodes; software development is conducted at the local level without organizational coordination.
- Level 2 Established: ecosystem nodes are implemented; software development is conducted at the organizational level with aggregate planning processes; the planning period is cyclical based on annual or quarterly cycles.
- Level 3 Integrated: ecosystem nodes work together; drive each other and respond to the outcomes of each other's contributions; software development shifts from cyclical-based planning to embrace event-based triggers.
- Level 4 Monitored: ecosystem nodes operation is measure for efficiency and effectiveness; urgency is identified at an operational level which could be with a business unit or set of business units; software development shifts away from cyclical-based planning to event-based triggers; traditional cyclical-based planning embraces temporal events to accommodate traditional cycle-based projects.
- Level 5 Optimized: ecosystem nodes operate in an urgent state in which projects are initiated in near a real-time dimension; urgency expands to be a business-critical level with a going concern impact; software development feeds from a continuous pipeline of software development from which deployment can be obtained for a "ready" version of production software "awaiting" a trigger to active it.

As shown in Figure 2 to realize USD, an organization moves through five levels of maturity. The maturity model has a time dimension on the x-axis and the ecosystem on the y axis. Time is on the x axis because USD organizations evolve from a cyclical to continuous definition of time. At the initial level as software development is conducted, month-to-month or week-to-week time dimensions are used to organize software development work efforts. At the initial level, time dimensions are abstracted for individual projects. At the established level, the time dimensions are extrapolated from aggregate planning for a portfolio of projects. This extrapolation is predictive in nature because the organizations seek to manage aggregate supply and demand of resources. At the integrated level, a shift occurs in the time dimension. Time embraces the unpredictable, unanticipated triggers that "force" a solution. At the monitored level, the concept of predictability diminishes. Plan-based project become temporal event-based projects that can drives plans. Finally, at the optimized level, the time variable changes. Rather than being driven by triggers, software development proceeds continuously with the possibility of small chunk modifications as a viable production response to the trigger. In theory, the deployment is real-time with the trigger, even though in practice the time interval for production deployment could be a short one. To achieve urgency, an organization evolves from operating using a discrete time interval to a continuous time measure.

Urgent Software Development Maturity Model

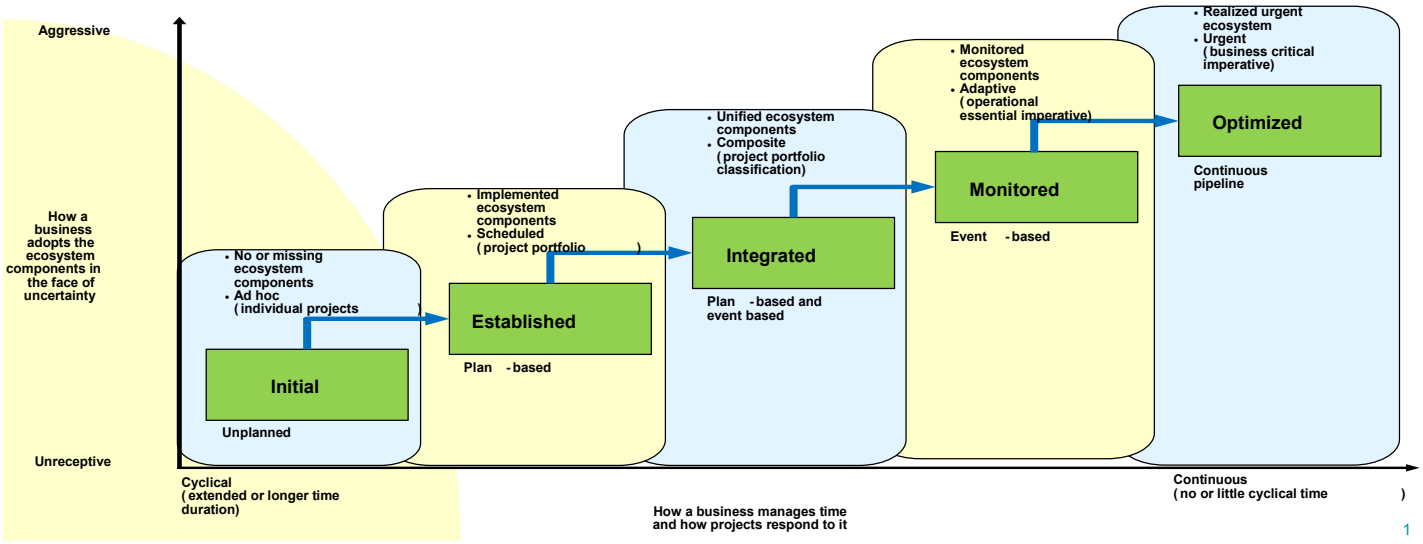

Figure 2 Urgent Software Development Maturity Model

Along the y-axis, the implementation of the ecosystem nodes is represented. In a perfect world, the limitations and constraints of ecosystem nodes would not exist. For example, an organization would have no traditional software development. An organization would operate full blown DevOps with CR, CI, CT, and TDD. At level one in the USD maturity model, an organization has not implemented the ecosystem nodes. The management and leadership style might not be progressive, or the IT investment portfolio is filled with traditional software development projects and no urgency classification is utilized. As an organization matures to the integrated level, the ecosystem nodes interact and affect the performance of one another. At this level, the application portfolio management and IT investment portfolio management processes interact with the methodology element. For example, not all the installed base and software development projects can be urgent and conducted iteratively. Installed applications and software development projects can be classified as mission critical, mission essential, or mission foundational to establish priorities with the portfolios. The governance process can be refined for oversight of these higher priority software resources. At the integrated level, event-driven development is legitimate, and urgency is formally a part of software development and solution delivery.

Once urgency is formally accepted, the time dimension grows more important. At the monitored and optimized levels, the time dimension for urgent software development shifts from cyclical to continuous. To response in urgent situations, an organization focuses on an operational imperative in the organization's business model. The goal is not to implement a solution on time. Rather the goal is to implement a solution that solves a critical operation imperative. The impact of the degradation or failure of this operation is mission critical to the company. To determine if USD is being delivered in an urgent manner, measurements are taken. Both qualitative and quantitative criteria are applied to measure USD performance. One qualitative criterion is timeliness, which is the ability to meet deadlines. Another qualitative time-related criterion is delays, which are arbitrarily long solution deliveries. A third USD performance criteria is quantified.Worst-case delivery times to occurring event and triggers provides measurements for development urgency. Utilizing a total cost of ownership (TCO) discipline, overall project costs (the bottom-line) yield the fiscal measurement of software development in an urgent state. At the optimized level, the USDEoperates as a wholly integrated ecosystem. The organization's urgent software development responds readily to events and triggers in a situational, continuous time context. Solution deployment is completed within targeted days or hours of an overall business critical imperative requirement.

Summary

USD is an innovation that goes beyond hype. "A hype cycle is a graphical representation model produced by Gartner Inc. that helps organizations understand the maturity and adoption of new and emerging technologies and how they can be used to address and solve real business problems."(What Does Hype Cycle Mean?, 2021)Gartner's Hype Cycle is a graphical depiction of a common pattern that arises with each new technology or other innovation.(Hype Cycle, 2021)USD is a cultural, managerial, technical, methodology, tools, and practice architecture for software development in a continuous time dimension. It is not proposed that the USDE outlined in this paper is the "optimum" one. Rather, it is proposed that the presented ecosystem is the initial foundation for moving USD to the forefront of organization's vision if it seeks to operate more efficiently and effectively in today's trigger-driven, continuous time environment.

The most fundamental USD requirement is the ability of the ecosystem to respond to external events with very short, deterministic delays. USD usually runs with critical deadlines. Therefore, an ecosystem must have a fully deployed and integrated set of nodes operating in a continuous time scale. The explicit involvement of the dimension time distinguishes USD from other forms of software development. This is expressed by the following two software development requirements, which USD must fulfil even under extreme time pressures: timeliness and near simultaneity of trigger and deployment behavior. These requirements are supplemented by two additional ones of equal importance: predictability and dependability. When triggered by external events, development, quality assurance, and deployment must be performed on time. The mere solution response speed is not decisive, but the timeliness of reactions within predefined and predictable time bounds is vital. Hence, it is characteristic of USD that its functional correctness does not only depend upon the processing results, but also upon the instants, when these results become available. Correct solutions are determined by the ecosystem, which yield to a quality solution from a development pipeline in contrast to published traditional and agile methodologies.

Refences

Create Your Center of Business Observability. (2021). Retrieved from AppDynamics: https://www.appdynamics.com/product. Accessed on 10 August 2022.

DevOps Platform. (2021). Retrieved from GitLab: https://about.gitlab.com/. Accessed on 10 August 2022.

Enterprise Architecture Governance. (2021, February 6). Retrieved from CIOIndex: https://ciowiki.org/wiki/Enterprise_Architecture_Governance. Accessed on 10 August 2022.

Goncalves, M. (2015, January 6). Progressive Leadership Begins with a Progressive Leader. Retrieved from Linkin: https://www.linkedin.com/pulse/progressive-leadership-begins-leader-marcus-goncalves-ed-d-/. Accessed on 4 September 2022.

Hype Cycle. (2021). Retrieved from Gartner: https://www.gartner.com/en/information-technology/glossary/hypecycle. Accessed on 4 September 2022.

Jacoco (Java Code Coverage) - Tutorial. (2021). Retrieved from Vogella: https://www.vogella.com/tutorials/Jacoco/article.html. Accessed on 4 September 2022.

Lindros, K. (2017, July 31). What is IT governance? A formal way to align IT & business strategy. Retrieved from CIO: https://www.cio.com/article/2438931/governanceit-governance-definition-and-solutions.html. Accessed on 4 September 2022.

Micro Focus Fortify Software v20.2.0. (2020). Retrieved from Micro Focus: https://www.microfocus.com/documentation/fortify-software-securitycenter/2020/FortifySW__RN_20.2.0/FortifySW__RN_20.2.0.htm. Accessed on 4 September 2022.

Operation Warp Speed. (2021, February 11). Retrieved from GAO@100. Accessed on 4 September 2022.

Pipeline. (2021). Retrieved from Jenkins: https://www.jenkins.io/doc/book/pipeline/. Accessed on 10 June 2022.

Ragin, M., & Garibaldi, C. (2021). Project Portfolio Management Services. Retrieved from Deloitte: https://www2.deloitte.com/us/en/pages/consulting/articles/project-portfolio-management-

services.html?id=us:2ps:3bi:consem21:eng:cons:60520:nonem:na:OXe97WtG:1191982706:76759794503399:be:Business_Tech_Transformation:Portfolio_Management_Services_Exact:nb. Accessed on 4 September 2022.

SonarQube 8.8. (2021). Retrieved from SonarQube: https://www.sonarqube.org/sonarqube-88/?utm_source=bing&utm_medium=paid&utm_campaign=sonarqube&utm_content=sonarqube. Accessed on 4 September 2022.

Splunk Enterprise 8.2.0. (2021). Retrieved from Splunk: https://www.splunk.com/en_us/download/splunkenterprise.html?utm_campaign=bing_amer_en_search_brand&utm_source=bing&utm_medium=cpc&utm_term=splunk&utm_content=Splunk_Enterprise_Demo&_bt=71811971151777&msclkid=3f857e37007116e 594871e8173950ffd. Accessed on 16 June 2022.

The Definitive Framework for Application Rationalization. (2021). Retrieved from Apptio: https://www.apptio.com/resources/ebooks/definitive-framework-applicationrationalization/?utm_source=bing&utm_campaign=core-dbv_ams-multi-en_apprat&utm_medium=cpc&utm_term=application%20portfolio%20management&utm_source=bing&utm_me dium=cpc&utm_term=appli. Accessed on 4 September 2022.

Tools & Techniques that Elevate Teams to Greatness. (2021). Retrieved from Cucumber: https://cucumber.io/. Accessed pn 16 June 2022.

What Does Hype Cycle Mean? (2021). Retrieved from Techopedia: https://www.techopedia.com/definition/29298/hype-cycle. Accessed on 4 September 2022.

Why Use JUnit for Testing? (2021). Retrieved from stackoverflow: https://stackoverflow.com/questions/10858990/why-use-junit-for-testing. Accessed on 4 September 2022.

Your Observability Wherever You Need It. (2021). Retrieved from Grafana Labs: https://grafana.com/. Accessed on 16 June 2022.