

CONTINUOUS DELIVERY

CSC



Contents

- Executive Summary 1
- Introduction 1
- The Call to Action: Continuous Delivery Hits the Enterprise..... 1
- The Problem: Traditional Enterprise Delivery Process Increases Cost of Migration as Value of Change Increases..... 2
- Traditional Value Cycle 3
- Customer Needs Continue to Evolve 5
- The Solution: Continuous Delivery Drives Value through Iterative Releases 5
- Applying Agile Standards and Frameworks 6
- Information Technology Infrastructure Library 7
- Continuous Delivery Processes and Releases 7
 - Inception..... 8
 - Construction 8
 - Verification..... 8
 - Delivery 8
 - Considerations Within the Software Development Life Cycle..... 9
 - WiRelease Application to Increase Customer Influence..... 9
 - Guiding Principles 9
 - Quality and Performance along the Assembly Line 12
 - Summary Phase..... 12
- Quality 14
 - Quality Assurance 14
 - Quality Control 14
- Metrics for Success..... 15
- Customer Impact and Benefits..... 16
- Managing Change Effectively 16

Figures and Tables

Figure 1: Traditional Delivery Process 3

Figure 2: Traditional Value Cycle 4

Figure 3: Value and Effort are Related 4

Figure 4: Changing the Focus to Time-To-Value (TTV) Through Small Increments 5

Figure 5: Incremental Improvement with Quality 6

Figure 6: Software Development Life Cycle..... 8

Figure 7: WiRelease Application Use Cases 9

Table 1: Guiding Principles Scorecard..... 10

Table 2: Guiding Principles Descriptions 11

Figure 8: Overview of the Core Continuous Delivery Process..... 12

Figure 9: Continuous Delivery is Driven by Customers 13

Executive Summary

The value of any enterprise-level software is only achieved once the software is in production — serving its purpose to lower operational costs and increase productivity. As operational needs change, it becomes necessary to modify the software. The faster these changes can be adopted, the sooner CSC customers see value.

The need to make changes quickly is the driving factor for Continuous Delivery, an advanced process through which CSC customers receive new functionality as soon as it becomes available. Continuous Delivery releases small increments of code that can be moved into production immediately. With Continuous Delivery, customers can react to market changes faster and continually benefit from up-to-date technology.

Introduction

The purpose of this document is to provide an overview of Continuous Delivery, including a description of the factors that led to its creation.

The objectives of Continuous Delivery are to provide CSC's P&C customers with software and services that are:

- High quality
- Able to receive incremental code delivery
- Easier and more manageable to upgrade
- Unified on a common code base
- Grown through rich contributions to the shared code and object base by a network of contributing partners and customers
- Products of an agile software development life cycle (SDLC)
- Documented comprehensively

To meet these goals, we rely on an independent Quality Office, an agile SDLC, sensitive metrics and an online collaborative community called WikonnecT. CSC's groundbreaking social networking site, WikonnecT, leverages Web 2.0 for rich and frequent interaction among our customer community and our staff. Aligning our culture with that of our customers is a very high priority, as it ensures a cooperative and ever-improving implementation of CSC's P&C software.

As more completely explored below, enterprise software development has been focused on updating the next big release of a package. This situation has increased barriers for customers. Continuous Delivery helps overcome these barriers. Continuous Delivery keeps looking forward by ensuring that the smaller, incremental releases focus on customer needs and demands, and evolving changes in technology. At the same time, it keeps looking backward, in a sense, by introducing these changes incrementally with minimal interruption.

The Call to Action: Continuous Delivery Hits the Enterprise

Historically, enterprise software practices have adopted capabilities from diverse software and hardware environments. This scenario includes a layered architecture for both logical and physical views. While this provides benefits for separation of concerns, scalability and other dimensions, it has also bloated the typical product life cycle with large, development-heavy releases. The added propensity for enterprise IT organizations to customize, and a reluctance to change content practically ensures harmful financial results if upgrades are completed incorrectly. As a result of this, introducing change has historically been heavily managed, process-intensive and expensive.

CSC has changed the entire dynamic by focusing on the following criteria for success:

- **Customers adopt Continuous Delivery in Exceed C.0 and POINT IN C.0**

Exceed C.0 and POINT IN C.0 are the baseline versions upon which all future releases are built. Continuous Delivery provides value to CSC customers that adopt the baseline releases and build upon them through continuous and progressive application of releases.

- **All packages of high quality are purple**

As will be explained in more detail throughout this document, Continuous Delivery focuses on the customer. Our internal organizations do not focus on delivery to the next internal organization. We see the customer as the delivery and implementation location. Experience as measured at customer locations is the significant yardstick against which all deliveries must be gauged. This is “purple.”

- **All packages follow the same process**

Consistently applied and improving processes drive higher quality with greater delivery efficiency. There are no pre-releases. There are no shortcuts. The process must be flexible enough to improve incrementally, but any process element important enough to have is important enough not to skip.

- **Continuous Delivery enables efficient and effective releases**

Continuous Delivery is measured for both efficiency and effectiveness, and then exposed to a continuous process improvement evaluation to maximize every opportunity for efficiency gain. Effectiveness is measured by the continuous delivery of features with an ever-decreasing rate of error.

- **Customers realize value of deliveries quickly and easily**

Continuous improvement on an expanding, shared code base enables customers to realize value more quickly and easily than with alternative approaches. Rather than focus on delivering new versions of the entire application, we continuously improve the existing version and the customers' time-to-value ratio.

- **Number of one-off deliveries is reduced or eliminated**

With Continuous Delivery, the common code base benefits the whole community. New features are introduced and shared with all customers at one time. Defect resolutions are delivered to all customers simultaneously. The net effect is a more focused and continuous path to a wider, shared code base with less redundant delivery efforts. Rather than fixing the same thing multiple times for different customers, the focus is enabling the same configurable change for multiple customers at one time.

In contrast to the traditional enterprise delivery process, Continuous Delivery improves and expands the application in place. With a strong focus on quality, large-scale migrations and upgrades become targets for dissection into smaller, incremental deliveries. Customers are encouraged to apply changes to their applications continually rather than relying on larger, riskier migration and implementation schedules. Continuous Delivery releases are delivered in smaller portions and on a more frequent basis. Likewise, customers are monitoring and applying changes in smaller, more manageable portions than with the traditional enterprise development and release cycle.

The Problem: Traditional Enterprise Delivery Process Increases Cost of Migration as Value of Change Increases

A common method of enterprise development can be divided into three primary domains as illustrated in Figure 1(see below). Typical organizations collect requirements and requests from a variety of sources and funnel them through a base development organization. Base responds to the requests by merging them into the next major release. The focal point for this type of organization is typically on the future and on the next big release. This organizational method also carries the burden of including defect fixes in the next release while making them available for incorporation into the various existing and frequently deployed code bases.

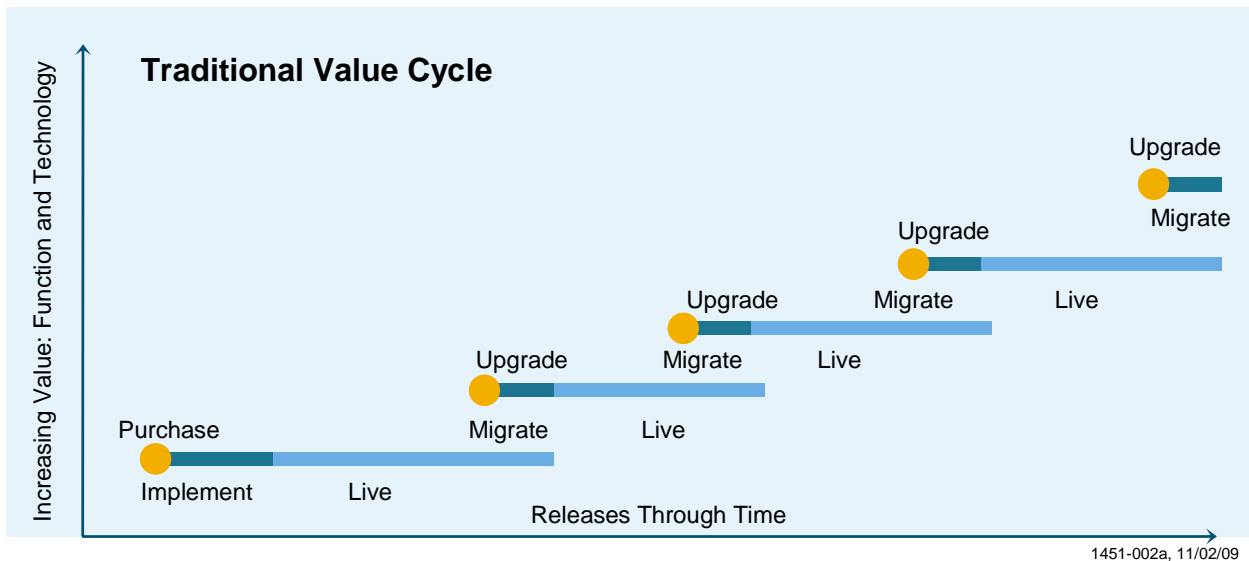


Figure 2: Traditional Value Cycle

If migrations and upgrades remain low-risk and low-effort then this model can work well. As Figure 3 illustrates, there is an underlying conflict inherent in this approach. Specifically, the relationship between the upgrade and migration effort is directly proportional to the release value of the changes. So, as software vendors continue to add more value through the changes made to the software, the customer must pay a proportional amount in order to take advantage of those changes as costs rise — and the friction associated with the changes increases. Thus, customers may find that given their particular business needs, it's more sensible to delay the introduction of changes into their environments. And with multiple requests from multiple customers, the software vendor must stretch its resources to develop many unique versions of the same software simultaneously.

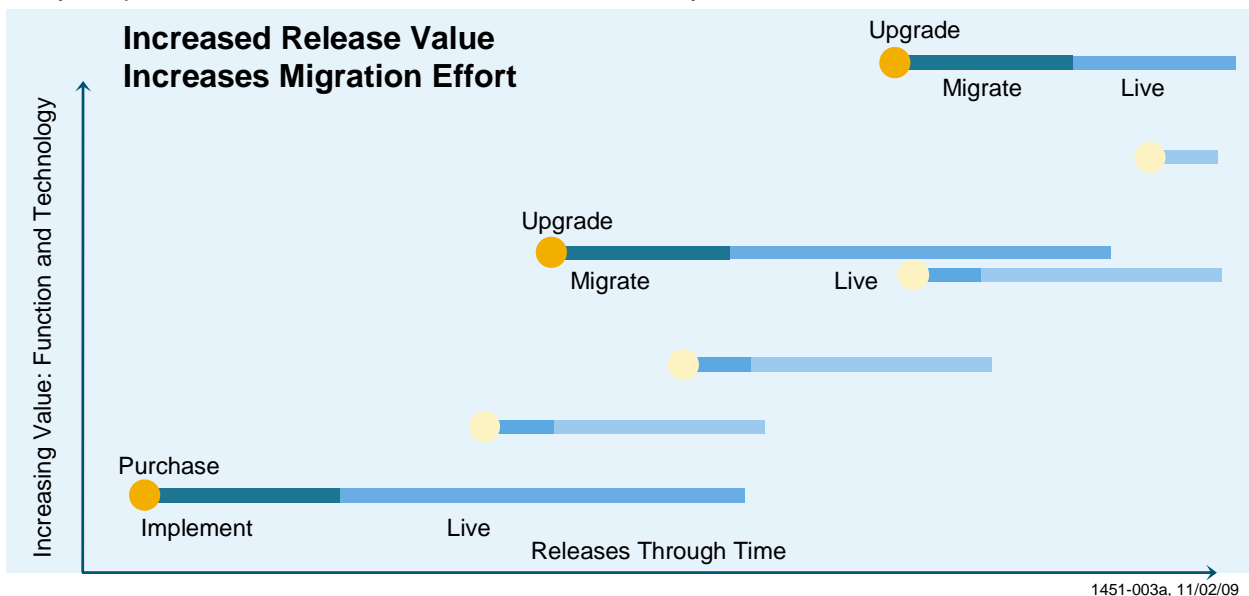


Figure 3: Value and Effort are Related

Customer Needs Continue to Evolve

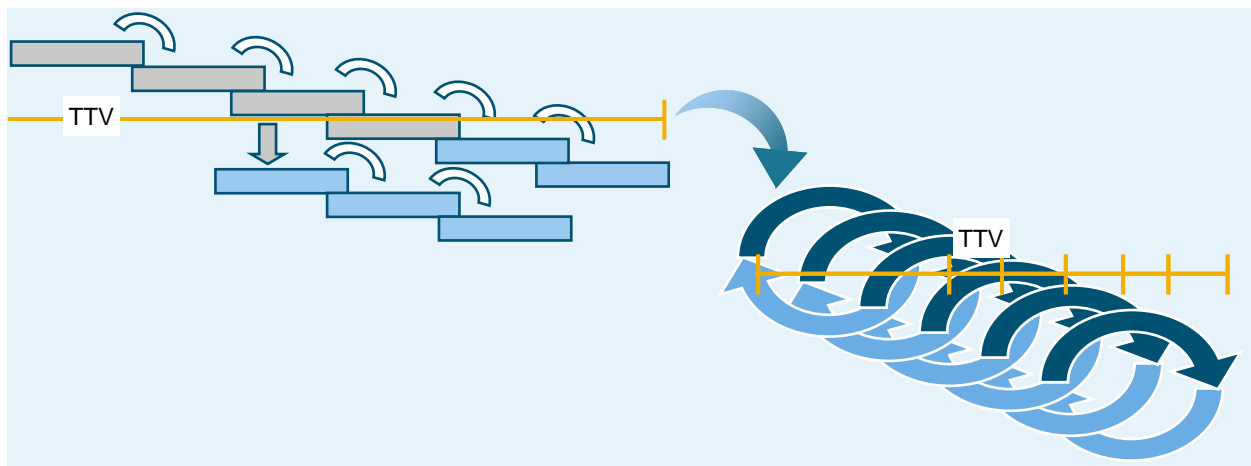
Within the traditional value and development cycles, customers are bombarded by new demands from a variety of sources including:

- Regulatory changes (i.e., Sarbanes-Oxley, USA PATRIOT Act, insurance regulatory boards)
- Product marketing and strategy needs
- Acquisitions
- Competitive pressure.

The velocity of these changes can be ferocious. The complexity of these changes and the expectations they create can add to the pressure to include even more customer-specific changes into the application. Compared to the rate of a vendor's traditional product development and delivery cycles, changes to customer needs expand much more rapidly.

The Solution: Continuous Delivery Drives Value through Iterative Releases

Here's another way to think about the way enterprise software has been delivered. Figure 4 illustrates many enterprise software development cycles focused on delivering large releases through a series of waterfall processes. Each step down the waterfall is intended to validate the software's achievement of quality goals and requirements for each stage of development. The later the problem is discovered, the higher the cost of resolution. The effect is multiplied if one has to repeat the movement through each of the stages that were previously considered complete. In some cases, emergency or urgent fixes might be pre-released to enable customers to more rapidly introduce changes into their environments. The blue boxes represent the waterfall steps defined in the processes for the software provider and the red boxes represent the waterfall steps defined for the customers' processes. As the diagram illustrates, the time-to-value (TTV) for each feature basically coincides with final release. Thus, the TTV for each feature is the same and large.



1451-004a, 11/02/09

Figure 4: Changing the Focus to Time-To-Value (TTV) Through Small Increments

By focusing on smaller, incremental releases, Continuous Delivery incorporates a number of software development life cycle (SDLC) processes that have proven to bring value to many organizations (see References 1, 2 and 3). These agile and hybrid approaches offer opportunities for delivering quickly without sacrificing the checks and balances one expects from more traditional waterfall practices. As illustrated in the right side of Figure 4, features can be delivered in a series of development cycles on a continuous basis such that any given feature of the TTV is smaller. Within each increment, the development process still includes checks and balances to ensure that quality attributes are included and verified. This is Continuous Delivery.

Applying Agile Standards and Frameworks

To achieve the goals of Continuous Delivery, CSC applies agile practices, standards and frameworks. The core methodology is to select features from a list, determine the sequence of releases and then execute iterative development cycles. This is based on Function Driven Development (FDD) as illustrated in Figure 5. We develop a Master List of options for inclusion in Continuous Delivery releases (CDRs), and we work with our customers to determine the severity and priority of these items. The most desirable and critical items (this does not mean ONLY critical items) are arranged along application timelines for inclusion in a release. The end results are a repeatable and measurable process that we continually improve according to pre-set goals, and the subsequent comparison of these goals against actual results.

In addition to new application features and the resolution of defects, each release scheduled intentionally targets improvements in processes and automation as well as architectural improvements (i.e., refactoring). Thus, the core infrastructure, surrounding management and configuration of our P&C applications get the same incremental improvement as core features.

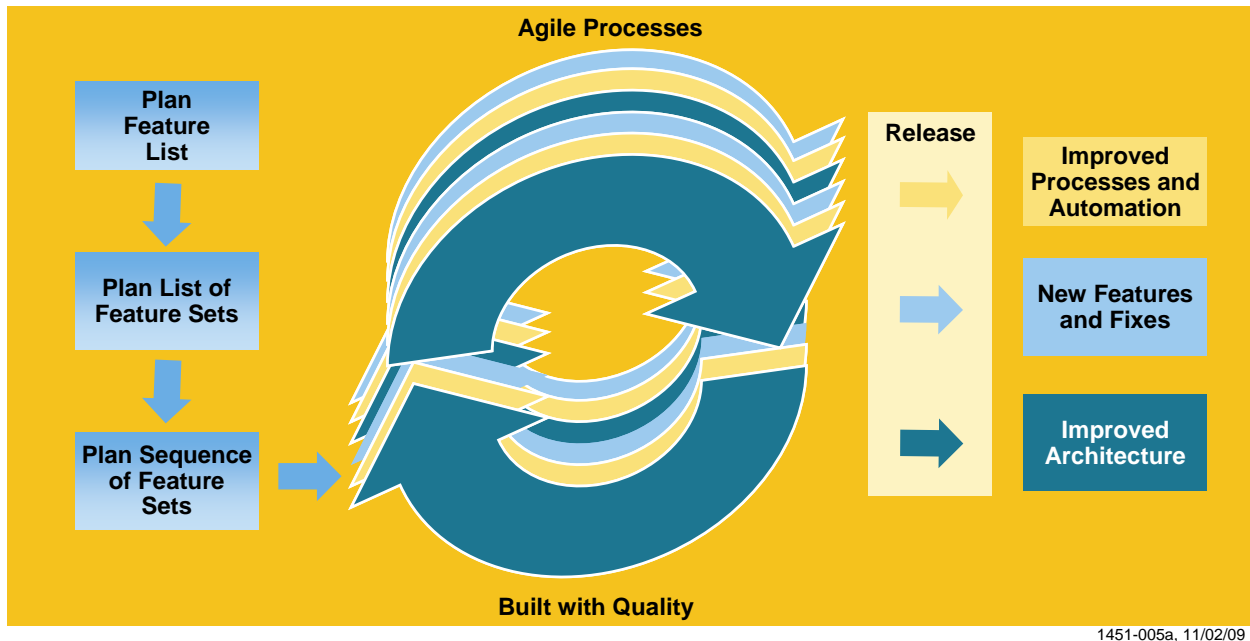


Figure 5: Incremental Improvement with Quality

This incremental approach allows Continuous Delivery to start while we define and run the process through an ongoing series of improvements. Incremental process improvements are managed by CSC's Delta Team, which interacts with all of the organizations that create, deliver and service CSC's P&C applications. The Delta Team represents several key areas of interest as it defines and refines the very processes that make our agile development process so successful.

Role

Configuration Management and Packaging

Customer Readiness

Responsibility

- Environment setup
- Library control
- Packaging

- Ensuring processes serve customer needs
- Custom modifications standards
- Innovative community standards
- Customer information tracking

Role	Responsibility
Documentation	<ul style="list-style-type: none"> • Define documentation requirements for each project • Identify necessary improvements for creating and maintaining documentation
Change Manager	<ul style="list-style-type: none"> • Software Development Life Cycle • Upgrade experience

Incremental improvement and iteration remains the overall goal while each individual project is kept independent of other projects and staffed with a small team. Each project receives treatment for major concerns about features, processes and architecture. Likewise, each release is intentionally designed to be installable and configurable independently. Customers can keep their code base current and choose when to take advantage of new features. Customers do not skip releases that are applicable to the subsystems they have licensed, but they do control the rate and interval at which they accept released packages. Because the intent is to keep the licensed systems current, intentionally delaying adoption of releases for extended periods defeats the purpose of the process and is not recommended.

We follow up with customers to ensure that metrics collected during and after implementation of Continuous Deliveries demonstrate that incremental improvements are performing as expected. We also incorporate continuous integration and test-driven development (see Reference 6) into our internal processing with the explicit intent of identifying build-and-logic problems as early as possible in the development cycle.

Just as deprecation notices are provided for standards and interfaces by many standards bodies and software vendors, CSC provides detailed statements of intent to discontinue support for specific levels of software. Our customers have ample notice of impending support changes, so they can plan for faster adoption of releases. Otherwise, they run the risk of having a code base that is no longer supported, or hearing the dreaded “you need to upgrade” response to requests for defect resolution.

Information Technology Infrastructure Library

The Information Technology Infrastructure Library (ITIL) is the framework against which we compare and shape CSC’s software development, management and delivery processes. This framework is used by many large software vendors to manage IT infrastructures and to compare nomenclature and processes. It enables Continuous Delivery to take advantage of best practices and tools that have and will emerge as standards-based dialogue continues to drive supporting tools and practices.

The first significant contribution to Continuous Delivery from ITIL is the use of a configuration management database (CMDB). This is one of the central concepts introduced with ITIL. Initially, CSC is using a CMDB to achieve two goals. First, we place and reference a CMDB at each customer’s location as a part of the installation and configuration process. This CMDB is used to capture, report and support the exploitation of information about the core installation and the application of particular releases for each customer. This is our customer implementation database. A second CMDB is maintained by CSC, and contains information about customer release requests, receipts and deployments. Like the overall process described above, the evolution of these CMDBs and their uses will change over time.

Continuous Delivery Processes and Releases

The software development life cycle as illustrated in Figure 6 is divided into four phases, each with a series of steps through which CDR projects are executed. Prior to the Inception phase, CSC interacts with customers, their representatives and our own internal marketing groups to shape the information contained in the Master List (described above). This helps to ensure that items going into the Inception

phase and, most importantly, to later phases are those that most closely match the desires of our customers and the general market.

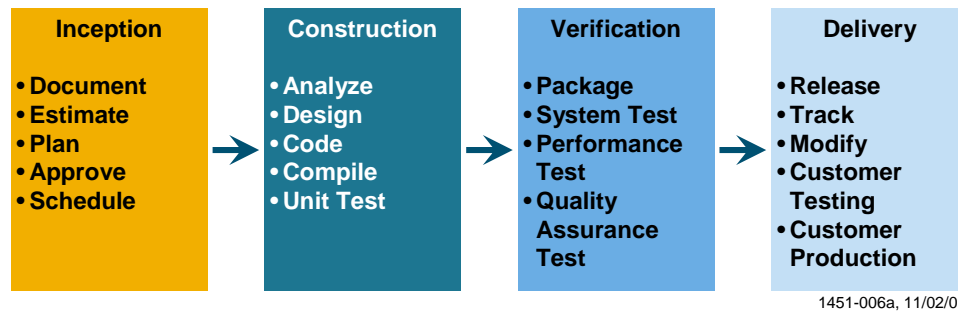


Figure 6: Software Development Life Cycle

Inception

The Inception phase is the initial phase of the software development life cycle. During this phase, the business requirements are received and documented. Possible approaches and rough order of magnitude (ROM) estimates are also created during this phase. The approaches are verified with the customer and an ultimate approach is decided. Once the approach has been defined and approved, then a more detailed estimate, such as a project brief, project change request or project management plan, is created. A high-level project schedule is also created during this phase. The Continuous Delivery Guiding Principles Scorecard (see Tables 1 and 2) is executed during this time frame to ensure that the project meets the key tenets of the process prior to project start. Once all artifacts are created, they are reviewed with key project sponsors. After approved, the project can then be scheduled.

Construction

Design documents and code are built and initially tested in the Construction phase, which employs a business and technical specification document to design how the business requirement will be achieved. Peer reviews of specifications are conducted during this phase. System changes to achieve the design are also created during this phase. They may include changes to code, database and data entries. Developers conduct tests in a unit test region and in an integrated unit test area during this phase.

Verification

After Construction is complete, the development organization tests the integrated version of the application for functional fit with a development system test (DST), and for performance with a performance system test (PST). The testing process includes direct verification of a release's installability. Given adequate performance among these criteria, the resulting packaged release is provided to the independent QA testing and performance lab for independent testing.

Delivery

Upon satisfaction of established criteria for both stages of Verification, the release moves into the Delivery phase, where it is made generally available to customers. The Delivery phase may also include retrofitting a release into an individual environment, modification for customer-specific needs, delivery to specific customers for further testing, and ultimately moving into production at customer sites. Measurement and tracking of the released items also occurs during this phase.

Further, all work conducted by CSC personnel follows the processes supporting Continuous Delivery. If the work is focused on a custom modification for a particular customer, it may not become a Continuous Delivery release but the same checks and balances to ensure continuous improvement and quality are still exercised.

Considerations Within the Software Development Life Cycle
WiRelease Increases Customer Influence

Using Wikonnect, our customers can see and influence the progress of the CDRs they sponsor. The WiRelease widget gives them reports, dashboards, summaries and some details of their projects and where they are in the SDLC. Thus, customers can see and influence the progress of their particular features into the base application via Continuous Delivery.

WiRelease provides more transparency to internal and external audiences for the contents of the Master List and the projects going through the Continuous Delivery process.

WiRelease is one of many Wikonnect widgets we use for online, real-time collaboration with our customers. Wikonnect communities, representing various constituencies, are able to post WiRelease blogs, message boards, RSS feeds, and other Web 2.0 widgets on their Wikonnect pages. Within the communities and pages, members rate and comment on releases, work items, new requirements, changing business environments and such. This interaction enables the network effect of multiple connections to drive value directly into our software and processes.

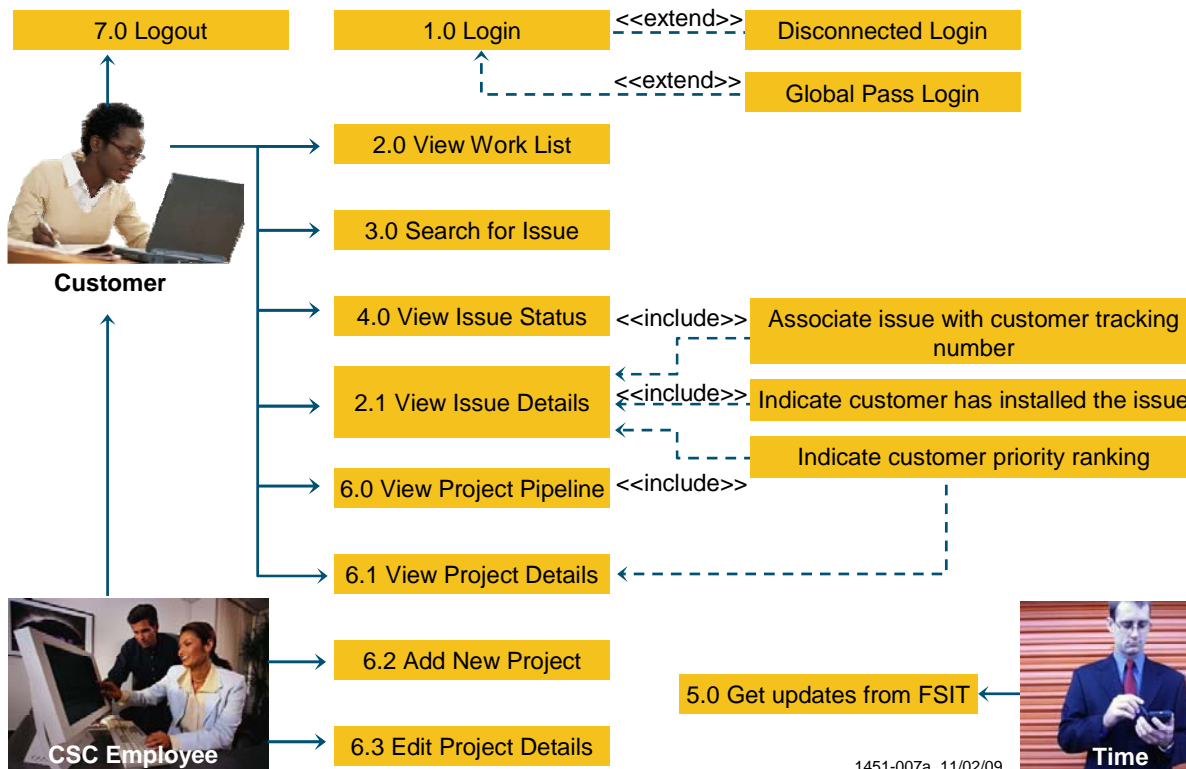


Figure 7: WiRelease Use Cases

Guiding Principles

One of the new items against which each Continuous Delivery project is measured is the Continuous Delivery Guiding Principle Scorecard. This scorecard looks like the following:

Principle	Score				Comments
	G	Y	R	N/A	
Overall					
Easier to upgrade					
Update database without downtime					
Use databases in a manner that promotes easy migration					
Contains comprehensive documentation					
Maximum customer code coverage is tested					
Configurable off or on (initially delivered off)					
Is backward compatible					
Conforms to standards					
Enable the Externalization of customer change to avoid inline code					
Improves interfaces					
Compartmentalizes application changes					
Can be installed standalone					
System Performance (<i>see below</i>)					
Overall					
Fast					
Efficient					
Scalable					
Easy to use					
Is independent					

Table 1: Guiding Principles Scorecard

The following provides a sample of the description and definition for some of these principles:

Criterion/ Principle	Definition
Overall	<p>The overall score is calculated taking the numeric average for each of the individual scores where:</p> <ul style="list-style-type: none"> • Green is tallied as 3 • Yellow is tallied as 2 • Red is tallied as 1 • N/A is ignored <p>Based on this average score, this element is scored according to the following criteria:</p> <ul style="list-style-type: none"> • Green if the average is greater than 2.75 • Yellow if the average is less than or equal to 2.75 and greater than 2.4 • Red if the average is less than or equal to 2.4 <p>Yellow scores require an exception process in which the project team defends why the project should continue, and senior management must agree. Red scores automatically halt the project until the approach taken is changed to force the score to a level lower than Red.</p>
Easier to upgrade	<p>The primary goal for Continuous Delivery is to ensure that each project incrementally improves the experience of applying changes to each customer's implementation. There are numerous ways that this objective can be met for an individual release. The project team is expected to explain how each project specifically ensures a good experience upon application to a customer's environment.</p>
Update database without downtime	<p>Customers require system availability 24 hours a day, 7 days a week. Data migration must not require the system to be down for any period of time.</p> <ul style="list-style-type: none"> • Inline migration is one method for achieving this. Migration of data should occur when the data is next accessed. For example, if a renewal row is to change, we change the data when the underwriting renewal process touches the affected table for the first time • Batch data migration is a secondary method and is discouraged unless there is no other efficient way to accomplish the data movement

Table 2: Guiding Principles Descriptions

The Guiding Principles Scorecard is completed by the project team and a variety of shared services. It is then presented to various department leaders to ensure that the plans for creation of and changes to application functions are in line with the goals of Continuous Delivery. These principles will evolve and change over time. This first set of principles focuses on changing the infrastructure of the application and its coinciding installation and maintenance processes to:

- Enable better separation of core application components
- Increase manageability of change
- Ensure an ever-improving upgrade experience for CSC customers.

Quality and Performance along the Assembly Line

The continuous progress within any given project includes the successful demonstration of achievements as one moves project artifacts from step to step within a phase as well as in between individual phases. These can include a variety of processes like peer reviews, execution of test plans, performance impact analysis and, when required, performance tests measured against identified targets. These processes are in place to ensure that project objectives are met without sacrificing quality or performance.

One can essentially think of releases as pieces of work that go through an assembly line, along which treatments are applied to ensure:

- Solutions meet requirements
- Releases align with published standards for architecture, coding, etc.
- Performance is measured against documented and approved performance expectations
- Demonstrable stability and quality are verified by an independent testing organization (see Figure 8 for overview).

All reported defects are added to the work list. CSC interacts with customers and internal stakeholders to select items from the work list for addition to projects. Projects result in completed releases. Because the unifying concept for a project is likely to be functional in nature, some projects may result in multiple releases that introduce a change into multiple subsystems to achieve the desired results. The desired results may be transferable to other subsystems in the future. However, any subsequent dependency must not require installation or rely on changes in a subsystem that a customer may choose not to purchase or upgrade. This keeps our individual releases segmented by application according to the Guiding Principles.

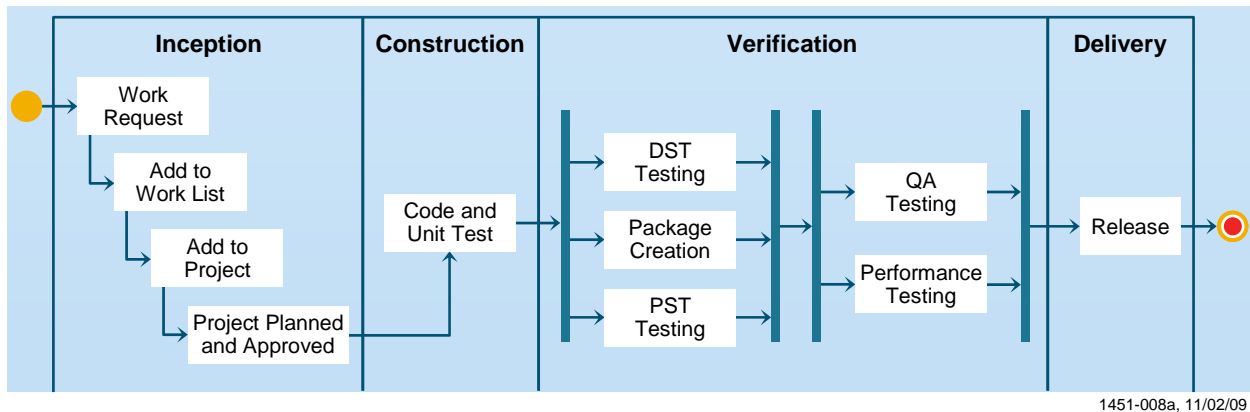


Figure 8: Overview of the Core Continuous Delivery Process

Summary Phase

After Delivery, we transition to the Summary phase, which includes specialized processes for several classifications of work, broadly separated into two main categories: planned and unplanned. Planned work includes user group enhancements, projects sponsored directly by customers as a part of CSC's Innovation Community, normal warranty and enhancement work, and custom modifications per customer request. Unplanned work includes defect resolution and emergency fixes. The process for emergency fixes permits early release to a customer's environment, but still concludes with completion of each of the processes listed above as the fix is released.

We work on multiple releases simultaneously with varying release schedules depending on the scope of work, complexity required for the solution and other variables that directly affect schedule and effort. This permits Continuous Delivery in a manner that sees release rates ebb and flow. The ebb and flow is

determined by customer and market demand for specific fixes and features as well as the availability of resources to complete selected projects. CSC targets a 6-month running rollout for each application. These targets can change as customers and the market dictate a change in priority.

Each release is constrained by a few factors to prevent the scope from growing to a size that creates resistance to migration – and defeats the purpose of the CDR program. Specifically, releases are staffed by teams of 6 or fewer, and elapsed project time is limited to fewer than 90 days. Some projects are smaller than this. And in rare cases, some projects require a larger staff or a longer timeline than this.

Continuous Delivery reflects the bulk of the creation and change activities of CSC's participants. The whole process around Continuous Delivery includes customers in the beginning, middle and end, as this is required to truly be "purple." Continuous Delivery is driven by customers as depicted in Figure 9. Work requests of any type can be suggested by all participants: customers and CSC employees alike. All requests are added to the work list for review. WiRelease manages the work list, provides online views and mechanisms to help prioritize, and groups requests into projects that result in the creation of releases. One should also note that custom modifications, if built by CSC, participate in most of the same process as Continuous Delivery releases. While a custom modification is not a Continuous Delivery release, the process provides quality controls that are critical to all CSC deliverables. There is no shortchanging quality in the process.

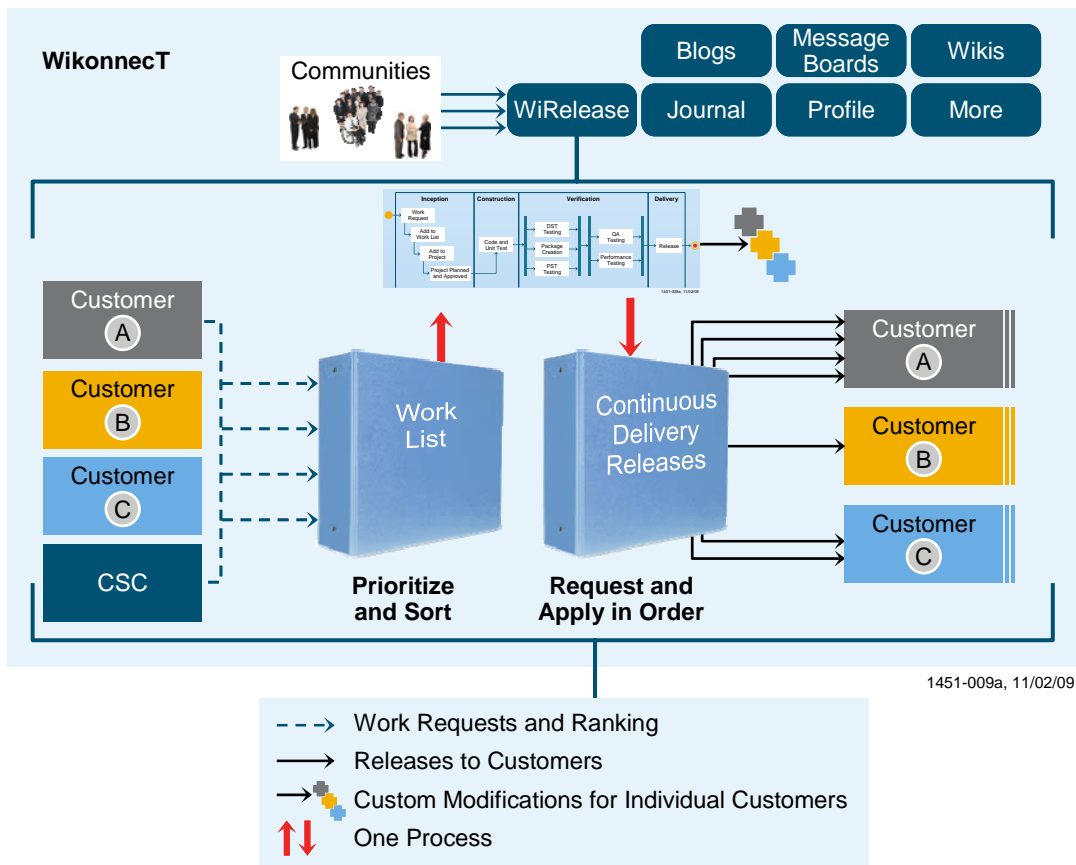


Figure 9: Continuous Delivery is Driven by Customers

Quality

Quality is the single most important targeted attribute for CSC's Continuous Delivery and the single most significant goal of our associated processes. The Quality Office addresses the two major aspects of quality management, which are quality assurance and quality control.

Quality Assurance

The goals of our quality management process are to:

- Specify CSC's quality standards for each release's deliverables and services
- Define the methods, techniques, standards and processes that will be used to fulfill both CSC's and the clients' quality standards
- Ensure that quality management activities are planned and executed
- Objectively verify that critical processes and Continuous Delivery activities adhere to applicable standards, processes, procedures and requirements.
- Inform stakeholders of quality management activities and of the status of quality
- Ensure that a defined escalation process exists and is followed to bring any noncompliance issues not resolved by the project team to the attention of senior management.

Quality assurance is concerned with how well the project conforms to the documented Quality Management processes throughout the entire project life cycle. Quality assurance activities are integrated into the very fabric of Continuous Delivery.

In the Inception stage, quality assurance activities include:

- Monitoring and validating peer reviews on requirements
- Delivery assurance review of project estimates to ensure use of standard estimating templates and approved estimating techniques
- Review and approval of project schedules to verify adherence to best practices and viability of plan
- Review and approval of project management plan/project brief
- Mentoring on quality assurance best practices
- Monitoring and tracking defect reporting.

In the Construction stage, quality assurance activities include:

- Monitoring and validating peer reviews on specifications, code and test plans
- Review and approval of test strategy document
- Verification of customer review of test plans
- Mentoring on quality assurance best practices
- Monitoring and tracking defect reporting
- Verification of phase entry and exit criteria.

In the Verification stage, quality assurance activities include:

- Monitoring defect tracking and reporting
- Mentoring on quality assurance best practices
- Verification of phase entry and exit criteria.

In the Delivery stage, quality assurance activities include:

- Quality assurance final approval on every release
- Monitoring and tracking defect reporting
- Mentoring on quality assurance best practices.

Quality Control

Quality control is a function of the Quality Office Testing Team, which is kept intentionally independent of the development team. The goal of this team is to reduce the risk of unacceptable code being released to

production. It accomplishes this by validating and verifying the proper execution of defined testing-related SDLC processes.

The Quality Office Testing Team participates in the SDLC by executing planned, systematic activities (i.e., requirement reviews, test planning, test cases, peer reviews, defect tracking, automating scripts). This ensures that software processes and products adhere to approved standards, requirements and procedures.

The Quality Office Testing Team takes a four-pronged approach emphasizing:

1. Focused testing of only those things that changed or affected critical functions
2. Integrated, end-to-end testing of workflows and complex business scenarios
3. Simulated customer configurations
4. Regression testing, both manual and automated.

Quality Office testing will not only try to reduce the risk of unacceptable code (i.e., defects) reaching production but also verify that the release under review does not negatively affect existing code. This will be accomplished by a variety of testing techniques including, but not limited to, negative testing, boundary testing, regression testing, end-to-end testing and integrated testing.

For example, the end-to-end tests will incorporate workflows and complex business scenarios that represent a much more accurate simulation of how our customers use the system. The integrated tests will focus on application interaction by having test cases that include more than just one application. In addition, the Quality Office Testing Team will execute its Test phase in an environment that includes previous releases. The Quality Office Testing Team engages the customer community by soliciting examples of specific scenarios and by sharing test strategies, test cases and test results. Customer community input is used to expand the Quality Office Testing Team test case bed.

Remember — to improve the quality of the releases we deliver to our customers, we focus on testing “purple,” which means that we focus on customer experiences and successes. With this in mind, we enhance our test bed to include customizations and test data that more closely match those of our customers.

For Continuous Delivery to be successful, the duration required before the customer experiences value from a release must be reduced. This includes the amount of time spent testing, one of the most time-consuming activities within the SDLC. By improving the overall quality of releases, we can reduce the duration of the customers’ test periods and improve their time to value.

Metrics for Success

Metrics have been defined for each life cycle phase to provide visibility into key project aspects and to drive continuous improvement. For example, defects are measured by life cycle phase to determine if an adequate proportion is being driven out as early as possible:

- Percent of defects found during Requirements – target is 20 percent
- Percent of defects found during code/unit test – target is 35 percent
- Percent of defects found during integration and acceptance test – target is 20 percent

Other metrics covered include:

- Age/duration of projects in each life cycle phase
- Success/fail test execution ratios per project
- Percent of impacted function test coverage – target is 100 percent
- Number of defects per thousand lines of code (KLOC) reported before release
- Percent of active customers installing releases
- Percent of packages following Continuous Delivery process – target is 100 percent

Customer Impact and Benefits

Continuous Delivery offers customers increased visibility into features being selected for release and, more importantly, directs influence into selecting those features. Customers request releases just as they did in the past. However, the releases are smaller and are specifically targeted to permit incremental addition to a customer's existing environment or implementation. Customers will gradually experience a shared code base that is reflective of a much richer code base with clear delineations between that which is customer specific and that which is not. Since the delineations are clear and managed, customers can more effectively introduce changes from CSC into their environments.

The increased transparency allows customers to see incremental progress for each item of interest. WiRelease provides monitoring of projects through the process much like many of us have seen from providers of physical package delivery. This permits customers to see what is coming and if it's progressing as planned. Thus, customers can plan and adjust plans accordingly.

Managing Change Effectively

Continuous Delivery customers get a managed upgrade experience. Customers pulling releases into their Exceed or POINT IN environments know that they are incrementally adding function that is shared by others with similar interests, and that it is supported and warranted. Customers still need to determine how they wish to set up and manage the various environments. By doing so they verify readiness for the movement of changes from initial installation and verification through their quality assurance processes, and subsequently into production. This increased transparency for processes, project content and explicit documentation of dependencies, as detailed by the guiding principles, enables customers to more effectively introduce change, which is after all, the single driving factor for CSC's Continuous Delivery.

References

- [1] Rapid Development: Taming Wild Software Schedules, Steve McConnell, Microsoft Press, 1996.
- [2] Integrating Agile Development in the Real World, Peter Schuh, Charles River Media, 2005
- [3] Agile Project Management with Scrum, Ken Schwaber, Microsoft Press, 2004.
- [4] ITIL Foundations, James Pengelly (ed), GTS Learning, 2004.
- [5] Managing Infrastructure using ITIL, Lata Arora, Shaila Singh, and Poonam Sharma, SkillSoft Press, 2006.
- [6] Continuous Integration, Martin Fowler, <http://www.martinfowler.com/articles/continuousIntegration.html>

This documentation contains trade secrets and highly confidential information that are proprietary to Computer Sciences Corporation. The use, reproduction, modification, distribution or disclosure of the documentation, in whole or in part, without the express prior written permission of Computer Sciences Corporation is strictly prohibited. The existence of this notice shall not cause, nor be construed as causing, any part of this documentation to be a published copyrighted work or to be in the public domain. This documentation is also an unpublished work protected under the copyright laws of the United States of America and other countries. If this documentation becomes published, the following notice shall apply: Copyright 2009, Computer Sciences Corporation. All Rights Reserved.



BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING

WORLDWIDE CSC HEADQUARTERS

THE AMERICAS

3170 Fairview Park Drive
Falls Church, Virginia 22042
United States
+1.703.876.1000

EUROPE, MIDDLE EAST, AFRICA

Royal Pavilion
Wellesley Road
Aldershot, Hampshire GU11 1PZ
United Kingdom
+44(0)1252.534000

AUSTRALIA

26 Talavera Road
Macquarie Park, NSW 2113
Australia
+61(0)29034.3000

ASIA

139 Cecil Street
#06-00 Cecil House
Singapore 069539
Republic of Singapore
+65.6221.9095

ABOUT CSC

The mission of CSC is to be a global leader in providing technology-enabled business solutions and services. With the broadest range of capabilities, CSC offers clients the solutions they need to manage complexity, focus on core businesses, collaborate with partners and clients and improve operations. CSC makes a special point of understanding its clients and provides experts with real-world experience to work with them. CSC is vendor independent, delivering solutions that best meet each client's unique requirements. For 50 years, clients in industries and governments worldwide have trusted CSC with their business process and information systems outsourcing, systems integration and consulting needs. The company trades on the New York Stock Exchange under the symbol "CSC."

ABOUT CSC IN FINANCIAL SERVICES

CSC distinguishes itself through its time-tested ability to plan, build and operate highly reliable, efficient and secure business and IT solutions for leading financial services firms around the world. To complement its capabilities in consulting, systems integration and outsourcing, CSC brings financial services industry knowledge and experience, a comprehensive portfolio of financial services application software and an extensive network of industry and technology partners. More than 10,000 CSC employees are dedicated to serving financial services clients, which include more than 1,200 major banks, insurers and investment management and securities firms.

www.csc.com