Practical Experiences with Software Factory Approaches in Enterprise Software Delivery

Alan W. Brown, Ana Lopez Mancisidor, Luis Reyes Oliva
IBM Rational
Sta Hortensia, 26-28, Madrid, Spain
(alanbrown, ana.lopez, luis.reyes)@es.ibm.com

Abstract—**There are many pressures on software delivery organizations to produce more software faster in the context of extreme cost pressure and growing globalization of the software delivery organization. The concept of a *Software Factory* is beginning to emerge as one way to address these challenges. This paper discusses the principles of software factories for enterprise software delivery using practical examples that explore the how software delivery quality can be managed across the software supply chain. In particular, we discuss two case studies where large commercial organizations have achieved significant improvements in software quality when adopting a software factory approach. We conclude with a set of observations that highlight where this work can be usefully refined and extended.**

Keywords – software engineering; software process improvement; application management; software delivery

I. Introduction

Many companies have experienced a great deal of change over the past few years due to evolution of the business environment, financial upheavals, societal changes, and technical advancement. Key to addressing these changes has been analysis of core business processes to see how they can be refined and optimized, followed by a restructuring of those business processes to better meet the new context. This business process reengineering has helped to refocus on the most compelling and valuable aspects of the business, and is a first step in readjusting investment priorities toward those business activities that are considered essential, while looking to divest those considered secondary [1].

At the same time all IT groups have been forced to lower operating costs across the organization. The direct implication is that they must not only minimize waste and inefficiency, but increase productivity and relevance to the businesses they serve.

This combination of business process restructuring and close focus on delivery efficiency have been seen in many business domains, and have resulted in techniques such as "lean manufacturing", "supply-chain management", and "product line engineering". The application of these ideas in software delivery is what we refer to here as a "software factory approach" to enterprise software delivery [2, 3].

In this paper we examine this view of enterprise software delivery. We first explore the idea of the "software supply chain" and introduce the concept of the software factory. We then detail the characteristics of the software factory approach, and illustrate those concepts using real world examples. We conclude with several key observations.

II. Elements of a Software Factory Approach

Analogous with changes in the industrial sector, a software factory approach to enterprise software delivery aims to reduce time to market for new products, increase flexibility and agility in component assembly, and reduce costs of production while increasing quality and end-user satisfaction. It is important to highlight several key elements of such an approach that impact enterprise software delivery.

A. Aligning business and engineering

A software factory approach to enterprise software delivery requires a well-established, multiplatform process with tooling that aligns business strategy with engineering and system deployment. Critical in building applications that meet the needs of the customer, such processes can help to identify business needs and stakeholder requirements, and drive those business goals into enterprise software delivery projects and solutions, ensuring that the final product meets the business objectives with the lowest possible cost and highest possible quality.

B. Automating processes and tasks

Automating the enterprise software delivery lifecycle can help reduce errors and improve productivity, leading to higher quality products. An integrated portfolio of tools can help teams automate specific, labor-intensive tasks—similar to the way automation is used to perform repetitive manual tasks in manufacturing. Using automation, practitioners are able to focus on creating more innovative solutions with industry-leading design and development environments that help support the delivery of high-quality, secure and scalable products. Companies that invest in automation and a more efficient means of production and delivery can experience a sizeable jump in productivity, quality, time to market and scalability.

C. Leveraging assets across the enterprise

Modern architectural and product development frameworks can be considered complex supply-chains that integrate third-party, custom, off-the-shelf and outsourced components in the overall software or system. This has led to approaches such as Service-Oriented Architecture (SOA) frameworks, which focus on assembly of standard components to promote reuse across the enterprise, and more generally to product line engineering (PLE), where the focus is strategic reuse in developing portfolios of similar products that share many components but are differentiated by variations in features and functions [4].

The first step is to understand what assets exist and leverage them to create reusable components to extend architectural frameworks in meaningful, predictable ways.

D. Supporting integration and lean processes

Today's enterprise software delivery teams can be highly distributed geographically. Consequently, they need flexible and agile processes with real-time collaboration, integrated across disparate platforms, roles and geographies to reap the benefits of modern software and systems frameworks. Globally distributed development can be facilitated through defined, customizable processes and best practices to support flexibility, mitigate risk with comprehensive quality management and enhance developer productivity through task and process automation.

E. Automating operational measurement and control

To help ensure predictable outcomes, the enterprise software delivery process must be governed so it can be continuously measured and improved. A fundamental aspect of this is the definition and codification of processes for developing products. These processes and best practices are corporate assets, and need to be captured in an actionable form so teams can be guided to adhere to appropriate best practices through automated workflows.

Relevant metrics should be gathered automatically at each step, including after software and systems are delivered into production. By constantly, automatically measuring the specific key value aspects of processes, these metrics can provide insight into the efficacy of existing processes and identify areas for improvement. Automated measurement and control is also critical in tightly regulated industries, such as government, aerospace, medical or financial sectors.

III. REALIZING A SOFTWARE FACTORY

Realizing a software factory requires a blueprint to organize and structure the methods and tools that deliver the capabilities to make this real.

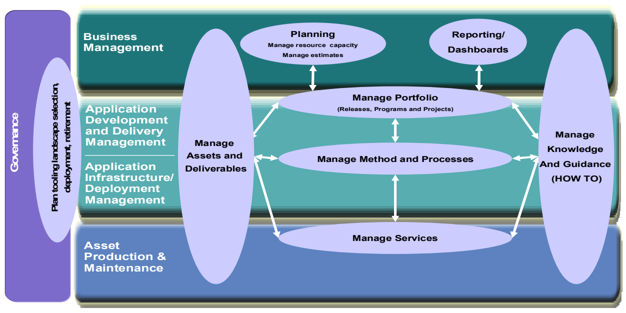

Governance
Plan tooling landscape selection, deployment, retirement
Business Management
Application Development and Delivery Management
Application Infrastructure/ Deployment Management
Asset Production & Maintenance
Manage Assets and Deliverables
Planning
Manage resource capacity
Manage estimates
Reporting/ Dashboards
Manage Portfolio
(Releases, Programs and Projects)
Manage Method and Processes
Manage Services
Manage Knowledge And Guidance (HOW TO)

Figure 1. A Simplified Blueprint for a Software Factory.

Figure 1 illustrates a simplified software factory blueprint that we have used as the basis for several large scale enterprises. In this approach the software factory provides a collection of capabilities that support the management and delivery of enterprise software, covering 5 key areas. We briefly review each of these in turn.

A. Business management

Effective business and IT planning and portfolio management helps to streamline the business by empowering faster, better-informed decisions, and can reduce costs by prioritizing enterprise software investments to support business goals. Ultimately, proficiency in this area allows strategic intent to be converted into executable processes with measurable business results. To implement this typically requires several elements:

- **Enterprise architecture management** to help make faster, better-informed strategic and tactical decisions, prioritize enterprise software investments to support business goals, and analyze, plan and execute change with reduced risk.
- **Business process management** to help to optimize business performance by discovering, documenting, automating, and continuously improving business processes to increase efficiency and reduce costs.
- **Requirements definition and management** to minimize the number of inaccurate, incomplete, and omitted requirements. This helps teams collaborate effectively with stakeholders, reduce rework, accelerate time to market, and respond better to change.

B. Asset production and maintenance

Knowledge management and reuse best practices allow organizations to discover and leverage existing data and assets. With an understanding of the key assets, it is possible to enforce policies and best practices, manage model dependencies and even trace assets to versioned artifacts.

It is important to determine what assets exist by providing the ability to search and select across multiple asset repositories and data warehouses, relate assets to one another and leverage existing assets for reuse. Such solutions can also help administrators enforce policies and best practices, manage model dependencies and trace assets to versioned artifacts, creating a link between systems, sub-systems, code, requirements, test cases and delivered solutions. Finally, teams create new assets, transforming code into standardized artifacts such as Web or Business Process Execution Language (BPEL) services that can be used as components for building value-added applications.

C. Application development and delivery management

Smart product design and delivery optimization requires collaboration across teams to deliver quality software and systems. In addition, applying lean processes with disciplined teams in focused "centers of excellence" ensures flexibility and facilitates globally distributed enterprise software delivery. Collaborative services, automation and measurement feedback throughout the software development lifecycle are essential to achieve levels of productivity and consistency beyond those accomplished using traditional, craft-oriented software development tools.

This requires capabilities that can help teams to collaborate across the lifecycle and automate routine tasks. Key capabilities include:

- **Change and release management.** Improving quality and productivity by effectively unifying distributed teams by managing change processes from requirements gathering through to deployment.
- **Quality management**. Advancing quality across the entire software delivery lifecycle from requirements, design, development, quality assurance, security, and compliance to deployment.
- **Architecture management**. Software development tools for design, development, and delivery that support modeling and coding activities in appropriate high-level languages, supported with a range of analysis capabilities for maintaining the architectural quality of the delivered solution.

D. *Application infrastructure and deployment management*

A modern application infrastructure allows organizations to cost effectively build, deploy and manage applications and products for varying business needs. Integrating service delivery across organizational boundaries and all stages of the lifecycle helps to improve time-to-market and reduce cost and risk while providing the visibility, control and automation needed to deliver a dynamic infrastructure that adapts to changing business requirements. These solutions provide capabilities to help organizations develop a robust application infrastructure, including capabilities for:

- **Product deployment.** Offering services to automatically deploy, track, and manage applications across the lifecycle.
- **Application delivery.** A set of technologies that support system build and deployment across mainframe and distributed environments.
- **Connectivity and application integration.** Services that foster collaboration, insight and cost effective re-use of data and knowledge across the organization.

E. *Governance*

Automated capabilities to monitor operational environments and provide feedback to the software and systems delivery processes are critical in a modern approach. Iterative improvement across the entire lifecycle ensures timely problem resolution and ensures flexibility to adapt to change in today's business environment. These solutions for operations provide capabilities to help organizations develop a robust set of practices for automating operational monitoring and measurement. These solutions can help in:

- Application health monitoring.
- Performance management.
- Security and compliance.
- Service management.
- Performance optimization.
- Monitoring and measurement.

IV. RATIONAL JAZZ: AN INTEGRATED SOFTWARE FACTORY PLATFORM

A common collaborative platform is critical for effectively introducing a software factory approach. A collaborative development platform automates and simplifies the challenges of enterprise software delivery from project management, to the ability to leverage innovation, to the visibility and access of the development and delivery teams across the distributed supply chain.

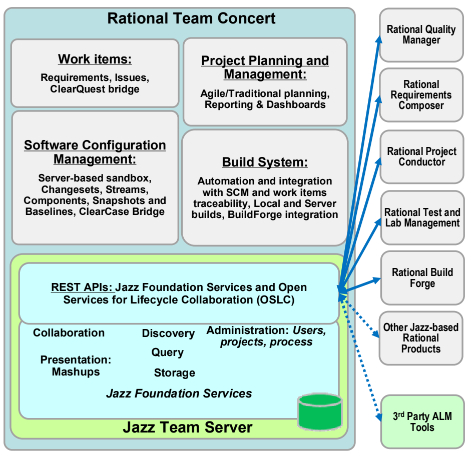

Rational Team Concert
Work items:
Requirements, Issues, ClearQuest bridge
Project Planning and Management:
Agile/Traditional planning, Reporting & Dashboards
Software Configuration Management:
Server-based sandbox, Changesets, Streams, Components, Snapshots and Baselines, ClearCase Bridge
Build System:
Automation and integration with SCM and work items traceability, Local and Server builds, BuildForge integration
REST APIs: Jazz Foundation Services and Open Services for Lifecycle Collaboration (OSLC)
Collaboration
Discovery
Administration: Users, projects, process
Presentation: Mashups
Query
Storage
Jazz Foundation Services
Jazz Team Server
Rational Quality Manager
Rational Requirements Composer
Rational Project Conductor
Rational Test and Lab Management
Rational Build Forge
Other Jazz-based Rational Products
3rd Party ALM Tools

Figure 2. The Rational Jazz Platform.

IBM Rational's approach to provide a collaborative platform for software factories is the Jazz technology – a set of integrated capabilities to unify all stakeholders in the software supply-chain, a basis for governance and management of standardized delivery processes, and the glue that enables visibility and transparency across the complete software delivery process [5].

As illustrated in Figure 2, the Rational Jazz platform consists of a set of capabilities that deliver the services necessary for a software factory. There are 2 major parts to the solution. The first is the Jazz Team Server comprising core capabilities for integrating and collaborating across teams. Access to these capabilities is via the Open Services for Lifecycle Collaboration (OSLC) interfaces. The second is the Rational Team Concert solution that embeds the Jazz Team Server as the basis for delivering core services for work item management, project planning and management, source code management, and build management. Any software factory solution built on this technology customizes and extends this platform through the addition of specific capabilities in areas such as quality management, requirements management, and so on.

The Rational Jazz Platform has been used in several different kinds of scenarios. In particular, were companies are moving toward a software factory approach, this platform offers the core capabilities on which to build and deliver the essential characteristics of a software factory: collaboration, automation, and visibility.

V. Examples

We provide 2 examples [1] of organizations that have implemented a software factory approach, realized via the Rational Jazz platform.

A. *Subcontractor management at ABC Bank*

1) Challenges

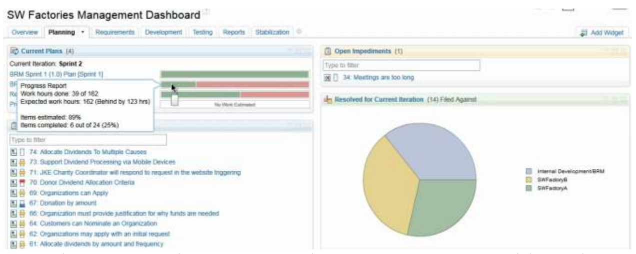
ABC Bank is a large worldwide financial institution with a diverse, widely distributed IT organization. In an attempt to efficiently manage rapid growth at ABC Bank, they have substantially focused on subcontracting major parts of their software development and delivery to large software centers in Latin America, Europe, and Asia. This approach was aimed at reducing fixed IT costs and increasing flexibility to adapt to customer demand. The growth of this subcontracting model is challenging due to:

- **Lack of governance to control project progress.** Across the organization, different teams were using different governance mechanisms, not connected, with progress measured during informal weekly meetings.
- **Poor communication due to different time zones, location, cultural and political differences.** Not all team members were fluent in English language and due to different time zones many discussions were inconclusive, or had to be postponed for days.
- **Inadequate planning and change management procedures.** Projects were subcontracted using a fixed-price model, and there was little flexibility to negotiate changes or modify initial planning.
- **Mismatches between user expectations and the real outcomes.** End users were not involved in requirement analysis or reviews and there were many rejected requests and conflicts across the main stakeholders.
- **Poor infrastructure for remote access and lack of a common asset repository.** Distributed teams were not notified when new versions of the common architecture framework were released and a lot of rework had to be done to adapt these changes at the last minute.
- **Unclear information sharing and privacy rules.** Integrating components from different providers raised many poorly addressed privacy and security issues.

2) Approach

To recover from this situation, ABC Bank directly focused on supply-chain management issues as part of a wider software factory approach to software delivery. They changed their development processes and infrastructure, and implemented a common development environment based on Rational Jazz platform to mitigate hidden costs and issues. In their first wave of changes they focused on:

- **Organization changes** by creating a new Software Factories Project Office in charge of negotiating and managing subcontracted projects;
- **Common infrastructure** based on a central repository, accessed from external locations using standard Internet connection protocols. This central repository is used to share and integrate information across the teams.
- **Governance Dashboards** producing metrics and reports that measure progress of individuals, teams, and software factories to assess their performance. This central governance dashboard was updated automatically with project information in real time, allowing the enterprise to keep external developments under control and to reduce meeting and travel costs.



SW Factories Management Dashboard

Overview | Planning | Requirements | Development | Testing | Reports | Stabilization | Add Widget

Current Plans (4)

Current Iteration: Sprint 2

BRM Sprint 1 (1.0) Plan [Sprint 1]

Progress Report
Work hours done: 39 of 162
Expected work hours: 162 (Behind by 123 hrs)

Items estimated: 89%
Items completed: 6 out of 24 (25%)

No Work Estimated

Type to filter

74: Allocate Dividends To Multiple Causes
73: Support Dividend Processing via Mobile Devices
71: JKE Charity Coordinator will respond to request in the website triggering
70: Donor Dividend Allocation Criteria
69: Organizations can Apply
67: Donation by amount
66: Organization must provide justification for why funds are needed
64: Customers can Nominate an Organization
62: Organizations may apply with an initial request
61: Allocate dividends by amount and frequency

Open Impediments (1)

Type to filter

34: Meetings are too long

Resolved for Current Iteration (14) Filed Against

Figure 3. Software Factories Governance Dashboard.

- **Planning and change management processes** were adapted to augment traditional waterfall software development processes with iterative, agile techniques, to enable faster response to changing demands.
- **Confidentiality** was addressed by identifying every critical private data element that the company had, isolating it from subcontractor access, and explicitly granting permissions for common assets to be shared with subcontractors via a central register of shared artifacts (e.g., common architecture components).

3) Results

Implementing these changes was not easy and caused many political and technical conflicts inside ABC Bank, and across the supply-chain. However, as a result of this transformation the enterprise was able to adapt to this new software delivery model, and is starting to benefit from the reduction of fixed costs and increased flexibility into their development activities across their suppliers.

Thanks to the governance dashboard they are now able to measure status and progress of each supplier, penalizing or terminating contracts of those with less efficient delivery.

B. *Testing Factory Services at XYZ*

1) Challenges

XYZ is a European software services and consulting company, specializing in software development and testing services. One of the company's most important concerns is consistency and quality of delivered services, and a repeatable lean approach to software delivery. To that end, XYZ has standardized many key practices, and has obtained a CMMI Level 3 certification [6]. XYZ's engineering and quality assurance (QA) culture encourages agile practices along the full software lifecycle, making it compatible with the rules and constraints associated with CMMI.

[1] Although the examples are real, we use fictitious name for reasons of privacy.

As part of their global strategy, XYZ offers expertise and services in software testing, and provides a catalog of services to their customers, including:

- Services to assure the quality of the work-products generated in each phase of the software development lifecycle, reducing defects across phases;
- Playing the role of facilitator between the different actors in the QA process to objectively assess quality practices and review project milestones;
- Advising on quality processes and practices, including assessing deliverables, and designing appropriate test and support processes to increase quality.

A software factory approach to XYZ is important because it encourages early project involvement of testers and other quality-focused roles, and makes quality management throughout the project lifecycle a high priority.

2) *Approach*

To realize these needs, XYZ decided to set up several large European delivery centres, and to adopt a software factory approach to deliver its quality-focused services. It refers to these as a "software test factories".

The basis for their software test factories is an infrastructure supporting software configuration management, build management and continuous integration practices, and agile project management. These core capabilities are fully integrated with specific testing services to manage the test plans, execute tests and assess test coverage against the project requirements.

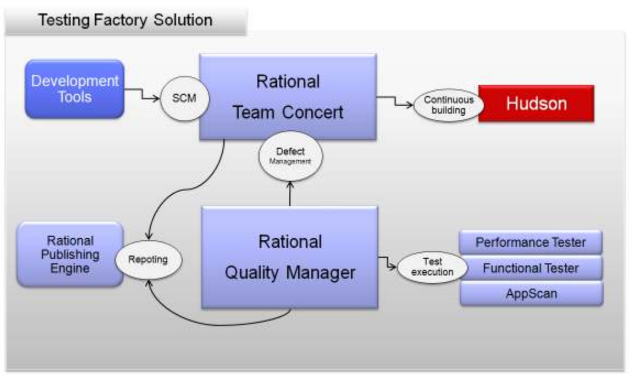
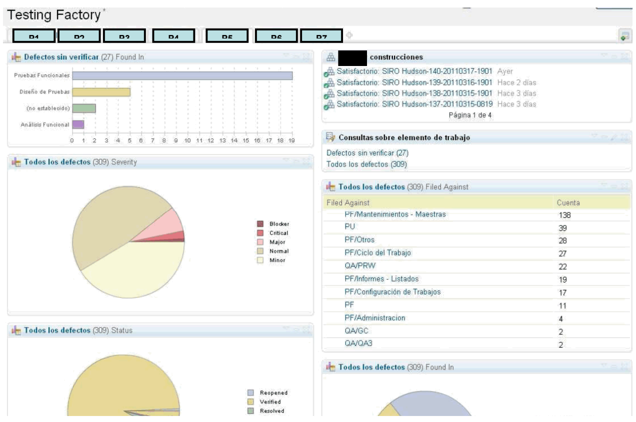


Figure 4. Testing Factory Solution Architecture.

A streamlined software factory infrastructure for QA is the cornerstone of the XYZ solution. This solution, augmented with additional tools for developers or test engineers in their day-to-day work, offers the integrated capabilities that ensure XYZ delivers quality solutions to its clients. As illustrated in Figure 4, the testing factory services were automated by providing a suite of integrated tools covering several process areas of CMMI.

In addition, the need for supporting geographically distributed teams and roles (PMO, Test Manager, Test engineers, etc.) was critical. A series of customized processes were designed that resulted in:

- **Templates** to specify test cases and test scripts to define automated or manual tests;
- **Automation** of building and continuous integration process, regression testing or other quality standards such as those imposed by CMMI;
- **Visual dashboards** to monitor the state of the projects, delayed activities, open vs. closed defects, metrics of coverage, productivity of the team, etc.



[redacted] construcciones

Satisfactorio: SIRO Hudson-140-20110317-1901 Ayer
Satisfactorio: SIRO Hudson-139-20110316-1901 Hace 2 días
Satisfactorio: SIRO Hudson-138-20110315-1901 Hace 3 días
Satisfactorio: SIRO Hudson-137-20110315-0819 Hace 3 días
Página 1 de 4

Consultas sobre elemento de trabajo

Defectos sin verificar (27)
Todos los defectos (309)

Todos los defectos (309) Filed Against

Filed Against	Cuenta
PF/Mantenimientos - Maestras	138
PU	39
PF/Otros	28
PF/Ciclo del Trabajo	27
QA/PRW	22
PF/Informes - Listados	19
PF/Configuración de Trabajos	17
PF	11
PF/Administracion	4
QA/GC	2
QA/QA3	2

Figure 5. A Dashboard from XYZ´s Software Test Factory.

3) *Results*

The software factory approach to testing and quality is a critical part of XYZ´s strategy. As a result of this initiative, XYZ has successfully increased the efficiency of their processes, improving the productivity of their key testing factory in Salamanca, Spain. This solution is now being introduced at test factories across the XYZ organization.

VI. OBSERVATIONS

A. *Agile Software Factories and CMMI*

Is it possible to make agile methods and process maturity compatible? In a software factory context, this goal is not only possible, but mandatory. In all enterprise organizations there are policies and rules which enforce achievement of certain maturity levels. At the same time, the agile paradigm is an emerging necessity to address the challenges of flexibility in software development and delivery.

For many organizations, significant investment has been made to improve the maturity of the key management processes, with the CMMI as a focus for much of that effort. To achieve many operational goals of a software factory, CMMI can be very valuable in areas such as:

- Reducing operational costs;
- Improving the quality of the service;
- Managing capacity and resources efficiently;
- Industrialize the software lifecycle through reuse.

In contrast, the agile methods [7] (primarily oriented to optimize software development teams) are very useful for helping to ground the more abstracts 'process areas' into a more concrete and concise practices which will be used by the development and testing teams in:

- Delivering working software on a frequent basis;
- Promoting whole team planning and face-to-face collaboration;
- Reacting to frequent changes and reprioritizations with rapid fact-based decision making.

At first glance, these agile practices do not align well with more traditional high maturity approaches such as CMMI. However, in our experiences we can adapt these ideas in the context of a software factory model to balance the need for repeatability and governance essential for CMMI with the pragmatic needs expressed in agile approaches. Achieving this balance is critical in enterprise organizations such as ABC Bank and XYZ. A software factory approach provides a framework to realize this.

B. Software Factories in the Cloud

There is a high level of excitement about cloud computing and the promises that it brings to enterprises to reduce cost and increase flexibility of service delivery [8]. Many organizations are already involved in pilots, or are actively using cloud technologies and cloud-based services.

Initially, we have seen many traditional enterprise software solutions ported to a cloud platform, and included in platform images that can be uploaded to a cloud infrastructure. This is an important starting point for enterprise system use on the cloud. However, it is very limited in terms of many of the important usage scenarios for cloud technology. There is less understanding of which new enterprise software capabilities, services, and approaches will be needed in much more complex scenarios. For example, we already are seeing interesting scenarios that are raising new challenges for enterprise software delivery organizations:

- Several teams are deploying business application onto a public cloud infrastructure for access by clients around the world. How do those teams collaborate to share information to ensure that they do place sensitive data on the public infrastructure? What coordination is given to the teams to ensure the management of shared images is handled effectively?
- Multiple System Integrators and specialist vendors must deliver different parts of key enterprise solutions as part of a software supply chain that must be integrated to be delivered into production. How can the cloud be used as the delivery platform to coordinate and govern delivery and integration of these components?

These, and many more such scenarios, are stretching conventional processes, skills and technologies for enterprise software delivery. Software delivery organizations are actively working on new deployment approaches that provide the additional governance, visibility, and control that is demanded in such situations.

C. Metrics and measures for Software Factories

Although there are different development standards to measure in-house development, there is little standardization in evaluating supply-chains and software factories. Standard approaches such as function point analysis and defect density can be applied, but in practice they appear inadequate.

With more complex supply-chain delivery models becoming more common, we need metrics that help us address different questions: Which software factory is more productive? How many defects are still opened? Which software factory is delayed in their deliverables? These and many other measures need to be defined and an automatic mechanism to collect these metrics must be implemented to help compare results across external providers in real time.

VII. SUMMARY

A fast-paced evolution is taking place in the context of very dramatic shifts in how IT organizations view the value they bring to their varied stakeholders, the services they deliver to clients, and the way they invest to achieve their goals. As a result, the last few years has seen a significant change in the way enterprise systems are developed, delivered, and maintained. By introducing a software factory view to enterprise software delivery, organizations can focus attention on the software supply chain, address inefficiencies in software delivery, and gain greater control and visibility into the delivery process.

In this paper we have examined the key principles that underlie this kind of software factory thinking to enterprise software delivery, and provided 2 real-world examples to illustrate how solutions can be introduced that provide value to the IT organization. The technology underpinning such an approach is critical. We briefly discussed one example technology approach based on Rational Jazz platform, and illustrated its primary characteristics as the basis for a software factory.

Much further work remains. We have made a number of observations on critical areas requiring additional work. Over the coming years we expect to see significant progress in these, and in several other key areas.

REFERENCES

[1] J. Jeston and J. Nelis, "Business Process Management, Second Edition: Practical Guidelines to Successful Implementations", Butterworth-Heinemann, 2008.

[2] M. Poppendieck and T. Poppendieck, “Lean Software Development: An agile toolkit”, Addison Wesley, 2003.

[3] M. Hotle and S. Landry, "Application Delivery and Support Organizational Archetypes: The Software Factory", Gartner Research Report G00167531, May 2009.

[4] P. Clements and L. Northrop, “Software Product Lines: Patterns and Practices”, 3rd Edition, Addison Wesley, 2001.

[5] M. Goethe et al., "Collaborative Application Lifecycle Management with IBM Rational Products", IBM Redbook, December 2008.

[6] “CMMi for Development, Version 1.3”, CMU/SEI-2010-TR-033, November 2010.

[7] R.C. Martin, “Agile Software Development: Principles, patterns, and practices”, Prentice Hall, 2002.

[8] G. Reese, “Cloud Application Architectures: Building Applications and Infrastructures in the Cloud”, O´Reilly Press, 2009.