RC25521 (WAT1502-022) February 10, 2015
Computer Science

IBM Research Report

A Virtual Factory Model for Complex Applications Development and Support

Jarir K. Chaar
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
USA

Ralph E. Nelson
IBM Watson Group
Richmond, VA 23236
USA

Research Division
Almaden – Austin – Beijing – Brazil – Cambridge – Dublin – Haifa – India – Kenya – Melbourne – T.J. Watson – Tokyo – Zurich

A Virtual Factory Model for Complex Applications Development and Support

Jarir K. Chaar **Ralph E. Nelson**

Over the last 10 years, the IBM Application Factory has woven together factory-floor assembly models and automation principles, sophisticated scheduling techniques, workflow management, and software quality methods in order to deliver high-quality software. It has progressively evolved to offer a unique approach to industrializing the development and support processes of complex applications through automation in three key areas of application lifecycle management.

- Global governance, achieved by efficiently partitioning and managing the work of a fully productive, globally distributed workforce.
- Lifecycle Acceleration, achieved by leveraging modeling techniques and patterns from requirements through test.
- Labor Acceleration, achieved by capturing expert how-to knowledge at the process activity level for rapid, consistent execution by a globally distributed workforce.

Today, the Application Factory highly integrated platform is the virtual home of more than 25,000 application designers, developers and testers that are responsible for delivering around 100 application development and support services. Its highly disciplined delivery methods have generated an average of 20% to 35% reduction in the costs of implementing these services. Its underlying framework offers adequate capabilities for implementing virtual factory models that leverage a globally distributed workforce to also deliver infrastructure and business processes services.

1 Introduction

In the highly competitive application services marketplace, businesses continue to actively seek new ways to leverage the right skill in the right place at the right price. The post-Y2K era has witnessed an increased reliance on global resources in implementing complex applications. As with other services, in the **first wave** of the move towards global delivery, the emphasis was on reducing the cost structure by building a global workforce. The competition among the major industry players generated the need for aggressive differentiation that went beyond the benefits of lower cost – improving quality and reducing time-to-market became key differentiators.

In practice, traditional project management techniques that worked well with small co-located teams did not seem to scale well to a global workforce. Strong evidence suggests that in order to be effective, the development process adopted for global delivery needed to be implicitly partitioned. As such, a **second wave** of innovation led to the adoption of software development and deployment practices that improve quality and increase productivity while preserving the advantages of the global workforce[29, 45, 49, 68, 69].

As the application services market place has matured, it has become increasingly apparent that the haphazard use of outsourcing is a double-edged sword, which can quickly become a liability unless managed properly. Starting in 2005, IBM actively invested in standing up the Application Factory[29] with the stated goal of addressing the above problems by leveraging proven IBM methods, tools and best practices in a standardized automation environment. Application Factory applies factory floor assembly models and automation principles to the domain of geographically distributed software development and management. The adoption of the factory model to global delivery has resulted in new organizational efficiencies, reduced process variance and improved execution discipline.

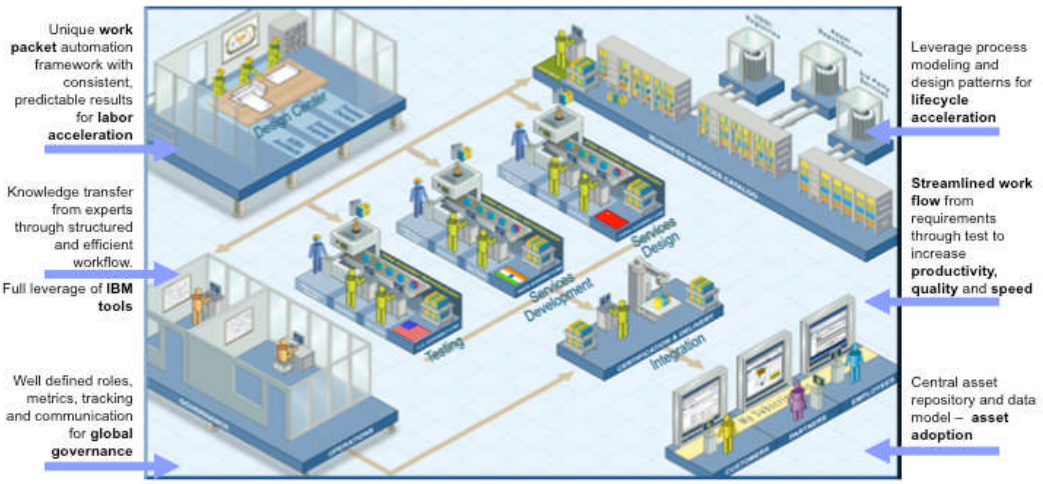

Figure 1. The Application Factory Structure

2 The Factory Structure

The Application Factory[29], depicted in Figure 1, offers a structured way of distributing work to remote teams that promotes collaboration and enforces good governance practices. Some of the key organizational entities in the Application Factory include ***governance and operations teams, functional design teams*** that build the architecture in support of Client requirements, and ***technology assembly centers teams*** that implement and test the application[36].

The Application Factory assumes many of the foundational elements of an organization, and therefore needs governance that integrates into existing governance models at the Client I/T shop. Governance enables the Application Factory to execute highly flexible and business driven processes in the context of a structured and compliant environment.

The Application Factory governance model augments Client processes with business direction and strategy for the factory and ensures that goals and objectives are clearly defined and measured. Financial and portfolio management are assigned to the factory governance team. As such, this team specifies the principles, policies and procedures that are carried out by the factory operations teams, aligns IT investments in support of the business directions and long term innovation needs, and, defines the IT policies and procedures necessary to sustain innovation and meet business goals[7].

The operations teams manage the daily activities required to meet business goals, while complying with the policies and procedures set forth by the governance team. Their delivery management activities include assigning work to the teams in the technology assembly centers; activities that entail managing the factory resources based on skills, seniority and availability[8,9].

Like in any factory, supply and demand planning is a critical success factor. During its annual budgeting process, the factory governance team works closely with project leads to understand their plans and staffing needs, including the number of resources, experience, skills, and training required to successfully deliver projects. Throughout the year, the operations teams regularly update the factory capacity model to show the consumption of committed resources by role for each project, the available capacity, and the anticipated forecast for the year. Their model is used to adjust the factory supply and demand quarterly projections.

The functional design teams combine a mix of skills that range from the Business Analysts responsible for collecting and formalizing the business requirements from the lines of business, to the IT Architects that translate these requirements into scenarios, use-cases and high-level designs. Design teams often include senior Client representatives that are actively involved in articulating the business requirements, validating the high-level designs before they are implemented by the technology assembly centers, and, leading the integration of new application components with their enterprise architecture.

The identification, creation, management and reuse of assets are key productivity and quality levers in software development. The last decade has witnessed major advances in transforming the middleware layer of the IT stack into an assortment of standards-based assets. Today, corporations are focused on transforming monolithic applications into more granular, SOA-based assets. The functional design teams manage this transformation and take responsibility for cataloging, indexing, provisioning, versioning and promoting the use of all factory process and service assets.

The technology assembly center is a highly structured, team-based, global delivery organization that combines the advantages of software development teams (stable and highly-productive) and those of services engagement teams (dynamic and highly-responsive), to deliver faster, better and more cost efficient services. It employs a framework that ensures a rigorous implementation of software engineering practices in carrying out detailed design, coding and test of custom and package applications. This is achieved by deploying common processes, methods and tools across projects and mandating the usage of assets and accelerators as part of the development process.

A person responsible for managing the careers and work assignments of technical resources leads the technology assembly center team. These resources range between senior developers with expertise in the functional domain served by the team and the more junior developers they supervise. For custom applications, the team will most likely add an IT architect to its ranks. The teams of a specific functional domain are clustered in specialization groups. These groups form virtual communities that span multiple technology assembly center locations. They are led by a subject matter expert that can offer expert help to any member in their specialization team, and, is responsible, through knowledge sharing, for managing the technical vitality and balance of skills across all member teams.

3 The Work Packet Construct

Work Packets represent a standard envelope by which every work order is authored, transported, and delivered. In order to coordinate the different pieces of work that constitute a global delivery project, the Work Packet must precisely define the work to be done, including such aspects as the tasks, input and deliverable work products, required team structure (e.g. roles and skill levels) and necessary guidance (e.g. methodology, policy and best-practices). In addition, the Work Packet defines sensors in the form of metrics and alert tables that monitor the runtime process for early prediction and detection of trouble and its immediate well-governed resolution[14, 16].

Work Packets represent process building blocks that are assembled into Project Templates[10]. Phases that implement independent sub-processes within the overall processs can also be assembled from Work Packets. Work Packet scope might involve performing a single development activity on a fully integrated system or a multitude of activities on a single component of a system. Each Work Packet constitutes a subset of activities that are bound to a larger project plan work breakdown structure (WBS) managed through a traditional project management tool[33, 40]. Experience shows that limiting the effort associated with a single Work Packet to a maximum of 40 person-hours and eliminating interdependencies between Work Packet activities can improve the productivity of teams and the quality of delivered functionality, while substantially reducing variability in executing project plans.

Technology assembly centers support multi-location delivery, enabling g*eo-political risk mitigation and time zone coverage.* Service requests initiated by the functional design team are directed to the appropriate technology assembly center where the complexity of the request is estimated and available resources are identified based on the roles, skills, activities and activity durations, as specified by the Work Packet associated with the service. Work Packets enable running multiple, geographically distributed technology assembly centers concurrently to maximize the throughput of the factory and realize cost efficiencies[25, 32].

Work Packets also provide self-guided educational links, normative guidance, linkages to best practices, and pointers to required manuals and tutorials. They endow practitioners with greater expertise through

the capture of arcane Knowledge, known only to those who have been assigned the same specific set of tasks on previous projects. Work Packets are the preferred means to support the rapid increase in the number of skilled practitioners in technology assembly centers; they provide practitioner mentoring and feedback formalization and accelerate the practitioner educational process and the development of skills.

Application Factory uses two basic techniques for expertise capture and normative guidance formulation; design patterns analysis (Figure 2), and exemplar project plan and work breakdown structure analysis (Figure 3). Both techniques start with the initial step of identifying practitioners specific to a domain, and establish linkages with mentors in both the technical and functional areas.

Interview sessions are used to establish how assigned practitioners went about implementing their tasks. This effort includes task enumeration and links to any manuals used in the delivery of the tasks. Links to tutorials on possible tool usage are also provided. If Work Packets are associated to specific I/T governance tasks, links to client methodology work products required for this task are also included.

The end goal of both techniques is to build a body of normative guidance around tasks associated with a specific expertise domain. Normative Guidance consists of a process, content, and delivery mechanism, and identifies

- Best practices that are prescriptive and captured in a form such that they can be delivered through existing tooling, documents, web sites, etc. where they are most likely to be used.
- Statements that are prescriptive and affirm how actions should be performed and assessed.

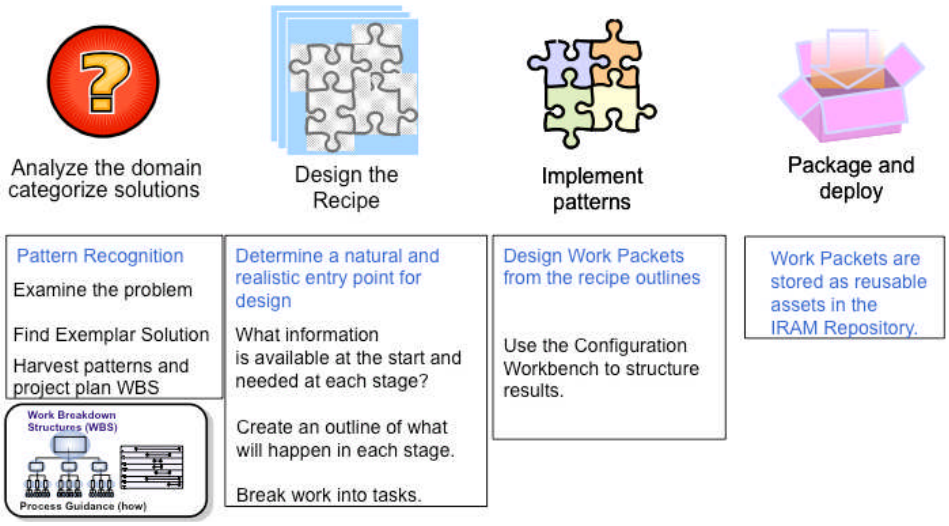

Figure 2. Expertise Capture using Patterns for e-Business Analysis

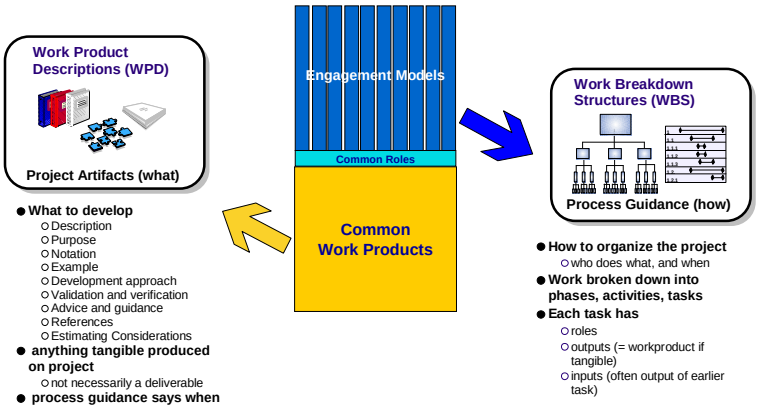

Figure 3. Expertise Capture using Exemplar Work Breakdown Structures

4 The Factory Delivery Model

Business-driven application development and support poses new challenges to traditional application lifecycle management processes. It requires very flexible environments in support of the rapid tailoring and adoption of the process itself. Furthermore, it expects consistency in duration, effort and quality of outcomes when performing the steps of established processes. Properly managing resources is further complicated by the need to manage a large globally distributed workforce with highly specialized skills and heterogeneous cost structures.

The Application Factory enables business-driven application development and support by offering a rich library of services (Figure 4), defined as Work Packets[35], that form the building blocks of the actual processes that are used in developing new applications or major enhancements to existing applications, and in managing incidents with bug fixes and minor updates[24] (Figure 5). These processes support account-specific and/or project-specific variations in tools[28, 31] and methods while preserving the traceability and monitoring requirements necessary for the factory[13, 17, 30]. Their operational data is leveraged to continually improve the accuracy of estimating the duration and effort of the Application Factory services[26, 27, 34].

Application Factory supports two classes of processes; complex, long running application development processes and a much larger number of simple, short running application support processes. In general, development processes are composed of multiple Application Factory services, with the duration and number of resources in assigned cells a function of the complexity of the work performed, and the number and dependencies between services governed by an overall project plan[6, 15]. On the other hand, support processes are restricted in scope to a single Application Factory service, with the duration and the number of resources in assigned cells a function of the severity of the incident and contractual service-level agreements[1, 3, 4]. In both cases, one or more members of a cell or one or more cells may be dedicated to a single Client. The component services of a process can be pulled from a global pool by one of the many teams that can perform their activities, or can be pushed to specific teams when their activities require special skills[12].

Custom Applications Development Services (14)
Services Design | Service Development | ESB Implementation | Java CAD (Application Development) | Microsoft CAD (Application Development) | eCommerce Service Implementation with WebSphere and Sterling DOM products | Learning & Content Development | UI Development (Portlet, JSF, JSP, ASP.NET) | Security & Privacy | WebSphere Portal Implementation | Mobile | Business Process Management (BPM) for WebSphere Process Server, Lombardi and PEGA Products | Curam | Cloud Enabled Packages | Language Translation Services
Information Management Services (7)
ETL-Data Stage | ETL-Informatica | Analytics – Cognos | Master Data Management | Documentum & Filnet | Ab Initio |Teradata Service
Custom Applications Support Services (7)
DBMS | Mainframe | AS/400 | Open System | Java AMS | Microsoft AMS | Maximo
Oracle Applications Development and Support Services (19)
eBS RICE | eBS Upgrade | eBS Admin | PeopleSoft RICE | PeopleSoft Upgrade | PeopleSoft Admin | Siebel RICE | Siebel Upgrade | Siebel Admin | Siebel Testing | Fusion Middleware | JDE RICE | JDE Upgrade | JDE Admin | OBIEE Implementation | Hyperion Implementation | Oracle Edge & Industry Apps: Retek and Billing and Revenue Management (BRM) | Oracle Functional Services (Functional AMS support in Oracle Service Line Product Groups (eBS, Siebel, PeopleSoft, JDE, OBIEF, Hyperion and Oracle Edge and Industry Apps)
SAP Applications Development and Support Services (13)
ABAP (Reports, Interfaces, Conversions, Enhancements, Forms) | XI / PI (SAP Middleware Services) | SAP Upgrade | BI | Basis & Security | Portals | SAP Mobile | SAP Testing | SAP Functional Services (Functional Design & Build and AMS support for F2R, H2R, O2C, P2P and R2R services)|SAP (Manufacturing Integration & Intelligence) MII
Applications Testing Services (8)
Full Lifecycle Testing | Test Automation | Performance Testing | Mobile Testing | BAO Testing | eCommerce Testing | Test Assessment | Specific Services (for specific products that currently include CAR, DAS, TPOW and Archetest)

Figure 4. The Application Factory Library of Services

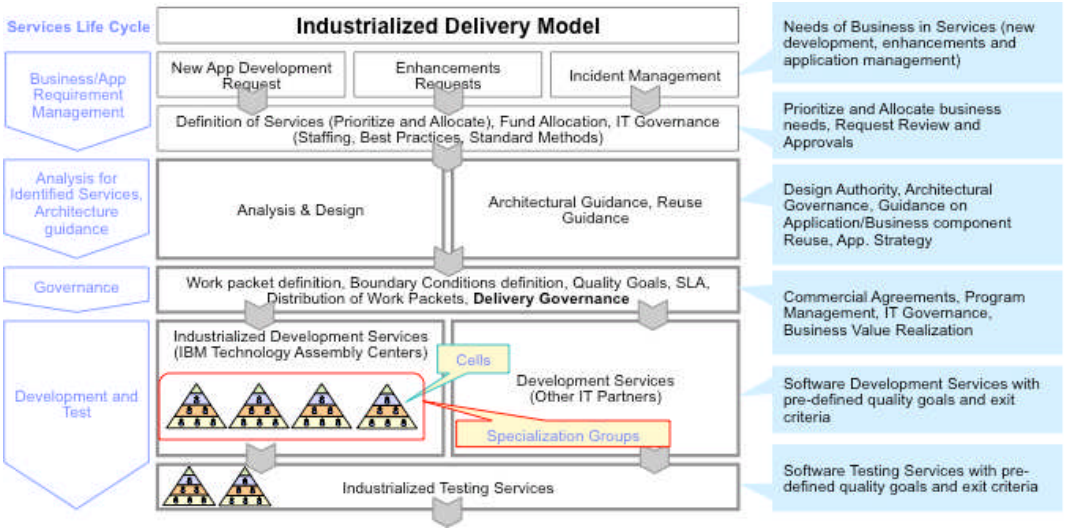

Figure 5. The Application Factory Industrialized Delivery Model

Industrialized delivery is enabled by coupling Application Factory services with built-in quality control in the form of Entry and Exit Criteria, such as Critical Design Reviews (CDR) and Test Readiness Reviews (TRR)[9, 20, 22, 23, 43, 51]. This ensures that the quality of the deliverables associated with a service always exceeds a specified target threshold.

4.1 Managing Development Projects

The Application Factory deploys a flexible and extensible project management framework in support of two fundamental changes in its development models. First, the execution of individual projects is broken up between globally distributed teams whose work is managed through Application Factory services and their associated Work Packets. Each team operates locally in what it perceives as a separate project – Application Factory will provide the balance in managing local issues with global project, program, enterprise, and customer concerns. Second, as the factory manages for an enterprise its entire application portfolio, it must also manage dependencies and balance priorities between all ongoing and planned projects[37]. Each project may impact different enterprise applications and cross-organizational boundaries within the enterprise and beyond[38, 39].

The Application Factory is focused on efficiently partitioning and managing work across the boundary of distributed teams. As such, there is significant emphasis on streamlining overall operations through structured, consistent workflow-driven automation, lifecycle acceleration through reusable templates that capture expertise knowledge and efficient usage of the skills and available capacities of participating functional design teams and technology assembly centers. This focus makes the factory project management framework distinct from available project management solutions, which have not necessarily been designed for managing a large, global workforce. Nevertheless, the Application Factory has leveraged the capabilities of traditional project management tools, such as Microsoft (MS) Project in streamlining global delivery.

At its core, the factory project management framework maintains a work breakdown structure that represents how coarse granularity work items are progressively refined, based on Application Factory services, into more atomic activities to be assigned to teams or individuals. These activities are modeled as work items in Rational Team Concert (RTC) -- a process orchestration tool for Agile development. Along with this structure comes a set of associated attributes such as, due dates, percentage work completed, and other mechanisms to govern work and keep track of progress (e.g., metrics, alerts and resolutions). The factory project management framework enables factory managers to import a set of factory services into MS Project, and as the activities of a service are being performed in RTC, the managers should be able to pull percent completion information to keep track of progress[2, 34]; or when an alert is raised on a factory service in a project that is being managed through MS Project, the alert may have to be escalated to the MS Project view, so that the project manager can proactively respond to it.

The factory framework offers adapters for MS Project that enable tracking the actual progress of planned project activities and updating, at all times, the original plans of the project. An open source project called MPXJ (Microsoft Project eXchange in Java) is used to export factory data in a XML format that MS Project can interpret. Some parts of the factory data model (e.g. work breakdown structure involving project, phase, Work Packet and task) map reasonably well into the corresponding MS Project constructs. However, mapping other concepts such as factory alerts that are not known to the tools is more complex. A standard meta-model for software development methodologies (e.g., the Software Process Engineering Meta-model (SPEM) from OMG[18]) can simplify such mapping.

4.2 Managing Support Activities

The Application Factory also deploys a ticket management framework that aims at achieving compliance with Client contractual SLAs while maximizing the utilization of support teams and minimizing any potential penalties that may result from missing such SLAs[1]. The Application Factory support teams manage the backlog of problem tickets logged against their portfolio of supported applications in Tivoli Service Request Manager (TSRM). A two-way bridge securely regulates the flow of ticket data between a Client's ticket management system and the factory. Direct access to the factory TSRM instance is offered to Clients with no ticket management systems of their own. Bridges between Client premises and the Application Factory are implemented as a number of software agents that monitor the ticket queues[7]. Such queues sort incoming tickets based on a single attribute or a combination of attributes such as application technology (e.g., Java, C++, SAP, Oracle), line of business (e.g., finance, HR, sales), ticket severity, etc. Following ticket resolution, the bridges are used to update ticketing data in the client ticket management system and declare a ticket closed[9].

Following the creation of a ticket in TSRM, the text mining capabilities of the statistical package SPSS are used to analyze the description field in order to determine the problem type (how to, break/fix, minor enhancement) and its trigger conditions. Problem type is used next to select the Application Factory service that will be deployed in resolving the ticket. Complexity of the problem ticket is then used to estimate the level of expertise and time required for performing the service. Severity is used to identify the deadline for resolving the ticket, in compliance with contractual SLAs. The ticket is then added to the backlog of unresolved tickets.

iLOG, a business rules management system, is used next to create or update the overall factory plan for handing ticket backlogs. High severity tickets are immediately scheduled for resolution. For other tickets in the list, iLOG scheduler is run periodically to add the new list of tickets in the backlog to those already scheduled for resolution. Once iLOG allocates a start date and the appropriate resources for handling a ticket, the process template associated with its problem type is embedded into the overall plan maintained by MS Project. The activities in these plans are instantiated as Rational Team Concert (RTC) work items that are performed by architects, developers, testers and artifact reviewers as soon as the conditions for starting them are satisfied.

The actual durations of the process steps performed in resolving tickets are tracked in RTC. These durations are used to recalibrate the process model estimates based on problem complexity. They may also impact the overall schedule if assets are leveraged and result in shorter durations for fixing some tickets or the complexity is underestimated and results in delays fixing others. In either case, iLOG is used to update the timeline for handling the remaining tickets in the backlog[34].

4.3 Managing Technical Resources

The Application Factory resource management process includes managing the skills and work assignments of technical resources in individual centers and the factory as a whole. At the strategic level, the process ensures that functional design teams and technology assembly centers invest in building an adequate supply of resources with the right mix of skills and expertise that can handle the anticipated workload[7]. The process also ensures that the technology assembly center cells expected to perform similar work stay balanced in terms of experience and skill profiles, and suggests resource rotations if needed to

maintain this balance[8]. This is particularly important for predictably performing repetitive work with low variance, which is one of the typical scenarios that benefit from a factory model.

On the operational level, the resource management process plays a crucial role in assigning factory personnel to the activities performed by the technology assembly centers of the factory. This is a complex exercise, which has to make optimal use of skills, availability, cost and quality while complying with any constraints on schedules[15]. The situation becomes even more demanding when there is a need to optimally assign a large number of such activities at a time, and to monitor the progress of ongoing activities in order to respond to the dynamics of the work environment; for example, a resource may suddenly become unavailable, or a task may get delayed, thereby requiring re-planning and impacting the schedule and allocated capacities for all the dependent tasks.

Resources are assigned to activities if they have the minimal skill levels demanded by these activities. Skill levels are determined based on seniority, recent training, the complexity of prior engagements and performance in such engagements. As such, skill matching in the Application Factory is guided by a skill ontology that is consulted when a lower level of skill may be the only one available or a skill match may be "approximate" (e.g., a Java programmer may have to substitute for a C++ programmer, if the latter is not available). The assigned activities can begin on the earliest date the resource is available, once all pre-requisite activities are completed. The availability of a resource depends on his/her current level of utilization on already assigned tasks. Two utilization adjustment factors were found to be particularly useful when assigning a resource to a task. The first is the overloading factor, which allows a technology assembly center manager to load a resource a certain percentage beyond the usual limit, to take care of short-term spikes in work. The other is the realization factor, which estimates the percentage of time spent productively, and is often set to less than 100%, to account for delays that are expected to occur when the resource waits for some information related to the task (which can come from another resource, team, or technology assembly center); this delay represents available capacity that may be utilized for another service.

Given the large number of activities that can be assigned to factory resources, the optimization space is exponential. An Application Factory service may be assigned to one of several teams with comparable capabilities. Moreover, several resources within the same team may be able to perform one or more activities of this service. A number of heuristics is used to reduce the search space; for example, when selecting resources for an activity, preference may be given to those whose average utilization is below a given threshold during the period the activity will be performed; in a similar manner, a greedy approach may be used, where the activity end dates for the different resources are determined, and only the top few resources who finish earliest are considered; preference may be given to resources who have performed services for the same project or client before; if there are stringent quality criteria demanded by a service, past performance may be used to identify the proper resources. Such approaches help reduce the number of possible resource combinations across activities. They also implicitly perform a degree of load balancing, by giving preference to those resources that are currently less utilized.

Once a service is committed to the application development or support plans of the factory, its activities are embedded in the plans and the schedules of all assigned resources are reserved for the dates and durations of their activities[40]. These schedules are managed on an ongoing basis to account for any disruptions in the plans. Events such as the sudden unavailability of a resource trigger a re-factoring of the plans, where the allocated capacity of an assigned resource is freed up and plans are updated to adapt to the situation, leading to fresh assignments being made and new capacity being reserved elsewhere. Again,

the actual effort needed for a resource to complete an activity may exceed the estimated effort, thereby necessitating an extension to the activity due date; this results in additional resource capacity being reserved and triggers re-planning for all dependent activities, which in turn, impacts the calendars of the resources who were assigned those tasks. Conversely, if an activity is delivered before the due date, any remaining capacity allocated to it is freed up, so that the capacity may be re-used for other activities.

4.4 Managing Service Quality

The Application Factory has defined, implemented and deployed a number of select Work Packets for governing the business requirements reviews, system requirements reviews and application design reviews during application build (Figure 6), for executing integration test, and for managing and executing system test. Such reviews provide formal exit criteria to select activities of the Application Factory services[22, 23].

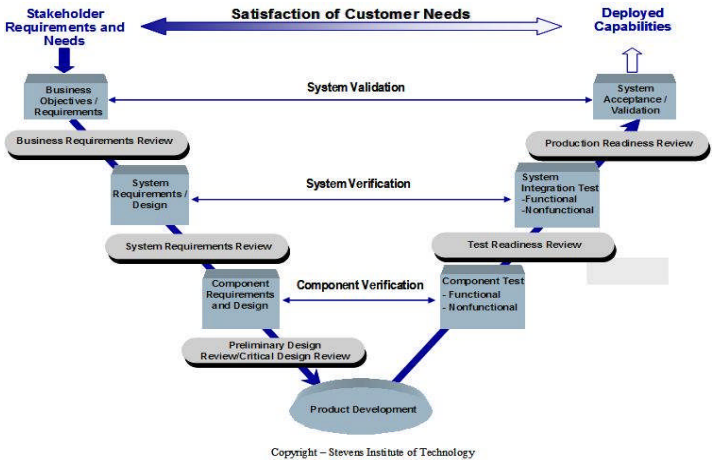

Figure 6. Illustration of the systems engineering discipline

- **Business Requirements Reviews**: The process is used to analyze the customer wants and needs. Its Work Packets include: plan for preliminary review of business requirements document; prepare for business requirements review; conduct business requirements review; work business requirements review open items; and close business requirements review.

- **System Requirements Reviews**: The process is focused on more detailed application and component level requirements. Its Work Packets include: create and update technical deliverables; conduct system requirements document and system interface agreements reviews; prepare for system requirements review; conduct system requirements review; work system requirements review open items; and close system requirements review.

- **Application Design Reviews**: The process reviews a specific application design, with emphasis on interfaces with other applications and systems. It Work Packets include: create and baseline system and feature requirements document; update, create and baseline system interface agreement; and incorporate design-time changes in system interface agreements.

- **Execute Integration Test**: The process covers the activities of planning, preparation, execution and verification of integration test. Its Work Packets include: plan integration test; prepare for integration test; conduct integration test; and verify integration test is complete.

- **Manage System Test**: The process manages overall system test activities. Its Work Packets include: approve project requirements; manage project – design center; manage project – technology assembly center; write system test plan, write system test cases, execute system test cases; support post-development testing.

- **Execute System Test**: The process covers the activities of planning, preparation, execution and verification of system test. Its Work Packets include: plan system test; prepare for system test; conduct system test; and verify system test is complete.

Experience shows that more effort is expanded on reviews during custom application development compared with package application development, and, for requirements and design reviews compared with test and production readiness reviews. Further, reviews account for ~15% of the total costs of development of custom applications and ~8.5% of the total costs of development of package applications.

4.5 Managing Operations

The measurement framework associated with factory operations is a dynamic system that is designed primarily with automation, extensibility, and flexibility in mind. First it addresses operational excellence and risk through monitoring for patterns of trouble, and when such patterns are detected, initiating associated resolution activities. Then, through feedback loops that provide ongoing learning and improvement, it provides a transformational impetus to the organization. Metrics of interest include average variances in effort, schedule and rework as well as savings per full-time equivalent (FTE). They are tracked at the activity level by comparing actual and plan durations, the level and hourly cost of resources performing planned as well as rework activities. To address the reality that the measurements that are needed vary by Stakeholder and Client, the framework supports multiple dimensions of measurements covering Client value, strategy, operations, risk, quality, and governance.

Metrics are defined in templates and then associated with a process template that collects them at various levels of granularity (e.g., project, phase, work-packet, activity, centers, teams, individuals). This approach separates metrics definition from process definition and encourages reuse of metrics across processes. When the process is enacted its associated metrics are gathered automatically and assessed against benchmarks or alert tables. Out of range values are mapped to a severity and an alert is raised, triggering immediate recovery actions. Collected data is periodically evaluated in order to continually upgrade the delivery processes of the factory.

The collection of metrics during a project's execution may rely on data that exists in external tools, such as project management, defect management and test management tools. Such tools may vary from project to project. In order to reuse metric definitions across different projects, the metric templates include a data-source and a target-system attributes. Data-source is used to bind a metric template at runtime to a specific tool, such as MS Project. Target-system allows the export of data, once persisted to a specific system, for reporting purposes.

The Application Factory data collection and reporting processes on standard Lean Sigma[48] measurements have been automated. Work Packet task execution drives measurement collection through time stamping, duration calculations, and integration with design, coding and test tools. Such automation significantly improves both the timeliness and accuracy of the collected data, generates automatic alerts with full traceability for root cause analysis and process improvement, and eliminates the Hawthorne effect, a common problem of many Lean Sigma initiatives in which individuals improve one or more aspects of their behavior in response to their awareness of being observed.

Select Application Factory measures are placed under statistical process control in order to define a capability index with upper/lower tolerances and are monitored for variation. The data can be continuously extracted from the Application Factory and the process capability index can be monitored on a regular basis with limited data collection effort.

5 Factory Architectural Framework

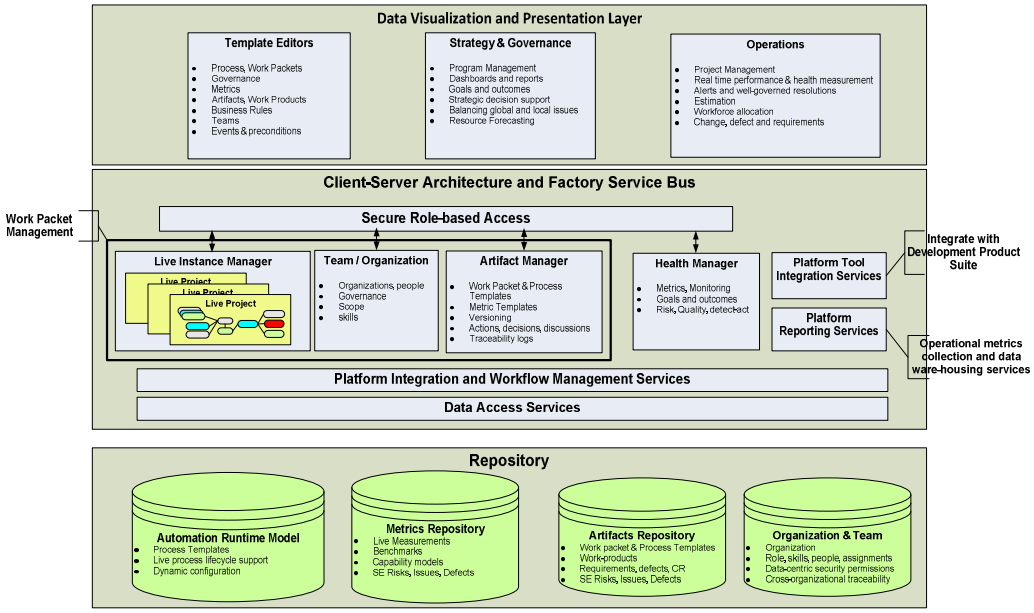

Figure 7. Application Factory Architectural Framework

The Application Factory architectural framework is a three-tier architecture (Figure 7) composed of the following layers:

- **Data Visualization and Presentation Layer:** This layer provides interactive interfaces and visibility into the factory operations for a variety of roles:
    - Subject matter experts (SME) who reside in the functional design center and customize the factory infrastructure and content to suit the needs of projects and their technology assembly centers.
    - Governance teams who measure, monitor and tune the factory for optimal performance across a portfolio of projects.

- Technology assembly center managers that provide process automation, coordination and governance functions for practitioners working on individual projects.

- **Client Server Architecture and Factory Service Bus:** This layer consists of process automation, artifact and pro-active health management systems that collectively provide factory management capabilities organized around the Work Packet concept. The layer also provides integration and process automation capabilities across a standard development product suite, reporting services that feed into business intelligence solutions and secure, scalable data access services with back-end support for several enterprise-class repositories.
- **Repository:** The factory repository consists of operational and historical data from real project instances covering the project lifecycle actions, metrics and artifacts. Some of the data may be located in the repositories of individual tools, such as, in the case of change management, defect management or project management. Our approach is to leverage the data where it resides by linking appropriate data elements to Application Factory process constructs, rather than aggregate the data into a centralized repository. Templates and other customizations that Subject Matter Experts perform to adopt Application Factory to specific client situations are stored centrally in the factory repository, in order to enable sharing best practices between engagements.

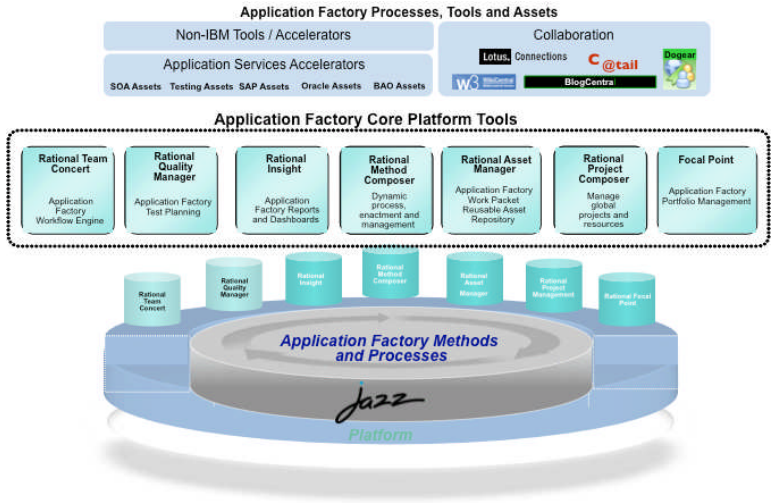

Figure 8. Application Factory Reference Implementation

5.1 Factory Reference Implementation

Figure 8 details a reference implementation of the Application Factory on the Rational Jazz Team Server. For the data visualization and presentation layer, the implementation supports multiple clients that are optimized for the governance, design center and the technology assembly center teams. Jazz server provides scalable end-user solutions with rich clients that require little or no installable code, and meet the requirements for a thin client. Other Rational products are used to capture business and IT requirements, architectural design, expertise and normative guidance, project/portfolio management, artifact management, and Work Packet authoring and automation.

Our reference implementation extends the Jazz service interfaces and client library as shown in Figure 9. These interfaces offer client-side instances that are supported by a Java-based client library that renders the client interface. A client library makes calls, over the network, to the associated service interface of the matching service on the server in support of processing client requests. The Application Factory extensions to this existing client-server architecture are built as eclipse plugins that provide enhanced functionality to manage process constructs, implement security requirements, enable metrics collection and provide event processing that enables governance activities.

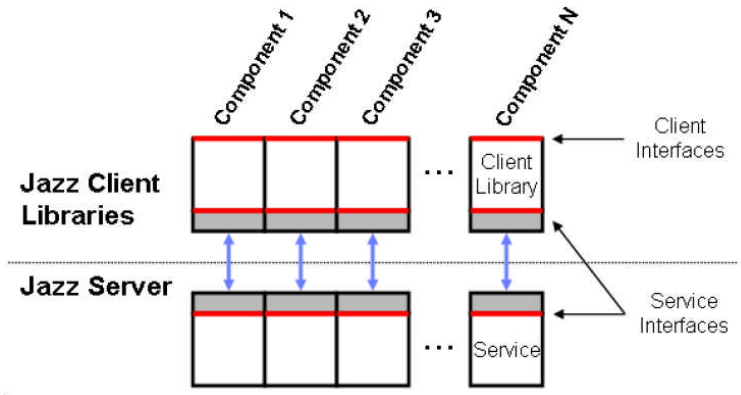

Figure 9. Jazz Client Server Architecture

The Jazz Platform repository is enhanced by the Application Factory data model extensions to support a variety of Application Factory process and metrics constructs. Figure 10, illustrates the Application Factory Work Packet with the additional attributes required to support metrics, processes and state transitions.

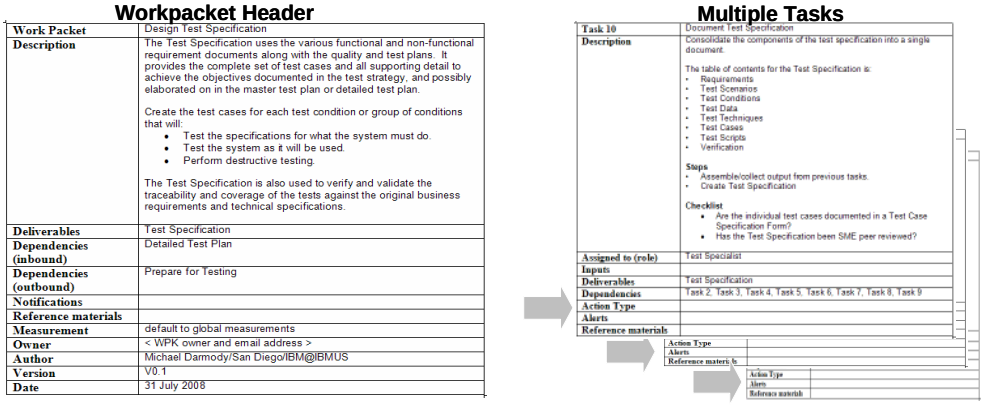

Figure 10. A Work Packet instance in the Application Factory Repository

The Jazz platform repository contains a comprehensive set of Application Factory solution artifacts as well as the data necessary to manage the linkage between the Application Factory repository and additional tool repositories required to support the total solution. It has three logical partitions:

1. An operational database that stores run-time (live) instances. Work items and additional Application Factory process constructs are represented in the repository as auditable items with built-in attributes represented as item properties. Custom attributes are stored as state extensions.

2. Index data that support rapid query resolution and retrieval, and full-text search.

3. A data warehouse and a snapshot framework for extracting historical data for reporting.

The repository provides standard data access services using server-side APIs for creating, retrieving, updating and deleting items. The logical design of the repository is independent of the semantics, storage models and Jazz APIs and provides an ideal extensible platform for Application Factory needs. Jazz provides a mechanism to correlate the data from external tools by way of a lightweight framework for proxy items. A proxy item has a one-to-one correspondence with a data object residing in another system.

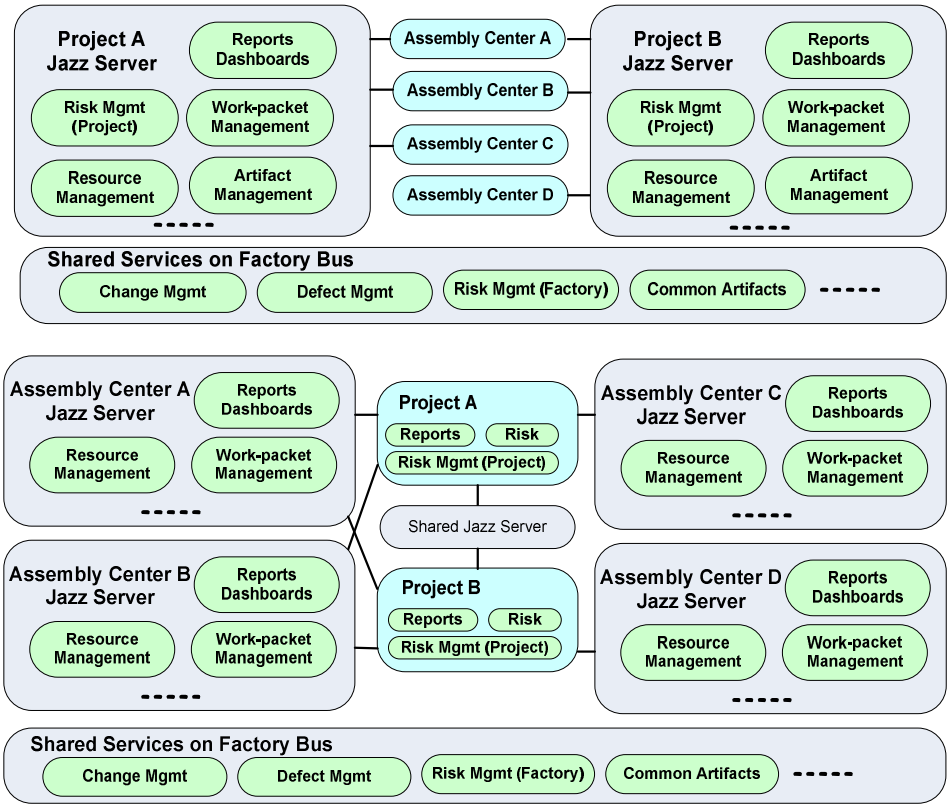

Figure 11. Different architectural topologies for Jazz-based Application Factory servers

5.2 Factory Topology Considerations

The Application Factory architecture can be deployed in multiple topologies, as stipulated in contractual agreements that specify the right level of access controls, scalability, sharing of process guidance and usability. Figure 11, shows two such topologies.

In the first case, individual projects are managed on separate Jazz-based factory servers, with a shared set of tools for managing change, risk, defects, requirements, artifacts and capacity. In this topology, a technology assembly center team receiving a Work Packet is remotely connected to the project's Jazz server and uses the client side functionality to receive work, access guidance documents, report status and perform other factory functions. There are significant downsides to using this approach – as resource planning becomes project centric, sharing of process improvements becomes difficult and responsiveness may suffer due to the performance latency involved in using a remote server.

In the second case, each technology assembly center uses a separate Jazz-based factory server and handles work from multiple projects. A separate central Jazz-based factory server acts as a host for one or more projects sending work to different technology assembly centers. The advantages of this model include easier resource planning and the sharing of process enhancements – practitioners performing similar work are hosted on the same factory server. This approach requires significant data flow and synchronization between the different factory servers involved in performing a project.

Other considerations considered in choosing a particular Application Factory topology include, among other factors, data security and access-control, resource planning, continuous process improvement, collaborative learning and information sharing, and responsiveness and scalability.

5.3 Dashboards and Information Visualization

In order for the Application Factory to orchestrate the activities of geographically distributed teams, its architecture should support the integration, correlation and visualization of data from multiple application development and management systems. Data visualization is best supported in a data warehouse model, with external tool integration handled as a separate function, using standard SOA extensions. This approach helps isolate workflow management from the classic data visualization support needed for reporting. Reporting capabilities include:

1. **Operational dashboards:** These dashboards provide the familiar controls needed by a project manager to support the day-to-day monitoring of managed workflows. They display timely alerts that are used to resolve exceptions and ensure development stays on track.

2. **Tactical dashboards:** Large services engagements are made up of a portfolio of projects, spanning multiple technology assembly centers. Tactical dashboards help manage, with the proper granularity, the workflows associated with project plans. They also provide the capability to roll-up these plans into an aggregate view in order to ensure visibility to the status of the entire portfolio.

3. **Strategic dashboards (or Scorecards):** Application Factory provides also the capability to trace requirements to deliverables and ensure that development can be validated and correlated against corporate strategic objectives at each level of the organization.

The Application Factory uses the Eclipse BIRT (Business Intelligence and Reporting Tools) for reasonably large projects and the Cognos enterprise reporting and data visualization tools for very large projects. These large projects require additional capabilities such as cross-tool data integration and data warehouse reporting and analysis functions. Figure 12 shows an example Application Factory dashboard.

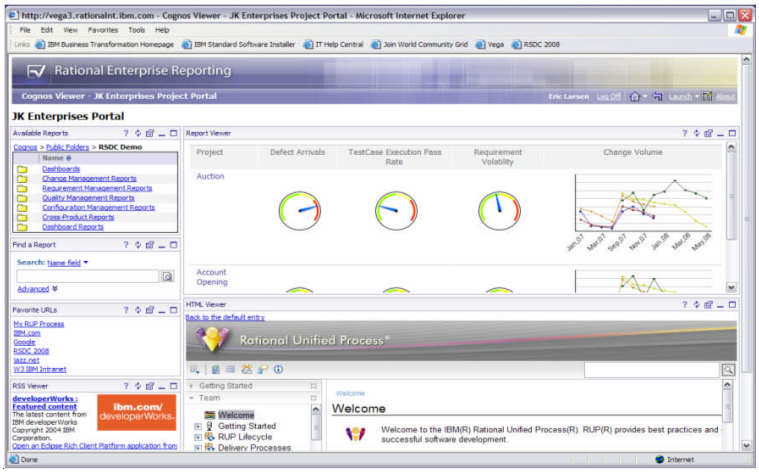

Figure 12. Application Factory Tactical Dashboard for Governance Team

5.4 Process Integration

The Application Factory Work Packet model extends the OMG[18] Software Process Engineering Meta-model (SPEM) with constructs aimed at capturing the project structure, the organization structure, the artifacts, the metrics, and the governance elements of its global delivery processes. Its process automation environment uses UML[67] as the language for modeling Work Packets as parametrized templates that are used in project instance execution.

The Work Packet authoring environment is built as an extension to Rational Method Composer (RMC). RMC is a flexible SPEM-compliant process specification environment. Since RMC is built based on the Eclipse platform, the Work Packet model is implemented as extension-points. Furthermore, the Project Templates are implemented as extensions to the method plug-in in RMC in order to support global delivery-specific annotations and the enactment and monitoring of the process instances. An extensive library of Work Packets, capturing the activities associated with performing the factory services, has been created. The durations of such activities have been captured and are continualy updated to reflect the degree of complexity and the overall skills of the implementors of a given service.

Work Packet instances coordinate the operation of global teams in the technology assembly centers by pushing the activities performed by such teams to their workspaces. This is accomplished by creating, in Rational Team Concert (RTC), Work Packet instances that are based on the RMC specification templates. RTC connects the activities of Work Packets to the appropriate development tools used in managing the artifacts of a project.

Several Application Factory operational processes, governing aspects of the factory that have high variability from project-to-project such as resource management, project management and defect/change management, need to be designed for seamless integration with traditional tools and practices. Integration of these processes is key to maintaining traceability from a data model perspective to core factory constructs like Work Packets, tasks, metrics and deliverables. As illustrated in Figure 13, standard query and link management capabilities provided natively by the tools are leveraged for the purpose of providing traceability. For example, the lifecycle of a change request that is logged against a project is managed in a change management tool such as ClearQuest, while a link (URI) to the change request is maintained in the factory data model. There are several advantages to this approach:

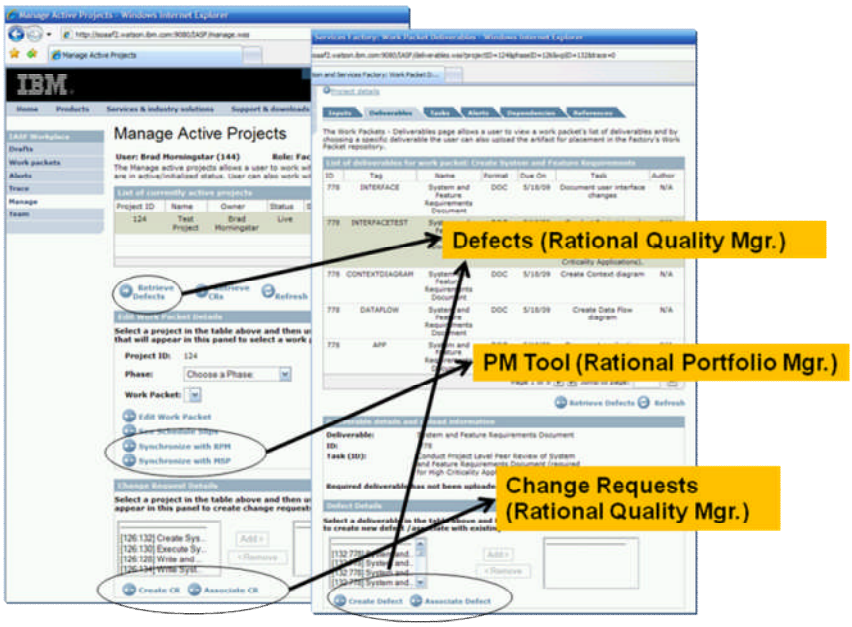

Figure 13. Web Interface of Loosely Coupled Integration with Change, Defect and Project Management Tools.

1. **Improved data consistency:** By allowing the persistence model for these processes to reside in the native tools, data replication and the need to synchronize data across repositories are avoided. The native data repositories of the tools used to execute the change/data management processes become extensions of the factory repository. Avoiding unnecessary data transformation and replication leads to improved reliability for the underlying data.
2. **Leveraging existing process definitions and tool customization:** Services focused on specialized areas such as packaged applications have highly specialized processes to manage specific activities such as capacity planning and change management with a defined set of roles, life-cycle states and data persistence requirements. Our approach provides for a quick adoption of these processes while preserving full visibility into the process execution.

3. **Ability to define a common minimum standard for process adoption:** Factory defines minimum requirements for querying, data visualization, link management and data (metrics) capture/aggregation for the individual factory processes to be eligible for adoption into the factory. Thus, using discrete tools to manage core factory processes does not result in a loss of the ability to monitor and carefully manage these processes. A recommended set of tools such as Rational Quality Manager (defects, changes), Rational Requisite Pro (requirements), Rational Asset Manager (artifacts) that satisfy factory process management requirements provide an out-of-the-box solution.

4. **Extensibility to support new process components:** Depending on the industry segment and complexity of a project, process controls in addition to the ones that are standard across all projects may be required. Examples include processes to enforce compliance with government regulations and data persistence in financial institutions. The loosely coupled process management approach espoused by the factory is eminently suitable to cover such situations.

6 Related Technologies

Business reasons that drive distribution of software development and support across multiple sites and organizations have been well documented in recent years[52]. The challenges associated with distributed development of software have also been extensively studied and recorded[49, 59, 62]. Many of these challenges may be traced back to inadequate (informal and formal) communication between team members separated by distance and time-zone differences, which lead to irregular information flows and frequent misalignment. To address this, two broad directions have emerged. On the one hand, there has been a push to make software development tools and environments more collaborative[5, 21, 60], with emphasis on informal synchronous and asynchronous contextual communication and mutual awareness of activities, as embodied in tools like Jazz[42], Egret[63] and Sangam[61].

On the other hand, there have been several efforts towards formal process modeling and enactment in a distributed setting, to automate the flow of work between global teams and manage and coordinate the same[46, 50, 54]. This is similar to the factory approach of allowing process templates to be authored, and then instantiating and executing them using the automation middleware.

Software process models describe various concepts (e.g., activities, people and resources) that are involved in the software process. Many process modeling approaches have been described in the literature, with advantages that are well discussed: automation, controllability and reusability, among others. Barnes et al. define the software process through the meta-schema of the database in order to use it as a process management system[11]. Zhao et al. propose an agent-based approach to modeling the software process and constructing a process-centric software engineering environment[75]. UML-based software process modeling[53] can be seen as one of the most promising approaches, but UML is by design a general modeling language that lacks specific details for the software process. The Software Process Engineering Meta-Model SPEM[64] is the standard from the Object Management Group (OMG) for describing software engineering processes that serves as a reference for the process model in the Application Factory. It has extensive support for defining UML profiles that describe method content in the form of activities, roles, work products, and guidance that can then be assembled to specify a delivery process. However, since SPEM is not specifically designed for the global delivery context, it lacks constructs for describing the work encapsulated for a remote team. In addition, without enhancing the metric definitions to link with business goals, the work descriptions in SPEM cannot be seen as a contract with the remote team for Application Factory usage. To deal with exception conditions, Application Factory processes also require

the extension of specifying governed resolution processes for handling alerts that are raised or metric values that fall outside of a benchmark range.

Due to the similarity with traditional workflow systems, some literature extends the workflow model to support the description of software processes[41]. The Web Services Business Process Execution Language WS-BPEL[70] is a well-known process language that describes the control flow and data flow between atomic web services to achieve service composition in the area of SOA[47]. It also supports the description of abstract process structure without coupling to specific implementation methods. Other well-used languages, such as XML Process Definition Language XPDL[74], also support the automation of software process in workflow systems. However, all these languages are designed primarily for automating processes that are executed by machines. Despite extensions that include more consideration of human features, such as BPEL4People[72], they remain less applicable for the large-scale human intensive workflow scenarios at which the Application Factory is targeted. Some important elements of concern to the Application Factory, such as work products and guidance, are missing in such languages.

In addition, expertise and knowledge sharing are increasingly used in distributed projects to help team members execute their assigned tasks efficiently. There are tools like the Expertise Browser[57] that can learn the expertise of different individuals from change management systems, versioning systems etc. and then help new team members locate required expertise; there are also tools like Hipikat[44] that recommends relevant software development artifacts based on context-specific user requests.

Closer to our approach, Milos[55] structures knowledge according to processes and realizes a knowledge management approach that enriches process models by attaching skill models to tasks (e.g. technical skills needed to perform the task), and also attaching links to background knowledge (e.g. online tutorials, specifications from standard bodies); the skill model may be taken into consideration by a manager when assigning a task to an individual, while during execution, the workspace of users will provide access to the input documents needed for the task.

What differentiates Application Factory from these endeavors, however, are the integration of process modeling, enactment and associated knowledge management in a truly industrial scale based on robust tooling that can be successfully deployed in large distributed projects, and more importantly, going beyond these to provide a sophisticated metrics and governance framework that supports proactive monitoring of project health and informed decision making.

Reducing process variance through adoption of techniques to improve production efficiency from the manufacturing arena for improving IT Service delivery is becoming more popular, as evidenced by experiments in Wipro detailed in a Harvard Business School case study[68]. However, translating LEAN techniques to IT faces significant challenges – careful inventory management and the improvement of production line efficiency[58], which form cornerstones of LEAN in a manufacturing setting, do not have a direct analogue in service delivery.

In an interesting second stage update on this case study[66], it was found that LEAN adoption for services resulted in the adoption of an agile methodology for software development (as opposed to a waterfall approach) resulting in significant improvements in quality and productivity. Improvements were attributed to a willingness to quickly share and learn from mistakes, standardization and promotion of

creative problem solving. Our approach to the Application Factory takes thinking a step forward by creating a structured approach to understand and monitor service delivery in standardized Work Packet containers, while promoting innovation in the delivery of individual Work Packets.

Another distinguishing feature of the Application Factory is its sophisticated resource management framework that optimally assigns resources across multiple sites to planned activities and monitors progress in performing such activities in order to minimize the impact of any disruptions. The framework also helps mitigate the risks associated with resource turnovers and fluctuations in skills of delivery teams. Xiao et al propose a similar, but more limited, constraint-driven approach to the optimal scheduling of resources of mature development teams[73]. On another front, Wickramaarachchi and Lai offer a method for distributing work to multiple locations[71]. Work is first mapped into phases of the Software Development Life Cycle (SDLC) that are grouped based on the Software Process Model (SPM). The final work distribution uses criteria such as work and site dependencies and characteristics in order to meet project objectives.

Last but not least, meeting Client expectations in today's competitive marketplace requires continuous delivery of complex applications with predictable quality. A pragmatic approach to achieving consistent quality was outlined by Sprenger in an invited industrial talk. This approach is based on a well-balanced system of processes and continuous measurements and control that bear great similarities to test series and assembly lines in the automobile industry[65].

The Menlo software factory is a recent initiative that employs techniques from across the technical and managerial spectrum: the Project Management Institute, Alan Cooper's user-centered design, Kent Beck's extreme programming and test-driven development, W. Edwards Deming's quality teaching and more[56].

Japanese firms created "software factories" in which large numbers of people engage in developing software in cooperative ways and reuse code segments whenever possible[45]. Their strategies mark an early attempt to applying engineering and production management skills to software development.

The software factory model is now seen as a critical factor in the successful development of efficient and cost-effective software products and medium to large-scale software projects[69]. The Application Factory is based on similar principles that have been enhanced to create a virtual factory of geographically distributed assembly lines.

7 Conclusion

The paper describes in detail the process automation, Work Packet structure, core processes, architectural framework and measurements that help deliver on the Application Factory key value propositions of governance, lifecycle acceleration and labor acceleration. The paper also provides insights into the customization, extensions and trade-offs involved in deploying the Application Factory platform to global teams servicing a multitude of Clients, in order to promote sharing of process improvements across such teams while maintaining Client confidentiality.

Application Factory breaks new ground on several concepts for effectively managing distributed teams. The Work Packet forms a central process construct that enables a breakdown of a complex project into process constructs that can be repeatedly used across several engagements. Work Packets play several critical roles, from promoting knowledge sharing and providing deep instrumentation and measurement of delivery processes to being effective mechanisms for decentralized project and process management.

Expertise capture by Work Packets enables the continual evolution and sharing of targeted and up-to-date knowledge and solutions among team members.

A central tenet behind the Application Factory platform is to make the architecture flexible enough to co-opt rather than replace existing tools, processes and repositories. To this end, several core factory processes that deal with capacity, project, risk and artifact management can be implemented with a variety of tools. The platform provides a central portal to query the data in and connect to these tools, but allows considerable latitude in defining the combination of tools and processes used for deployment.

Today, the Application Factory services are widely used in managing application development and support for IBM Clients. Once a Client request for proposals (RFP) is issued, the Application Factory services catalog is consulted and relevant services are embedded as solution building blocks. The risks associated with delivering such services within the timeframe and delivery constraints identified by the Client are then assessed. The availability and skills of eligible design, development and support teams are checked in order to identify the teams with the best track record delivering each of the services. The likelihood of engaging a specific team is computed next in order to avoid any delays in solution delivery. The above assessment may also point to some skill gaps that need to be filled. It is worth noting that most Application Factory development and support services are offered as fixed-price services because the effort required to perform their Work Packets can be estimated with little variability.

As the odds of the Client selecting the proposed solution increase, the Application Factory delivery teams are notified and their work schedules are updated to reflect their assignments. Delivery starts with a workshop aimed at defining the structure and policies that will govern the business and technical interactions between the Client and Application Factory teams. This is followed by a series of face-to-face work sessions between the Client and Application Factory business analysts and IT architects that are aimed at gaining a deeper understanding of Client requirements and iteratively developing high-level designs that capture such requirements. A design iteration is planned to last between one and four weeks, with shorter cycles reserved for the more novel and significantly more complex elements in a design. The technology assembly center team leads responsible for configuring and integrating the Application Factory services included in the designs, will attend these work sessions in person or via video conferencing.

Rational requirements analysis and design tools are extensively used to document the original set of business and technical requirements, together with their associated refinements, high-level designs and requirements-based test plans. A separate service quality team reviews these documents before they are turned over to the technology assembly center teams. Moreover, the original solution delivery plan, created as part of the RFP response, is iteratively refined to capture the detailed Work Packets involved in delivering each design iteration. The execution of the Work Packet activities is then closely monitored in order to reduce and /or eliminate any delays associated with unforeseen circumstances such as medical emergencies and unplanned absences.

The virtual Factory model has progressively generated significant savings in the cost of delivery of complex applications. Savings in the range of 20% to 35% per service have been recorded, and the user population of the platform has exceeded 25,000 practitioners. Its underlying framework offers adequate capabilities for implementing virtual factory models that leverage a globally distributed workforce to also deliver infrastructure and business processes services.

References

1. S. Agarwal, S. Bagheri, J.K. Chaar, K.C. Ratakonda, V. K. Reddy and B. Sengupta; Optimizing a shared service delivery system; US PATENT 8,386,290

2. S. Agarwal, S. Bagheri, J.K. Chaar, A. Parandehgheibi, K.C. Ratakonda and B. Sengupta; Tool for manager assistance; US PATENT 8,417,554

3. S. Agarwal, S. Bagheri, J.K. Chaar, K.C. Ratakonda, and B. Sengupta; Optimized Collaboration Between Distributed Centers Of Global Service Delivery Systems; US PATENT Application Publication 2012/0284073

4. S. Agarwal, S. Bagheri, J.K. Chaar, K.C. Ratakonda, and B. Sengupta; Optimizing Service Delivery Systems; US PATENT Application Publication 2012/0284076

5. B. Al-Ani and D. Redmiles; "Supporting Trust in Distributed Teams through Continuous Coordination". IEEE Software, Vol. 26, Issue 6, pp. 35-40.

6. S. Bagheri, J.K. Chaar, Y-M Chee, D.V. Oppenheim, K.C. Ratakonda and N. Zhou; Scheduling resources from a multi-skill multi-level human resource pool; US PATENT 8,407,073

7. S. Bagheri, J.K. Chaar, Y-M Chee, K.C. Ratakonda and N. Zhou; Collaboration based capacity planning in a modular business setting; US PATENT 8,417,555

8. S. Bagheri, J.K. Chaar, Y-M Chee, K.C. Ratakonda and N. Zhou; Automated allocation of resources to functional areas of an enterprise activity environment; US PATENT 8,463,637

9. S. Bagheri, J.K. Chaar, Y-M Chee and K.C. Ratakonda; Work Plan Prioritization for Application Development And Maintenance Using Pooled Resources in A Factory; US PATENT 8,549,527

10. S. Bagheri, J.K. Chaar, Y-M Chee, F. Liu, D.V. Oppenheim and K.C. Ratakonda; Management of Template Versions**;** US PATENT Application Publication 2012/0291007

11. A. Barnes and J. Gray; "COTS, Workflow and Software Process Management: An Exploration of Software Engineering Tool Development", Proceedings of the 12th Australian Software Engineering Conference, Canberra, Australia; IEEE Computer Society, Washington D.C. (2000), pp. 221-232.

12. F. Bernardini, J.K. Chaar, Y-M Chee, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Open marketplace for distributed service arbitrage with integrated risk management, US PATENT 8,140,367

13. F. Bernardini, J.K. Chaar, Y-M Chee, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Self-healing factory processes in a software factory; US PATENT 8,271,94

14. F. Bernardini, J.K. Chaar, Y-M Chee, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Management of Work Packets in a software factory; US PATENT 8,370,188

15. F. Bernardini, J.K. Chaar, Y-M Chee, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Determining Competence Levels of Factory Teams Working Within A Software Factory; US PATENT 8,595,044

16. F. Bernardini, J.K. Chaar, Y-M Chee, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Staged Automated Validation of Work Packets Inputs and Deliverables in a Software Factory; US PATENT Application Publication US 2009/0300586 A1

17. F. Bernardini, J.K. Chaar, Y-M Chee, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Analyzing Factory Processes in a Software Factory; US PATENT Application Publication 2012/0245896

18. F. Bernardini, J.K. Chaar, Y-M Chee and K.C. Ratakonda; Model-Driven Assignment of Work to A Software Factory; US PATENT Application Publication 2012/0323624

19. I. Bhandari, M. Halliday, E. Tarver, D. Brown, J. Chaar and R. Chillarege, A Case Study of Software Process Improvement during Development, *IEEE Transactions on Software Engineering*, vol. 19, no. 12, December 1993, pp. 1157-1170.

20. I. Bhandari, M.J. Halliday, J. Chaar, R. Chillarege, K Jones, JS Atkinson, C Lepori-Costello, PY Jasper, ED Tarver, CC Lewis and M Yonezawa; "In-Process Improvement through Defect Data Interpretation", *IBM Systems Journal, vol. 33, no. 1*, 1994.

21. G. Booch and A. Brown; "Collaborative Development Environments". Advances in Computers Vol. 59, Academic Press, August 2003.

22. J. K. Chaar, M. Halliday, I. Bhandari and R. Chillarege; "On The Evaluation of Software Inspections and Tests", International Test Conference (ITC 93), 1993.

23. J. K. Chaar, M. Halliday, I. Bhandari and R. Chillarege; "In-Process Evaluation for Software Inspection and Test", IEEE Transactions on Software Engineering, vol. 19, no. 11, November 1993, pp. 1055-1070.

24. J. K. Chaar, R. D. Finlayson, T. A. Jobson Jr,, N. M. Mitsumori and F. X. Reddington; Rapid on-boarding of a software factory, US PATENT 7,958,494

25. J. K. Chaar, R. D. Finlayson, T. A. Jobson Jr,, N. M. Mitsumori and F. X. Reddington; Assembling Work Packets within a software factory; US PATENT 8,141,040

26. J. K. Chaar, R. D. Finlayson, T. A. Jobson Jr,, N. M. Mitsumori and F. X. Reddington; Software factory readiness review; US PATENT 8,296,719

27. J. K. Chaar, R. D. Finlayson, T. A. Jobson Jr,, N. M. Mitsumori and F. X. Reddington; Software factory health monitoring; US PATENT 8,327,318

28. J. K. Chaar, J. P. Huchel and T. A. Jobson Jr,; Supporting a Work Packet request with a specifically tailored IDE; US PATENT 8,336,026

29. J. K. Chaar, R. D. Finlayson, J.P. Giraldo, S.R. Lang, N. M. Mitsumori, H. Rajagopal, F. X. Reddington and T. A. Vines; Software factory; US PATENT 8,359,566

30. J. K. Chaar, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Application/service event root cause traceability causal and impact analyzer; US PATENT 8,375,370

31. J. K. Chaar and J. P. Huchel; Software factory semantic reconciliation of data models for Work Packets; US PATENT 8,418,126

32. J.K. Chaar, A. Hamid, R. Harishankar, J.P. Huchel, T.A. Jobson Jr, D.V. Oppenheim and K.C. Ratakonda; Work Packet delegation in a software factory; US PATENT 8,448,129

33. J. K. Chaar, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Work Packet enabled active project schedule maintenance; US PATENT 8,452,629

34. J.K. Chaar, W.R. Cope Jr, P.J. Kroll, and G.R. Williams; Automated time tracking; US PATENT 8,463,670

35. J. K. Chaar, R. D. Finlayson, T. A. Jobson Jr,, N. M. Mitsumori and F. X. Reddington; Life Cycle of a Work Packet in a Software Factory; US PATENT 8,464,205

36. J. K. Chaar, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Configuring Design Centers, Assembly Lines And Job Shops of A Global Delivery Network Into "On Demand" Factories; US PATENT 8,527,329

37. J. K. Chaar, R. D. Finlayson, T. A. Jobson Jr,, N. M. Mitsumori and F. X. Reddington; Work Packet Forecasting in a Software Factory; US PATENT 8,566,777

38. J. K. Chaar, R. D. Finlayson, T. A. Jobson Jr,, N. M. Mitsumori and F. X. Reddington; Project Induction in a Software Factory; US PATENT Application Publication US 2008/0256390 A1

39. J. K. Chaar, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Intelligent Job Artifact Set Analyzer, Optimizer and Re-Constructor; US PATENT Application Publication US 2010/0023920 A1

40. J.K. Chaar, J. P. Huchel, T. A. Jobson Jr, D. V. Oppenheim and K. C. Ratakonda; Work Packet enabled active project management schedule; US PATENT Application Publication US 2013/0185693 A1

41. K. C. Chan and R. P. H. Leung; "A Workflow Vista of the Software Process", Proceedings of the 8th International Workshop on Database and Expert Systems Applications, IEEE Computer Society, Washington D.C. (1997), pp. 62-67.

42. L-T Cheng, C. DeSouza, S. Hupfer, J. Patterson and S. Ross; "Building Collaboration into IDEs". ACM Queue 1(9), 2004

43. R. Chillarege, I. Bhamdari, J. Chaar, M. Halliday, D. Moebus, B. Ray and M.-Y. Wong; "Orthogonal Defect Classification - A Concept for In-Process Measurements", *IEEE Transactions on Software Engineering, vol. 18, no. 11,* November 1992, pp. 943-956.

44. D. Cubranic and G. Murphy; “Hipikat: Recommending Pertinent Software Development Artifacts”. International Conference on Software Engineering (2003), pp 403-418.

45. M. A. Cusumano; Japan’s Software Factories: A Challenge to U.S. Management, Oxford University Press, U.S.A., March 1991.

46. Z. Y. Duanmu, J. J. Yan and Y.F. Lee; “An Approach of Process Control in Software Product Line”. IEEE/ACM International Conference on Green Computing and Communications (2011), pp. 139-143).

47. D. F. Ferguson and M. L. Stockton; “Service oriented architecture: Programming model and product architecture,” IBM Systems Journal, 44, No. 4, 753-780. 2005

48. M. George; “Lean Six Sigma for Service.” McGraw Hill, 2003.

49. J. Greenfield, K. Short, S. Cook, S. Kent and J. Crupi; “Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools,” Wiley, 2004.

50. J. C. Grundy and J. G. Hosking; “Serendipity: integrated environment support for process modeling, enactment and work coordination”, Automated Software Engineering: Special Issue on Process Technology 5(1), January 1998

51. M. Halliday, I. Bhandari, J. Chaar, E. Tarver, D. Brown and R. Chillarege; “Experiences in Transferring a Software Process Improvement Methodology to Production Laboratories”, Achieving Quality in Software (AQuIS 93), 1993.

52. J. D. Herbsleb and D. Moitra; “Global Software Development”, IEEE Software, March-April, 2001

53. D. Jäger, A. Schleicher and B. Westfechtel; “Using UML for Software Process Modeling”, ACM SIGSOFT Software Engineering Notes 24, No. 6, 91-108 (1999).

54. F. Maurer, G. Succi, H. Holz, B. Cotting, S. Goldmann and B. Dellen; “Software Process Support over the Internet”. Proceedings of International Conference on Software Engineering (1999).

55. F. Maurer and H. Holz; “Integrating Process Support and Knowledge Management for Virtual Software Development Teams”. Annals of Sofware Engineering, vol. 14, 2002

56. The Menlo Innovation Team; Innovation Exploration: A Tour of the Menlo Software Factory, 2010

57. A. Mockus and J. Herbsleb; “Expertise Browser. A Quantitative Approach to Identifying Expertise”. International Conference on Software Engineering (2002), pp 503-512

58. Y. Park, H. Choi and J. Baik; “A Framework for the Use of Six Sigma Tools in PSP/TSP”. 5th Applications. (2007), pp 807-814.

59. A. Porter, C. Yilmaz, A. Memon and D. Schmidt; “Skoll: A Process and Infrastructure for Distributed Continuous Quality Assurance”, IEEE Transactions on Software Engineering, Vol. 33, Issue 8, pp. 510-525.

60. J. P. Rodriguez, C. Ebert and A. Vizcaino; “Technologies and Tools for Distributed Teams”. IEEE Software, Vol. 27, Issue 5, 2010, pp. 10-14.

61. http://sangam.sourceforge.net

62. B. Sengupta, S. Chandra, V. Sinha; “A Research Agenda for Distributed Software Development”. International Conference on Software Engineering (2006)

63. V. Sinha, B. Sengupta, S. Chandra. “Enabling Collaboration in Distributed Requirements Management”. IEEE Software, Sep-Oct 2006

64. *Software Process Engineering Meta-Model (SPEM)*, Object Management Group, Inc., http://www.omg.org/ technology/documents/formal/spem.htm.

65. T. Sprenger; “How Software Engineering Can Benefit from Traditional Industries – A Practical Experience Report (Invited Industrial Talk)”. International Conference on Software Engineering, 2012.

66. B. R. Staats and D. M. Upton; “Lean principles, learning and software production: Evidence from Indian software services”. Harvard Business School pre-print 8-001.

67. *Unified Modeling Language (UML)*, Object Management Group, Inc., http://www.uml.org.

68. D. M. Upton and V. A. Fuller; “Wipro Technologies: The Factory Model”. Harvard Business School Case No. 9-606-021, October 2005.

69. M. Van Genachten; Towards a Software Factory. Springer (1992, 2012).

70. *Web Services Business Process Execution Language (WS-BPEL)*, Organization for the Advancement of Structured Information Standards, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

71. D. Wickramaarachchi and R. Lai; A Method for Work Distribution in Global Software Development. 3rd IEEE International Advanced Computing Conference (IACC) (2013), pp. 1443-1448.

72. *WS-BPEL Extension for People (BPEL4People)*, Active Endpoints Inc., Adobe Systems Inc., BEA Systems Inc., IBM Corporation, Oracle Inc., and SAP AG, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/ library/uuid/30c6f5b5-ef02-2a10-c8b5-cc1147f4d58c

73. J. Xiao, Q. Wang, M. Li, Y. Yang, F. Zhang and L. Xie; “A Constraint-Driven Human Resource Scheduling Method in Software Development and Maintenance Processes”. International Conference on Software Maintenance and Evolution (ICSME) (2008), pp. 17-26.

74. *XML Process Definition Language (XPDL)*, Workflow Management Coalition, http://www.wfmc.org/xpdl.html.

75. X. Zhao, K. Chan and M. Li, “Applying agent technology to software process modeling and process-centered software engineering environment”, Proceedings of the 20th ACM Symposium on Applied Computing, Santa Fe, New Mexico; ACM, New York (2005), pp. 1529-1533.