Enterprise Architecture for education

Tim Bauer

TABLE OF CONTENTS

Executive Summary 1
General Background

Organizational Background

Enterprise Architecture Vision

Identification of Architectural Issues 3
Implementing EA 4

Lack of Enterprise Architecture

Technical Debt

Lack of Quality Processes

Software Complexity

Recommended Solutions 9

Conclusion

Roadmap To EA

Appendices 13

Appendix A: Technical Debt - Process

Source
Trigger
Condition

Appendix B: Technical Debt - Engineering

Source
Trigger
Condition

Appendix C: Technical Debt - Management

Source
Trigger
Condition

Appendix D: People CMM Matrix

Appendix E: Role- System Matrix

Appendix F: Benefits Diagram

Appendix G: - Current Architecture

Appendix H: - Future Architecture

Appendix I: Use Case Diagram
Executive Summary

The Organization is a School District providing primary and secondary education to over 65,000 students and 6,600 employees. Current economic events and business drivers for transformation require significant reliance on technology. The strategic objectives of School Choice, World-Class Education, and System Performance required an integrated environment to enable data-driven decision making. An Enterprise Architecture (EA) program will ensure the alignment of business and IT strategies, and result in the integration and standardization of core business processes. However, there are significant structural issues of technical debt, lack of processes, and the complexity of current legacy systems inhibit the organization to effectively change.

This document proposes an Enterprise Architecture (EA) program that supports the transformative strategic goals by providing organizational agility, business process efficiencies, and lowers overall total cost of ownership. Users are provided feature and content-rich integrated applications enabling a proactive transformation of processes, increased functionality, and a more responsive EA. The utilization of Web 2.0 technologies decrease the complexity of implementing the EA, reducing training and sustainment costs.

The EA program addresses the paradox of technical debt, system complexity, and software quality assurance. The organization needs to instantiate substantiative changes is software development processes to reduce complexity and to ensure overall quality, maintainability, and sustainability. The instantiation of modern design patterns and technology will provide the means to integrate legacy applications, reduce complexity in the enterprise environment, increase system flexibility, and reduce overall IT costs. These efforts will maximize the return on investment, ensure appropriate security mechanisms, and enable organizational transformation.

General Background

Organizational Background

The Organization is a School District providing primary and secondary education to over 65,000 students and 6,600 employees. The annual student population grows at an average 1,200 students annually, with approximately 1 school per year being constructed. To counteract current economic events, the District has established policies in reductions in administration and headquarters staff, to include ITS, and has an expectation that technology will provide innovation pathways for education delivery and administration.

The Organization is overseen by a seven member school board, elected on a four year cycle, with each member with jurisdictional of a high school and the corresponding feeder schools. The executive members are comprised of the Superintendent and cabinet members that represent the 11 business units. The business units include: Community Relations, Secondary Education, Ele-

The District strategic goals are School Choice, World-Class Education, and System Performance. School Choice is intended to provide parents the ability to evaluate school competencies and place their children in any school within the District. The World-Class Education (WCE) strategic goals are to provide personalized and sustainable learning environment that fosters creativity, problem solving, and innovation. The tools are to empower teachers and students to take ownership of their own pathway to success. The System Performance strategic goal is to develop a System Performance Framework (SPF) that is integrated with education delivery mechanisms. The SPF is intended to provide both the District and community accessible performance information for ultimate accountability and continuous system improvements. Teacher effectiveness and student performance metrics are determined by an internally developed District standards and curriculum.

**Enterprise Architecture Vision**

The Enterprise Architecture (EA) supports the transformative strategic goals by providing organizational agility and efficiencies in both business processes and IT systems lifecycle support (see Appendix H). The EA empowers the user by providing data access through feature-rich integrated applications. The EA utilizes a Web Oriented Architecture (WOA) that leverages Service Oriented Architecture (SOA), Business Process Management (BPM), and Web 2.0 technologies to provide a robust architecture that supports the integration of data and functional capability from disparate systems (Hoskins 2008). The effective use of SOA, BPM, and Web 2.0 increases functionality and IT value to the Organization whilst decreasing implementation and sustainment costs (Minoli 2008, 2009).

The SOA design pattern modularizes component capabilities into loosely-coupled ubiquitous services. These services communicate using interfaces based on standards enabling continuous reuse of services and negates the reliance of internal component implementation (Wagner 2007). The decoupling of components makes it possible for systems to be independent yet leverage each other irrespective of “technology, platform, location, or environment choices” (Minoli 2008, 202). The utilization of standards and platform independence provide the ability for legacy systems to be integrated with modern technologies increasing their value and longevity. Additionally, the independence of components provides agility through focused modernization without negatively impacting capability (Minoli 2008, 202).

The leveraging of services through process orchestration provide flexibility and the realization of business process transformation (Palmer 2005). SOA services are leveraged through Business Process Management (BPM) which models, automates, and manages business workflows. BPM allows continuous improvement of processes “resulting in significant time and cost savings before they are implemented.” (Carter 2007). The utilization of BPM with SOA permits the Organization to effectively transform processes proactively, maximizing the effectiveness, and reduce overall costs (Matei 2011).

The EA utilizes Web Oriented Architecture (WOA) technologies to simplify the implementation of SOA and expand usability to mobile and web platforms. WOA is the limiting view of SOA and focusing on Web 2.0 technologies such as Representational State Transfer (REST), Really
Simple Syndication (RSS), and Asynchronous JavaScript and XML (AJAX) to provide a simplistic means to aggregate lightweight and rich content across mobile and web platforms (Schroth 2007). The EA integration is internally driven, meaning that external organizations will not consume EA services, providing the opportunity to utilize static syndication of content and services decreasing complexity in implementation (Thies 2008). The EA implemented technologies provide opportunities to aggregate internal services and external Software as a Service (SaaS) offerings into a composite of functionality, called mashups, increasing the rate of rapid application development and available features (Thies 2008).

The proposed Enterprise Architecture (EA) supports the transformative strategic goals by providing organizational agility, business process efficiencies, and lowers overall total cost of ownership. Users are provided feature and content-rich integrated applications enabling a proactive transformation of processes, increased functionality, and a more responsive EA. The utilization of Web 2.0 technologies decrease the complexity of implementing the EA, reducing training and sustainment costs.

**Identification of Architectural Issues**

Identification of approximately 4-6 Major Architecture Issues associated with the project (and benchmarked “big picture” of which this project will be a part – provide a context for what part of a larger ecosystem you are addressing.

1. **Lack of Enterprise Architecture**
   Current environment was implemented without an Enterprise Architecture strategy resulting in a fractured ecosystem.
2. **Technical Debt**
   Current environment lacks resources and technical competency to address technical debt insolvency.
3. **Lack of Software Quality Processes**
   The lack of software development lifecycle processes effects software quality and increases maintenance and support costs.
4. **Software Complexity**
   Enterprise systems are tightly-coupled and lack cohesion causing significant software complexity increasing reliance on individual knowledge and support costs.
Implementing EA

LACK OF ENTERPRISE ARCHITECTURE

Literature Review

To enable transformation and agility, organizations require an effective Information Technology (IT) strategy for adequate decision making and expenditure of resources (Ross et al. 2006). Research conducted by MIT’s Sloan School’s Center for Information Systems Research shows that the most successful companies (in terms of profitability, time to market, and over all value of IT investments) digitized their core processes based on an effective strategy for enterprise architecture (Ross, et al. 2006). The development of a IT strategy is embodied in the discipline of Enterprise Architecture.

Enterprise Architecture (EA) is the culmination of processes, analysis, and documentation to ensure alignment of information systems to the organizations business goals and objectives (Townson 2008). The alignment of business and IT strategies, through a substantial EA approach, is critical to the integration and standardization of core business processes, lower technology costs, and increased value of the overall IT investment (Ross, et al. 2006). Simon Townson (2008) explains in Why does Enterprise Architecture Matter? The benefits of an Enterprise Architecture program includes:

- Helps achieve business strategy
- More consistent business processes and information across business units
- More reliability and security, and less risk
- Better traceability of IT costs
- Lower IT costs - design, buy operate, support, change
- Faster design and development
- Less complexity

Enterprise Architecture facilitates success through the effective use of information management strategies, resources and decision making (Hanifa and Kourdi 2013). The Enterprise Architecture permeates a holistic approach (total systems view) to provide flexibility, agility, and ultimately organizational transformation (Butler Group 2008). To develop an Enterprise Architecture program, frameworks are utilized to manage the complexity, processes, and artifacts across an organization.

Frameworks vary from industry specific to generalized as well as differentiate in their representation of business goals and architectures (business, information, and technology architectures) (Reynolds 2010). Examples of EA frameworks include: the Zachman Framework, the Open Group Architecture Framework (TOGAF), the Federal Enterprise Architecture (FEA), and the Gartner Enterprise Architecture Framework (GEAF). However, a literary review suggests that the implementation of such frameworks require time and resources (de Vries 2010), which are uncharacteristic of typical School Districts. Further studies conclude that socio-cultural alignment, the balanced IT contribution to changing stakeholder expectations, is unaccounted for in such frameworks (Magoulas et al. 2011). The selection of a EA framework within a School District must ensure: alignment of strategies, produce the necessary artifacts for implementation, be sustainable over time, simplify the complexity of EA, and be adaptable to the changing environment.
Organizational Issues
Results gathered from periodic student assessments and yearly State tests is manually collected or placed into application specific databases and inputted into the District supported Student Information System. However, not all schools or grade levels are required to place their information into the same system. Data has become disjointed and unavailable for reporting procedures and cannot adequately support administrative decision making. Information is scattered throughout the organization and centrally stored data lacks common definition or governance.

The District historically has acquired technology through a fractured acquisition process allowing both business units and schools to purchase solutions based on functional unit needs without a supporting Enterprise Architecture program. The current environment supports three database vendors, 302 different databases, 350 virtual machine servers, 2400 different reports, and 85 enterprise applications that utilize unique application servers from three different providers. All software systems are supported by 8 engineers for both project as well as operational and maintenance work.

Business Case
The District’s System Performance Framework is desired to meet mandates established by Federal and State law that are requiring greater accountability in education. The local State law specifically requires teachers evaluations to be based on 50 percent of student performance. To meet these mandates, information systems need to support data driven decisions based on a core infrastructure that can quantify key performance indicators (KPI). Teacher and administrative processes, with subsequent supporting information, require identification for standardization and integration to enable transformation goals. The information system environment requires realignment to the organizations strategies to ensure desired objectives are met. The realignment can only be accomplished through the establishment of a substantive Enterprise Architecture program.

Technical Debt
Technical debt is the monetary quantification of costs associated to refactoring production code to guarantee software quality. The term technical debt translates technical constraints in design and implementation to financial terms in order to quantify impact to stakeholders (Buschmann 2011). The debt equates to the amount of effort and resources necessary to redesign, recode, retest, and redeploy of prior solutions that disregarded good architectural and coding standards (Curtis 2012). Additionally, technical debt accounts for designs that inhibit a new functionality or feature from being instantiated. Interest accrues on the technical debt the longer the software quality issues are not addressed creating a persistent cost (O’Neill 2012). The longer the debt is neglected, the greater the increase of risk as well as the increase of both maintenance cost and refactoring (Buschmann 2011).

The phenomena of technical debt is caused from various factors that include technical proficiency, insufficient processes, complexity in design, project time constraints, and laziness (Curtis 2012). Although a portion of the debt can be the summative results of conscious decisions for
product delivery based on project timelines; due to the correlation of process and the accrual rate of the principal, a greater consequence is due to the maturity of technical management and software engineering processes (O’Neill 2013). Martin Fowler quantified technical debt into reckless and prudent actions based with deliberate and inadvertent consequences (Fowler 2009). Dr. Bill Curtis refined the quadrant (see figure 2: Curtis’ Technical Debt Quadrant) to be more applicable to decision making with correlation to refactoring prioritization (Curtis 2012). In his white paper, ‘Technical Debt’ in the Code, Dr. Curtis writes that technical debt is derived from three sources: process (see Appendix A), engineering (see Appendix B), and management (see Appendix C). Technical debt is a reflection of the quality of the product, processes used to create the product, and the team that creates it.

**Organizational Issues**

Technical debt is currently unaccounted for within the organization. The extreme numbers of supported systems per Full Time Equivalent (FTE) and the support of 6,700 employees places engineers in a reactive mode. Due to the lack of unaccounted technical debt and leadership pressures for immediate resolution, time is spent providing immediate fixes compounding the technical debt problem. Engineers work extensive hours to fix defects in code, creating sleep deprivation (sleep debt), which causing impaired cognitive functioning resulting in poor coding, adding to the debt principal. Customers and business owners are frustrated at the performance of systems and lack trust of the Information Technology department, yet engineers work extensive hours to support both new features and product maintenance. This causes a negative atmosphere and large turnover of technically proficient talent. Based on the People Capability Maturity Model (People CMM), developed by the Carnegie Melon University, the organization is at a maturity level 1 (see Appendix D) (Curtis, et al. 2001). As a result, the environment possesses the following characteristics (Curtis, et al. 2001, p. 19): 1. Inconsistency in performing practices, 2. Displacement of responsibility, 3. Ritualistic practices, and 4. An emotionally detached workforce. The technical debt is at a point of insolvency and will require significant effort to refactor the environment to ensure it can be adequately supported by the low number of support staff.

**Business Case**

Due to the direct correlation of an organizations ability to transform and information technology maturity (Ross, et al. 2006), both staffing and the technology space need addressing. Utilizing the People CMM as a roadmap for continuous workforce skill improvement, the District will need to supply adequate resources and funding for training and staff augmentation. In doing so, the Information Technology department and the systems they support will be able to continuous address the needs for organizational change.

**Lack of Quality Processes**

**Literature Review**

The online Merriam-Webster dictionary defines quality as: “peculiar and essential character”... “an inherent feature” (Merriam-Webster). Software quality is instrumental to successful software engineering (Jones 2010). Capers Jones provides a definition based on six characteristics (Jones 2010, p. 947-948):
1. Quality should be predictable before a software application starts.
2. Quality needs to encompass all deliverables and not just the code.
3. Quality should be measurable during development.
4. Quality should be measurable after release to customers.
5. Quality should be apparent to customers and recognized by them.
6. Quality should continue after release, during maintenance.

Software quality is a function of the software’s ability to meet functional and non-functional specifications (structural). Functional quality reflects a component’s compliance to intended behavior annotated as services, tasks, or functions as specified in user requirements. Where as, structural quality is identified in non-functional requirements stated as both qualities and constraints specified in architecture and system specifications (Malan and Bredemeyer 1999). The creation of high quality software is dependent upon substantive processes (requirements elicitation, design, code, test, deployment).

Fixing a software problem after delivery is more expensive than fixing it in early stages of initial development (large systems 100:1; small systems 5:1) (Boehm and Basili 2001). System rework and downtown can be avoided by including methodological requirements analysis, design, peer review and early user involvement in the software development process. Each technique identifies different defect classes at different points, therefore they should all be included in the software development lifecycle (Boehm and Basili 2001).

Organizational Issues
The current environment lacks critical processes of software development lifecycle. The development team utilizes no version control, peer reviews, nor substantive testing before placing components into production. In fact, the production environment is often used for development activities increase. There are no regression testing performed before deployment and no documentation standards. Lastly, there exists no configuration control or enforced change management processes.

Business Case
The lack of software development life cycle processes increases maintenance cost by a minimum factor of 5:1 (Boehm and Basili 2001). To support maintainability and sustainability requirements with the fewest staff members possible, SDLC processes need to be incorporated. The continuance of the current trend will result in more undocumented complexity and higher costs as time progresses.

Software Complexity

Literature Review
Based on the review of the literature, there is a high proclivity of poor software development processes, software complexity and technical competency to technical debt. Therefore, qualitative processes are required to moderate principal gains that will assist in identifying, preventing and resolving technical debt before reaching insolvency. Good design is indicative of process
and skills. Quality assurance processes incorporate peer reviews to provide an alternative perspective, provide component familiarity across the team. Peer reviews results in less rework, fewer defects, improved communication, and greater team cohesiveness (Kemerer and Paulk 2009).

Software complexity increases maintenance difficulties, defect rates, and creates atrophy that precludes adaptation (Jones 2010). Software complexity is the architectural summation of relationships and interconnections between elements utilized (paths, computations, inputs, outputs, data structures, etc.) in a given solution. The relationships between components are characterized by interdependence (coupling) and degree of functional similarity (cohesion). Coupling is the degree to which components require intimate knowledge of the internal implementation of another. The degree of coupling has a symbiotic relationship to the impact of changes of one component to its interrelated counterparts. Changes to a tightly coupled system increases the risk and complexity in isolating impact to the interrelated components. Conversely, loose coupling promotes a separation of concerns with changes isolated to the individual component (Mishara and Mohanty 2011).

Cohesion is determined by how strongly related functional capabilities are within a given component. If the similarity between a component’s methods (in terms of functionality) increase, then the component’s cohesion (weak versus strong) is reflected by the degree of equivalence. A component that has weak cohesion performs a multitude of tasks (data access, business logic, visualization, etc), and consequently increases its interrelationship between other components. As previously stated, interdependence increases the difficulty to understand and maintain. Conversely, a component with strong cohesion performs tasks that are singular in purpose and independent of others. Strong cohesion promotes reuse and decreases component complexity (Mishara and Mohanty 2011). Coupling and cohesion are inversely related; tight-coupled systems have weak cohesion and higher degrees of complexity. Software complexity increases risk to change whilst decreasing understanding of the environment (Mishara and Mohanty 2011). Software complexity increases sustainment costs due to reliance on individual understanding and reduces successful cross-training for support (Jones 2010).

Organizational Issues
Existing personnel are reactive to the needs of users at the expense of quality, increasing sustainment and maintenance costs. Personnel lack the technical competency in understanding alternative design patterns other than tight coupling within the data layer between components. Personnel also lack substantiative technical competence with 4th generation programming languages and require proprietary systems for development using PL/SQL. Substantive security policies are non-existent; systems and data at the risk of potential malfeasance.

Business Case
The organization must address software complexity by staff augmentation and training to enable organizational transformation. Not addressing complexity increases cost and decreases operation performance of business essential applications. To continue will progressively increase the risk of data loss and security resulting in lost revenue and potential litigation due the lack of compliance with State and Federal standards.
Recommended Solutions

**CONCLUSION**

In summary, technical debt is the monetary quantification of costs associated to refactoring production code to guarantee software quality. Good design is indicative of both process and skills, and the District will need to support both professional development and staff augmentation of the IT department. Quality assurance processes incorporate peer reviews and results in less rework, fewer defects, improved communication, and greater team cohesiveness. Software complexity and the degree of both coupling and cohesion increases the proclivity of technical debt. The IT department needs to instantiate substantiative changes in software development processes to reduce complexity and to ensure overall quality, maintainability, and sustainability. The creation of an Enterprise Architecture (EA) program will ensure the alignment of business and IT strategies, and result in the integration and standardization of core business processes. The instantiation of modern design patterns and technology will provide the means to integrate legacy applications, reduce complexity in the enterprise environment, increase system flexibility, and reduce overall IT costs. These efforts will maximize the return on investment, ensure appropriate security mechanisms, and enable organizational transformation.
# Roadmap to EA

**Project:** Enterprise Architecture  
**Stakeholder:** CIO

<table>
<thead>
<tr>
<th>Duration</th>
<th>Phase</th>
<th>Activity</th>
<th>Deliverables</th>
</tr>
</thead>
</table>
| 6 months | Phase 1: - Preliminary; Architecture Vision; | * Identification of Business Drivers  
* Prioritization of Enterprise Architecture Capabilities  
* Instantiate SDLC processes  
* Select SDLC tools  
* Training: Shell Scripting and BASH commands  
* Establish QA Team | * SDLC Process  
* Configuration Management  
* QA Team  
* Enterprise Architecture Stakeholders Matrix; Enterprise Capability Model; Enterprise Architecture Maturity Assessment; Architecture Governance Framework; Architecture Principles; Tailored Architecture Framework; Architecture Vision; Draft Architecture Definition Document; Communications Plan |
Project: Enterprise Architecture  - Stakeholder: CIO

<table>
<thead>
<tr>
<th>Duration</th>
<th>Phase</th>
<th>Activity</th>
<th>Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 months</td>
<td>Phase 2: Business Architecture</td>
<td>* Select Reference Models, Viewpoints, and tools</td>
<td>* Software Engineering PD program</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Develop Target Business Architecture Description; Perform Gap Analysis; Define Candidate Roadmap Components; Conduct Formal Stakeholder Review; Finalize Business Architecture; Create Architecture Definition Document</td>
<td>* Organization / Actor Catalog; Driver/Goal/Objective Catalog; Role Catalog; Business Interaction Matrix; Actor/Role Matrix; Functional Decomposition diagram; Use-Case Diagrams; Process Flow Diagrams; Event Diagrams</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Training: Basic Java; SDLC tools</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Develop Software Engineering Professional Development program</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Select Automated Testing Tools</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Develop baseline architecture description; Develop Target Architecture Description; Conduct Formal Stakeholder Review; Finalize Architecture; Create Architecture DD</td>
<td>* Draft Data / Application Architecture Definition Document; Draft Architecture Requirements specification; Information systems components of the Architecture Roadmap</td>
</tr>
<tr>
<td>Duration</td>
<td>Phase</td>
<td>Activity</td>
<td>Deliverables</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 2 months | Phase 5: Opportunities and Solutions      | * Training: WOA components; SDLC process  
* Determine business constraints for implementation; Review consolidate requirements; Consolidate and reconcile interoperability requirements; Refine and validate dependencies; Confirm readiness and risk for business transformation; Formulate Implementation and Migration Strategy; Identify Transition Architectures; Create the Architecture Roadmap & Implementation and Migration plan | * Capability Assessments; Architecture Roadmap; Benefits Diagram                                                                                             |
| 2 months | Phase 6: Migration Planning               | * Training: Enterprise Service Bus and Design Patterns  
* Confirm management framework interactions for Implementation and Migration Plan; Assign business value to each work package; Estimate resource requirements, project timings, and availability; Conduct cost/benefit and risk validation; Complete Implementation and Migration Plan; Complete the architecture development cycle and document lessons learned | * Implementation and Migration Plan; Project and portfolio breakdown; Project Charters; Finalized ADD; Finalized ARS; Finalized Architecture Roadmap; Implementation Governance Model; Change Requests |
| 3 months | Phase 7: Implementation Governance        | * Training: PHP; Governance  
* Confirm scope and priorities; Identify deployment resources and skills; Perform enterprise architecture compliance reviews; Implement business and IT operations; Implement Review | * Architecture Contract; Compliance Assessments; Change Requests; SLA(s); Recommendations on performance metrics; updated ADD; Business and IT operations models for the implementation solution |
|          | Phase 8: Architecture Change Management   | * Deploy Monitoring Tools  
* Manage Risks  
* Provide analysis for architecture change management  
* Develop change requirements  
* Manage governance process | * Architecture updates  
* Changes to architecture framework and principles  
* Compliance Assesments |
### Appendices

#### Appendix A: Technical Debt - Process

<table>
<thead>
<tr>
<th>Source</th>
<th>Trigger</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>Software Project Management</td>
<td>Where the software project management mode is low</td>
</tr>
<tr>
<td></td>
<td>Software Product Engineering</td>
<td>Where the software product engineering mode is ad hoc.</td>
</tr>
<tr>
<td></td>
<td>Iterative Development</td>
<td>Where incremental or iterative development of design levels and delivery stages is not used</td>
</tr>
<tr>
<td></td>
<td>Best Practices</td>
<td>Where the use of best practices is rated low</td>
</tr>
<tr>
<td></td>
<td>Metrics</td>
<td>Where metrics are not used</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td><strong>Trigger</strong></td>
<td><strong>Condition</strong></td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Quality Assurance</td>
<td>Where quality assurance is not in place and functioning</td>
</tr>
<tr>
<td></td>
<td>Defect Rate</td>
<td>Where the actual defect rate including both defect detection and defect correction exceed the expected</td>
</tr>
</tbody>
</table>

Table 1 *Source:* Adapted from Dan O’Neill, ‘*Technical Debt in the Code: The Cost to Software Planning.*’ Defense AT&L: March-April 2013. Table 3.
# Appendix B: Technical Debt - Engineering

<table>
<thead>
<tr>
<th>Source</th>
<th>Trigger</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering</td>
<td>Deep domain expertise</td>
<td>Where deep domain expertise is not widespread on the project</td>
</tr>
<tr>
<td>Software Architecture</td>
<td></td>
<td>Where software architecture is not tightly coupled with middleware, operating system, and network services</td>
</tr>
<tr>
<td>Requirements known</td>
<td></td>
<td>Where requirements are not fully known</td>
</tr>
<tr>
<td>Technical risk</td>
<td></td>
<td>Where the source of technical uncertainty in function, form or fit is high</td>
</tr>
<tr>
<td>Product size</td>
<td></td>
<td>Where product size estimates at completion exceed product size estimates planned</td>
</tr>
<tr>
<td>Complexity</td>
<td></td>
<td>Where cyclomatic or essential complexity trend upward from one product release to another</td>
</tr>
</tbody>
</table>

## Appendix C: Technical Debt Management

<table>
<thead>
<tr>
<th>Source</th>
<th>Trigger</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>Prioritized Goals</td>
<td>Where Schedule or cost is accorded priority over defect free delivery</td>
</tr>
<tr>
<td></td>
<td>Organization Levels</td>
<td>Where the software function is separated from program management by two or more levels.</td>
</tr>
<tr>
<td></td>
<td>Schedule</td>
<td>Where the number of months planned is less than the estimated month at completion</td>
</tr>
<tr>
<td></td>
<td>Cost</td>
<td>Where the budget at completion is less than the estimate at completion</td>
</tr>
<tr>
<td></td>
<td>Milestone completion</td>
<td>Where the completion schedule for any milestone completion planned date is replaced with a re-planned date</td>
</tr>
<tr>
<td></td>
<td>Headcount and effort</td>
<td>Where overtime, off-the-clock time, and personnel turnover rate is trending upward</td>
</tr>
<tr>
<td></td>
<td>Frequency of release</td>
<td>Where the frequency of release is daily or weekly</td>
</tr>
</tbody>
</table>

Table 3 Source: Adapted from Dan O’Neill, ‘Technical Debt’ in the Code: The Cost to Software Planning. Defense AT&L: March-April 2013. Table 1.
### Appendix D: People CMM Matrix

<table>
<thead>
<tr>
<th>Maturity Level</th>
<th>Focus</th>
<th>Process Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 Optimizing</td>
<td>Continuously improve and align personal, workgroup, and organizational capability</td>
<td>Continuous Workforce Innovation Organizational Performance Alignment Continuous Capability Improvement</td>
</tr>
<tr>
<td>4 Predictable</td>
<td>Empower and integrate workforce competencies and manage performance quantitatively</td>
<td>Mentoring Organizational Capability Management Quantitative Performance Management Competency-Based Assets Empowered Workgroups Competency Integration</td>
</tr>
<tr>
<td>3 Defined</td>
<td>Develop workforce competencies and workgroups, and align with business strategy and objectives</td>
<td>Participatory Culture Workgroup Development Competency-Based Practices Career Development Competency Development Workforce Planning Competency Analysis</td>
</tr>
<tr>
<td>2 Managed</td>
<td>Managers take responsibility for managing and developing their people</td>
<td>Compensation Training and Development Performance Management Work Environment Communication and Coordination Staffing</td>
</tr>
<tr>
<td>1 Initial</td>
<td>Workforce practices applied inconsistently</td>
<td></td>
</tr>
</tbody>
</table>


---

Names of organization, specific divisions, and employee names are kept confidential out of respect for NDA with the client.
## Appendix E: Role-System Matrix

<table>
<thead>
<tr>
<th>Physical Application Components Map to Roles</th>
<th>AP</th>
<th>HR</th>
<th>Purchasing</th>
<th>Warehouse</th>
<th>IS</th>
<th>Employee</th>
<th>Manager</th>
<th>Teacher</th>
<th>Principal</th>
<th>Specialist</th>
<th>Student</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERP AP</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERP GL</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERP Payroll</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBP HR</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Google Docs</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Google Search</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Hapagen (group)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Taleo (HICE)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Kronos</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPED</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISB</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LMS (Student)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LMS (FRO)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PaceDNA</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EmpTime</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Magica Carta</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ePassport</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RevTrak</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Employee Self Service</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eBenefits</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portal</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Ecosting</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Names of organization, specific divisions, and employee names are kept confidential out of respect for NDA with the client.
Appendix H: - Future Architecture

- Presentation Services
  - Mobile
  - Portal

- Line of Business Systems
  - ERP
  - SIS
  - SPED
  - Custom Applications

- Third Party Applications
  - HR
  - Payment Services

- Connectivity and Interoperability

- Metadata Services
- Master Data Services
- Data Services
- Content Services
- Content Workflow
- Content Data
- Content Services
- Operational Data

- Enterprise Information Integration

Notes:
- Names of organization, specific divisions, and employee names are kept confidential out of respect for NDA with the client.
APPENDIX I: USE CASE DIAGRAM
REFERENCES


