UST Consulting

Version – 1.0

[SAFEWAY J4U NEXT GENERATION ARCHITECTURE ASSESSMENT]

This document is an assessment of Safeway J4U Next Generation loyalty application which enables UI associated with clipping coupons to be brought in house. The assessment reviews the overall architecture and system interactions. The J4U application is also evaluated on the overall architecture, specifically around, reliability, performance, availability, security, supportability, extendibility, etc.

Table of Contents

Table of Figures

The graphic on the left with its associated star rating of 1 to 5 stars is used in this document to depict UST's evaluation of the success of a particular area.

Executive Summary of Findings

During the assessment of Safeway's J4U Next Generation project, we discovered several key success factors that were already in place and operating smoothly:

- Best of breed technologies (Oracle database, WebLogic, Spring, Hibernate, etc.)
- Smart, dedicated associates who want to see the project succeed. They have a can-do and focused attitude.
- Pragmatic approach to systems development: architecting for tomorrow, while developing for today.
- Well defined scope; the project is clearly envisioned, understood and communicated.

Although we did not encounter any weakness which would be, in our view, significant enough to stop the project from being successful, we did find several items that need attention. These are summarized below and explained throughout the document

Categories of Findings:

1. Environment
2. Implementation
3. Process
4. Communication

Time Horizons:

1. Immediate – Addressing these items will show results immediately
2. Short-term – Addressing these items will show results within a month
3. Mid-term– Addressing these items will show results in one to three months
4. Long-term – Addressing these items will show results in three plus months

Recommendations

Recommendation	Impact	Category	ROI	Reference
Use Automated Provisioning Tools	High	Environment	Short-term	Section 5.1.1
Use Configuration Management	High	Environment	Short-term	Section 5.1.1
Use Existing Tools	High	Environment	Short-term	Section 5.1.1
Document Systems and Processes	High	Environment	Mid-term	Section 5.1.1
Use stable OSSO environment	High	Environment	Immediate	Section 5.1.2
Use mock objects	Medium	Implementation	Immediate	Section 5.1.2

Hire Experts	High	Environment	Immediate	Section 5.1.2
OSSO Viability	Low	Environment	Mid-term	Section 5.1.2
Expand Unit Tests	High	Implementation	Immediate	Section 5.1.2
Generate Health Checks	Low	Implementation	Immediate	Section 5.1.2
Use Simple Continuous Integration	High	Implementation	Immediate	Section 5.1.2
Add Additional Services to Build Process	Medium	Implementation	Mid-term	Section 5.1.2
Perform Additional Analysis of Development Methodology	Unknown	Implementation	Mid-term	Section 5.1.2
Create Program Project Plan	Medium	Communication	Immediate	Section 5.1.3
Use Matrixed Resources	Medium	Communication	Short-term	Section 5.1.3
Use Project Portal	Medium	Communication	Short-term	Section 5.1.3
Create RACI Matrix	Medium	Communication	Immediate	Section 5.1.3
Coordinate Effort	Medium	Communication	Immediate	Section 5.1.3
Build Information Contexts for J4U – Readme First	Low	Communication	Long-term	Section 5.1.3
Conduct Design Reviews	Medium	Implementation	Immediate	Section 5.2.1
Conduct Code Reviews	Medium	Implementation	Immediate	Section 5.2.1
Create Component Scalability Tests	High	Implementation	Immediate	Section 5.2.1
Leverage JUnit for Scalability Tests	High	Integration	Immediate	Section 5.2.2
Use Credential Caching	High	Integration	Short-term	Section 5.2.2
Decouple System Dependencies	Medium	Integration	Mid-term	Section 5.2.2
Test for Failover and Error Scenarios	High	Integration	Immediate	Section 5.2.2
Detail Monitoring Specifications	Medium	Environment	Short-term	Section 5.2.3
Establish Metrics and Thresholds	Medium	Environment	Immediate	Section 5.2.3
Do Not Monitor MediaBin	Low	Environment	Long-term	Section 5.2.3
Define Scope of Monitoring	Medium	Environment	Short-term	Section 5.2.3
Define Alerts and Notification Systems	Medium	Environment	Short-term	Section 5.2.3
Define System Monitors	Medium	Environment	Short-term	Section 5.2.3
Freeze and Deliver Approach	High	Environment	Immediate	Section 5.3.1
Organize Structure and Environment “spin-up”	High	Environment	Short-term	Section 5.3.1
Document Requirements	High	Environment	Short Term	Section 5.3.1
Leverage Existing Tools	High	Environment	Mid-term	Section 5.3.1

Create Disaster Recovery Plan	High	Environment	Mid-term	Section 5.3.1
Establish Enterprise Monitoring	High	Environment	Mid-term	Section 5.3.1
Create Service Management Processes	High	Environment	Long-term	Section 5.3.1
Implement Full Redundancy	High	Environment	Long-term	Section 5.3.1
Establish Availability Reporting	High	Environment	Long-term	Section 5.3.1
Document all Error and Exception Paths	High	Implementation	Short-term	Section 5.3.2
Randomly Test System Outages	High	Implementation	Immediate	Section 5.3.2
Leverage Scalability Tests	High	Implementation	Immediate	Section 5.3.2
Create Continuation Strategies	High	Implementation	Short Term	Section 5.3.2
Run Load Balancers at 45% Capacity	Medium	Environment	Immediate	Section 5.3.2
Create an Emergency Demand Capacity Environment	Medium	Environment	Mid-term	Section 5.3.2
Use Lightweight Services	Medium	Implementation	Mid-term	Section 5.4.1
Use Domain Models	Medium	Implementation	Long-term	Section 5.4.1
Create an Enterprise Architecture Roadmap	Medium	Implementation	Long-term	Section 5.4.2
Create EA Projects	Medium	Implementation	Short-term	Section 5.4.2
Perform Additional Analysis	Unknown	Implementation	Short-term	Section 5.4.2
Establish Process Framework	High	Implementation	Long-term	Section 5.4.3
Perform an in-depth Analysis of Data Practices	Unknown	Implementation	Short-term	Section 5.4.4
Establish Data Tzar Role	Unknown	Implementation	Long-term	Section 5.4.4

Checklist for Frequent Tasks

Task	Frequency
Build code running unit and integration tests	Daily
Stand-up meetings between APPDEV and INFRA till clarity in requirements and expectations are met.	Daily
Code review (includes reviewing code coverage and cod health metrics)	Weekly
Design Review	Weekly
Run, performance, environment, scalability tests	Weekly
Synchronize project specific project plans with program project plan	Bi-Weekly

1 Assessment Overview

1.1 Scope

Perform time-boxed three week best effort technology assessment of the Next Gen J4U Loyalty Architecture and provide documentation of findings, including:

1. Evaluation of technology architecture for bringing Coupons Inc. functionality in house, specifically the user interface.
2. Evaluate Loyalty technology integration plan between data, network, infrastructure and the application stack (System Integration).
3. Provide guidance and gap analysis for loyalty technology architectural suitability to meet Safeway business demands, especially with regard to performance.
4. Ensure that plans for testing / quality assurance and support are adequate and that the plans meet the needs of the business plan.

During our three week assessment of the J4U Next Generation Application, our objective was to identify:

- Key areas of concern
- Findings that could potentially detract from delivering functionality or adhering to the timeline
- Problems in the underlying Loyalty architecture

Our assessment included the following steps:

1. Interviewing key stakeholders from all relevant internal Safeway teams
2. Reviewing documentation, including architecture, design, support, training and implementation materials
3. Summarizing findings
4. Presenting findings and recommendations
5. Compiling a detailed written report (this document)

2 UST Assessment Approach

UST has an established a proven approach for assessing the architecture of an application (diagram below). This standardized method has discrete, highly structured stages and has been customized to the needs of Safeway's application. The schematic below describes the stages of the assessment:

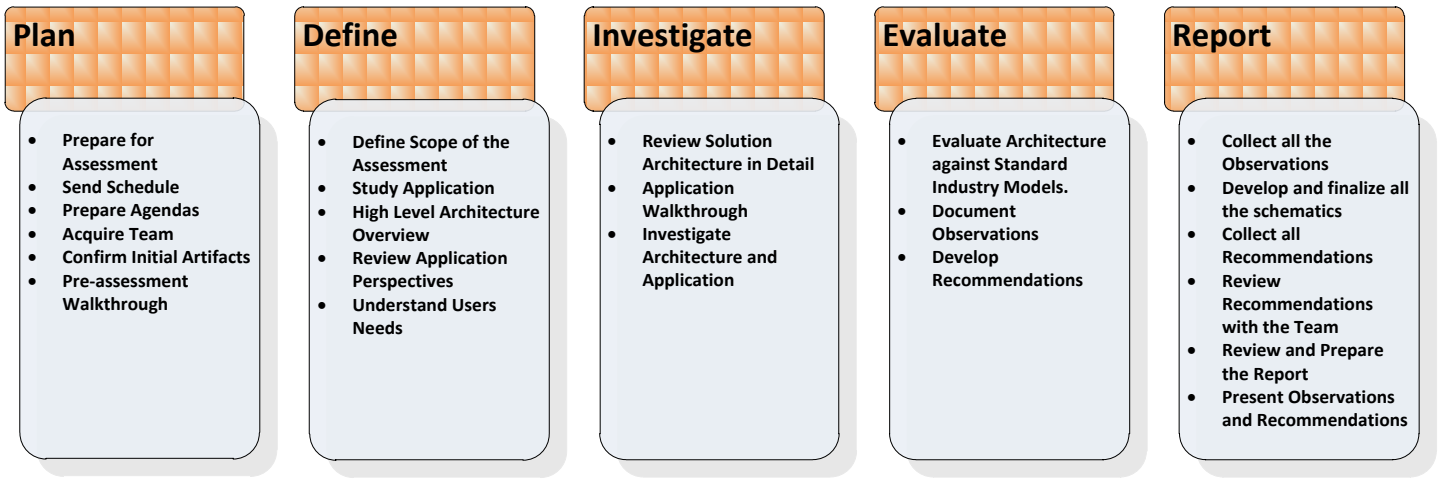


Figure 1: UST Global Architecture Assessment Approach

Following is a description of each stage of assessment.

Plan – Prepare for the assessment by setting up meetings with the key stakeholders. Safeway determined the appropriate stakeholders for the assessment. The agendas for the meetings were provided prior to the first week.

Define – Study of the 'As-Is' architecture of the Safeway application. The typical inputs are the initial vision documentation, the business process pertaining to this application, the established success factors and the current state of the application. Discussions with the key stakeholders covered topics such as value received, value provided, and overall perceptions on the application.

Investigate – Perform the initial assessment of the application architecture by reviewing the architecture artifacts and discuss with the key team members associated with the solution architecture of the application. All information pertaining to the current state and future state is captured and assessed against standards and guidelines

Evaluate – Perform the assessment of the solution by reviewing the current state of the architecture, the features and functionalities designed, the technology stack, the standards and guidelines applied, the use of the defined application architecture or enterprise architecture and the associated reference model.

Report – Prepare the assessment report by aggregating all the facts and preparing the narratives. All the quantitative and qualitative metrics and associated recommendations are prepared for the application.

Presentation – Present the assessment, the findings and the recommendations to the stakeholders and project sponsors.

2.1 UST Global Roles and Responsibilities

Listed below are the names, roles and responsibilities of the UST Global team members.

Name & Title	Role	Responsibility
Jim Mack **Enterprise Architect**	Lead Assessor	Assesses the Safeway application from a process, architecture and functional perspective
Vasu Vijay **Infrastructure Practice Lead**	Infrastructure Assessor	Assesses the Safeway application from an infrastructure perspective
Kuruvilla Mathew **Ecommerce Practice Lead**	Assessor	Assesses the Safeway application from an architecture and functional perspective
Ryan Prindiville **Safeway Relationship Manager**	Coordinator	Coordinates the engagement.

3 Assessment Methodology

3.1 Documents Used During the Assessment

These are the artifacts and documents referenced while performing the assessment.

1. 11578 Customer Portal Migration – SAD.doc
2. Business and Data Rule.xls
3. CIMeetingDiagrams-v1.vsd
4. CIReLuanchProjections.xls
5. Copient Central AsIs (v1.0).vsd
6. CSR-J4Uintegration.vsd
7. CustomerPortalJFU-V1.vsd
8. J4U Business Architecture Context V1.vsd
9. J4U Next Gen LDM as of 04262011.doc – Data Dictionary (Business terms)
10. J4U Next GEN LDM as of 04262011.pdf – ERD Diagrams
11. J4U NextGen 11662 - SAD.doc
12. J4Ui-SequenceDiagrams-V2.3.vsd
13. Just4You Logical Data Model.pdf
14. Loyalty Target Application Architecture-v1.vsd
15. Loyalty_PHX-DC_Architecture_Detailed_Design_ver_0_9 1.vsd
16. LoyaltyTTL.GEN+1.V1.3.vsd
17. offermgmt-V1.vsd
18. PerformanceObjectivesBasedonActuals-MLv3.xls
19. Program Planning (UST) v1.ppt
20. T-LayerNFRs.xls
21. UST Briefing Deck.pptx

3.2 Safeway Stakeholders and Key Associates

During our assessment of the J4U Next Gen application, we met with the IS product sponsor, the architects, quality assurance, project management, and the development leads. Listed below are their names, roles and responsibilities.

Name & Title	Role	Responsibility
David Ching **Chief Information Officer**	Safeway Primary IS Stakeholder	Safeway Executive Technical Vision and Information Services Execution
Carrie Rasmussen **VP of Services and Support**	Safeway Loyalty Program Lead	Safeway Technical Vision, Program responsibility and overall Safeway responsibility

Lawrence Loo **Lead Architect**	Safeway Design Assurance: Lead Architect	Designs technical architecture.
Abhishek Ranjan **Business Architect**	Safeway Design Assurance: Business Architect	Design of business technical architecture. Creates Business Domain Models.
Zeny Chan **Information Architect**	Safeway Design Assurance: Information Architect	Design of logical enterprise data structures
Michael Langois **Applications Architect**	Safeway Design Assurance: Application Architect	Designs the project specific architecture from the high level enterprise architecture and direction. Acts as the bridge between architecture and implementation.
Paul Rarey **Technology Architect**	Safeway Design Assurance: Technology Architect	Design of high level technical architecture.
Victor Tayao	Safeway Design Assurance: ISE Team	Designs the project specific technical architecture from the enterprise architecture. Creates project specific reference architecture for environment builds and rollout.
Brian Mundy **Business Architect**	Safeway Loyalty Strategy and Program Planning	Design of business technical architecture. Creates Business Domain Models.
Kalene Gomez	Safeway Delivery Assurance: Cross-IT	
Mike Friedel **Application Support**	Safeway Delivery Assurance: Support	Manages the support team which handles the day to day operations of supports applications after they go live.
Linda Wheatley **QA Lead**	Safeway Technical Quality Assurance: Test Lead	Manages the Quality Assurance team. Ensures that the project meets the requirements.
David Arbo Network Architect	Safeway J4U Network Architect	Network and Infrastructure Architect
Sujith Thulaseedharan Delivery Manager	UST Delivery Manager	Manages the UST resources that are engaged at Safeway

Dave Thomas Development Director	Safeway Development Director	Oversees all aspects of software development. Including development and QA.
Ed Lau DBA	Safeway J4U DBA	Application DBA for the J4U project.
Faye Taidi Project Lead	Safeway J4U Next Gen Project Lead	Manages the J4U next generation project deliverables and resources.
Rita Patel Project Lead	Safeway J4U Next Gen Project Lead	Manages the Loyalty program deliverables and resources.
Mark McKamey TSG Director	Safeway TSG Director	Manages TSG resources and network build out
Melanie Georgiani TSG Director	Safeway TSG Director	Manages TSG engagement on Loyalty
Michael Wolfson GVP TSG	Safeway GVP TSG	Manages all TSG Resources
Mark Grovhoug TSG Architect	Safeway TSG Architect	Manages Remodel Architecture
Houston Pagtakhan Delivery Architect	Safeway Delivery Architect	Manages Remodel Program Standards
Marcel Legee Delivery Architect	Safeway Delivery Architect	Manages Remodel Program Standards

3.2.1 Team Organization Chart

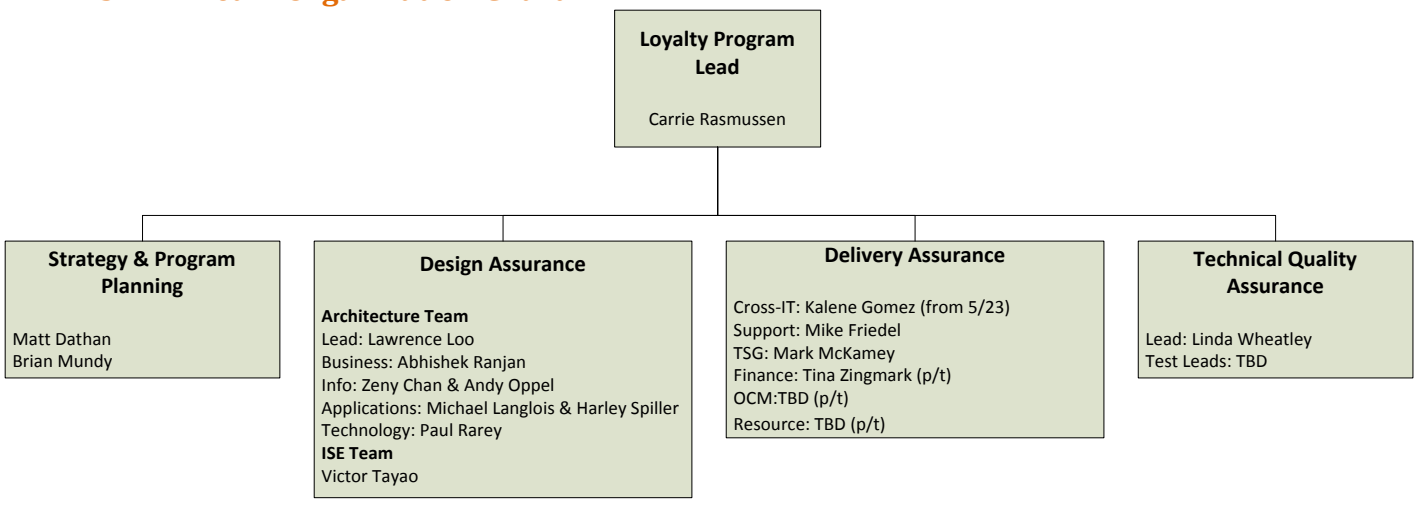


Figure 2: J4U Next Generation Organizational Chart

4 Application Overview

This section describes the high level architecture of the Just For You program. Later in this document when we discuss our observations and recommendations, we will refer back to this section.

4.1 Application Overview

Safeway customers have the ability to select coupon offers online. This is called the Just For You (J4U) program. The selected coupons are then associated to the customer's loyalty card for use in the store. The purpose of the J4U Next Generation project is to bring the coupon selection UI (user interface) portion of the J4U loyalty application in house. This function is currently being performed by Coupons Inc., an outside vendor, and is being brought in house because their performance has been less than optimal. The existing back-end systems described in the next few sections will remain unchanged when this functionality is brought in house.

4.2 Business Architecture

The business architecture defines how the application fits with the rest of the enterprise from a business perspective. The business perspective needs to be understandable by key business stakeholders and show system dependencies, high level system interactions, process and data flow.

4.2.1 High Level Systems Interaction: Customer Perspective

The diagram below shows how the customer would interact with J4U Next Generation.

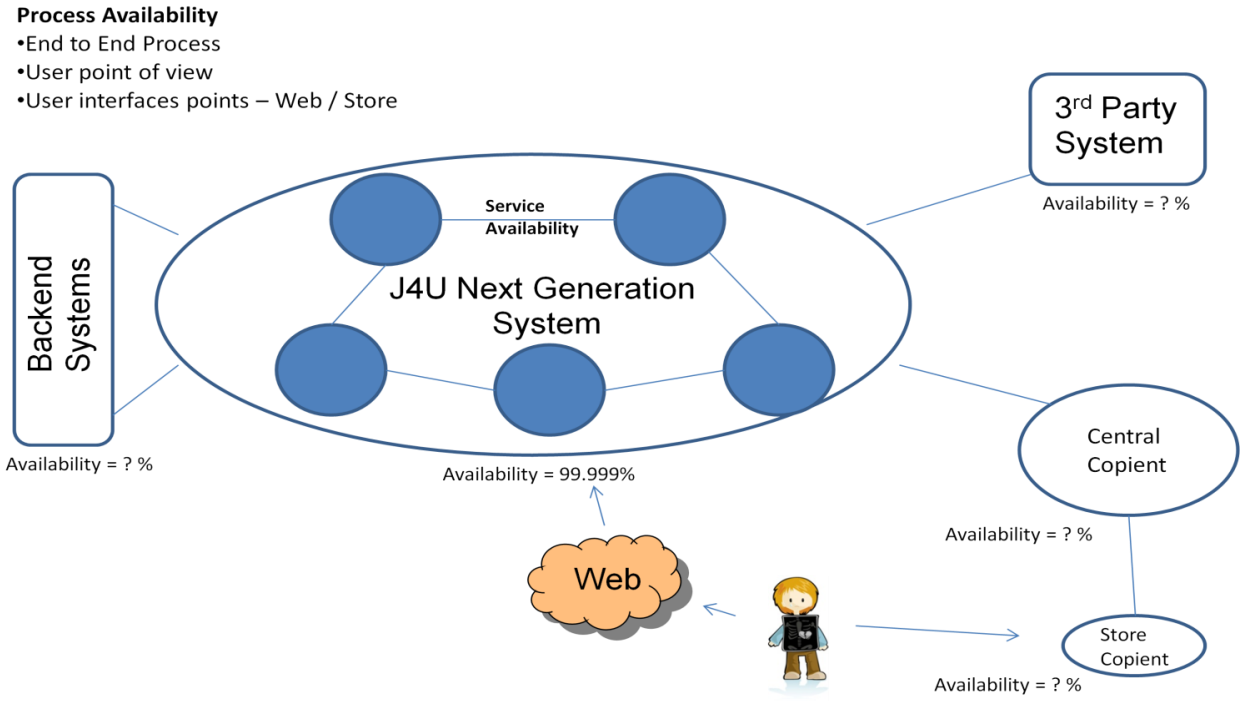


Figure 3: J4U Next Generation High Level System Interaction: Customer Perspective

4.2.2 Dependency Map

The diagram below illustrates how the J4U system integrates with the enterprise business applications.

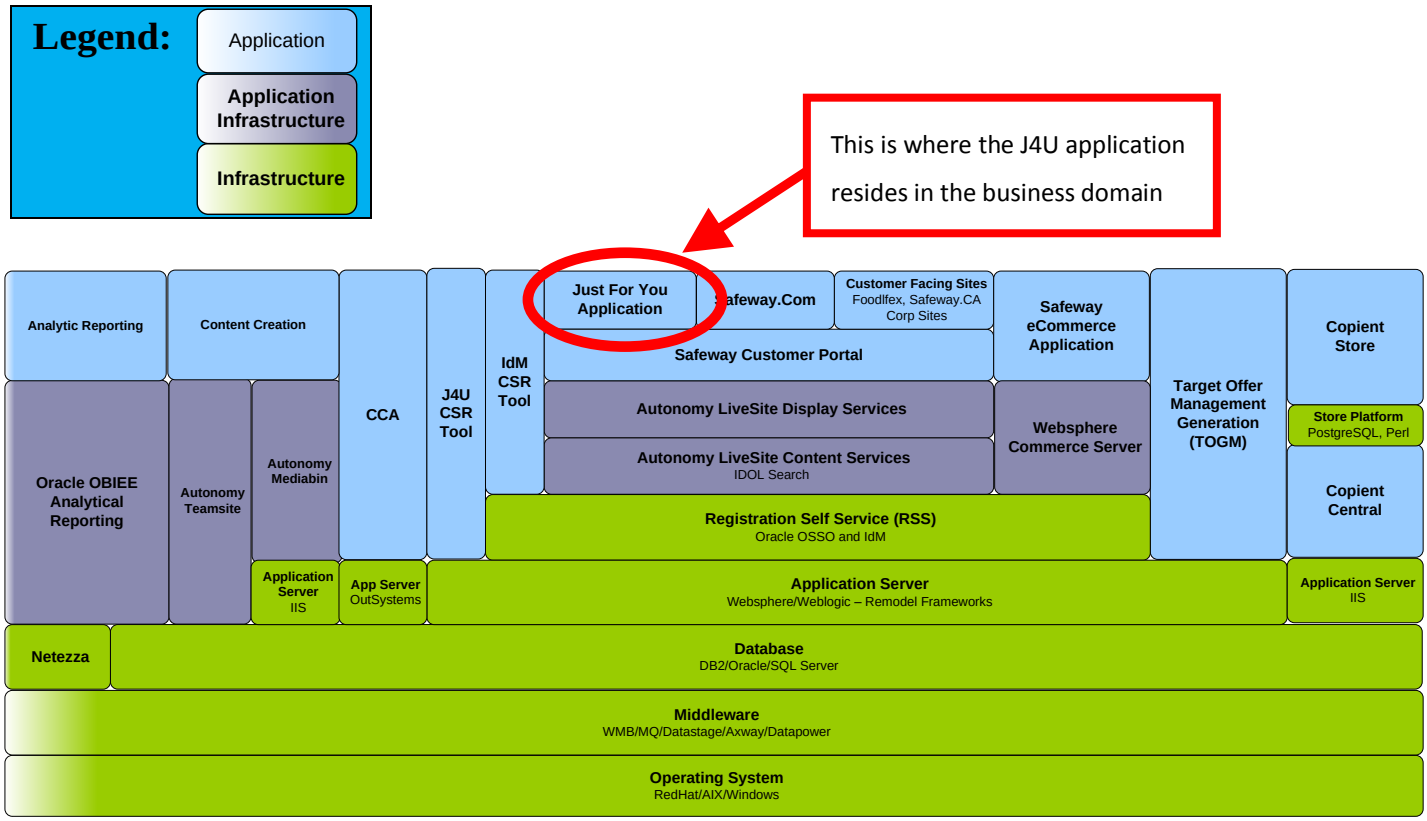


Figure 4: High level Business Dependency Map

4.3 Information Architecture

The data architects create high-level logical models that span the entire Loyalty program. The development team and an embedded DBA turn the logical model into a physical representation for each project. We were unable to review the logical model in depth due to time constraints. The physical model was also not reviewed as it had not been created at the time of our evaluation.

4.4 System Architecture

4.4.1 Systems Interaction: Technical Perspective

The diagram below shows the overall systems architecture including J4U along with external and internal systems. For a larger version of picture, please refer to the attachment - j4u systems arch.ppt

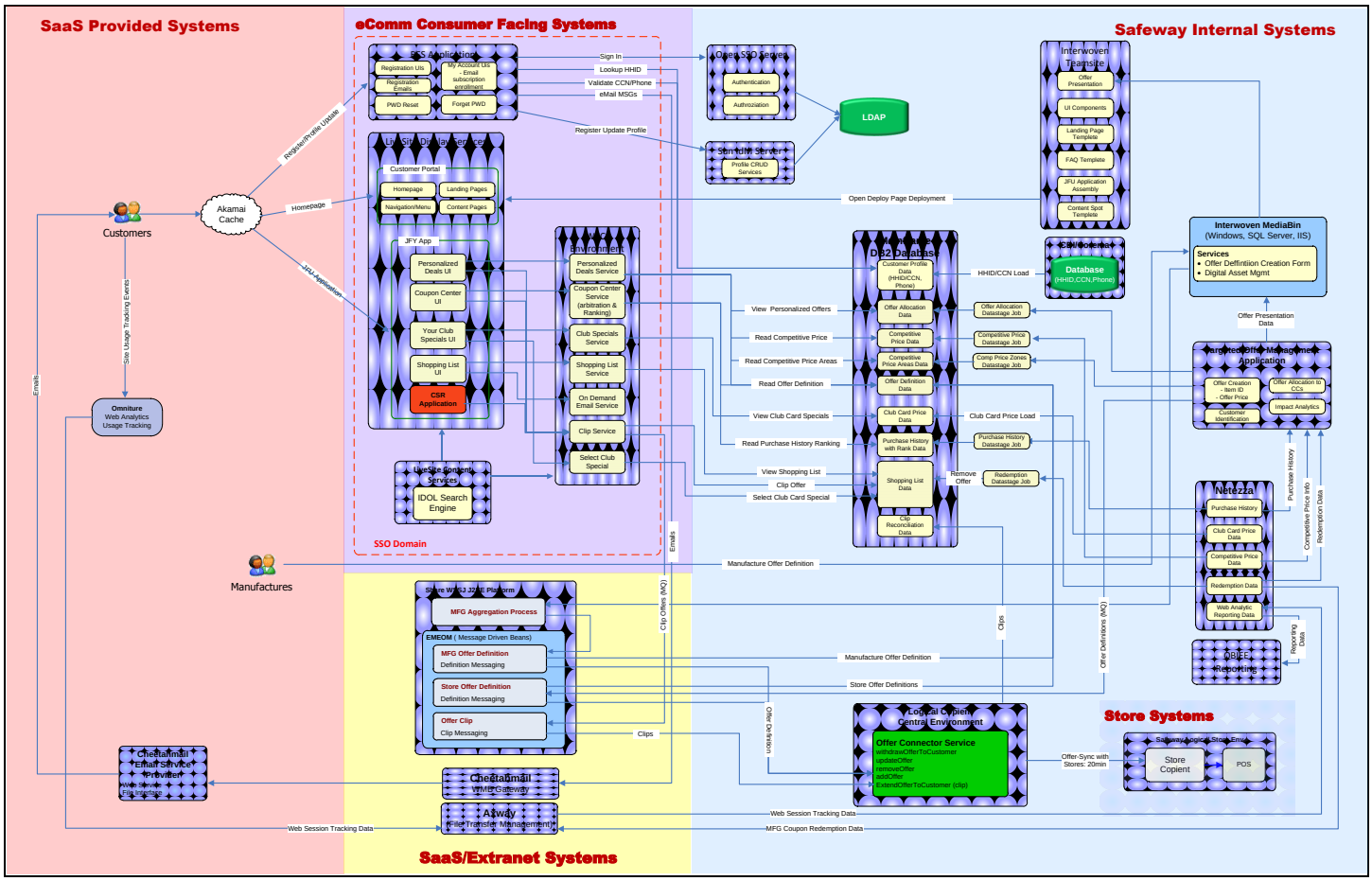


Figure 5: J4U Next Generation Systems Architecture

4.5 Technology Architecture

A new UI will be created as part of Next Generation. The UI will be portal based with custom AJAX components that will call back end services. The services are a mixture of Web Services and REST Services. The application is written in JAVA, JSP, JavaScript and SQL, leveraging the Spring (Dependency Injection) and Hibernate (Object to Relational Mapping ORM) frameworks.

Data will be stored in an Oracle 11g. J4U interacts with several keys external systems, diagrammed in figures 5 and 6.

4.5.1 Logical System Interaction Architecture

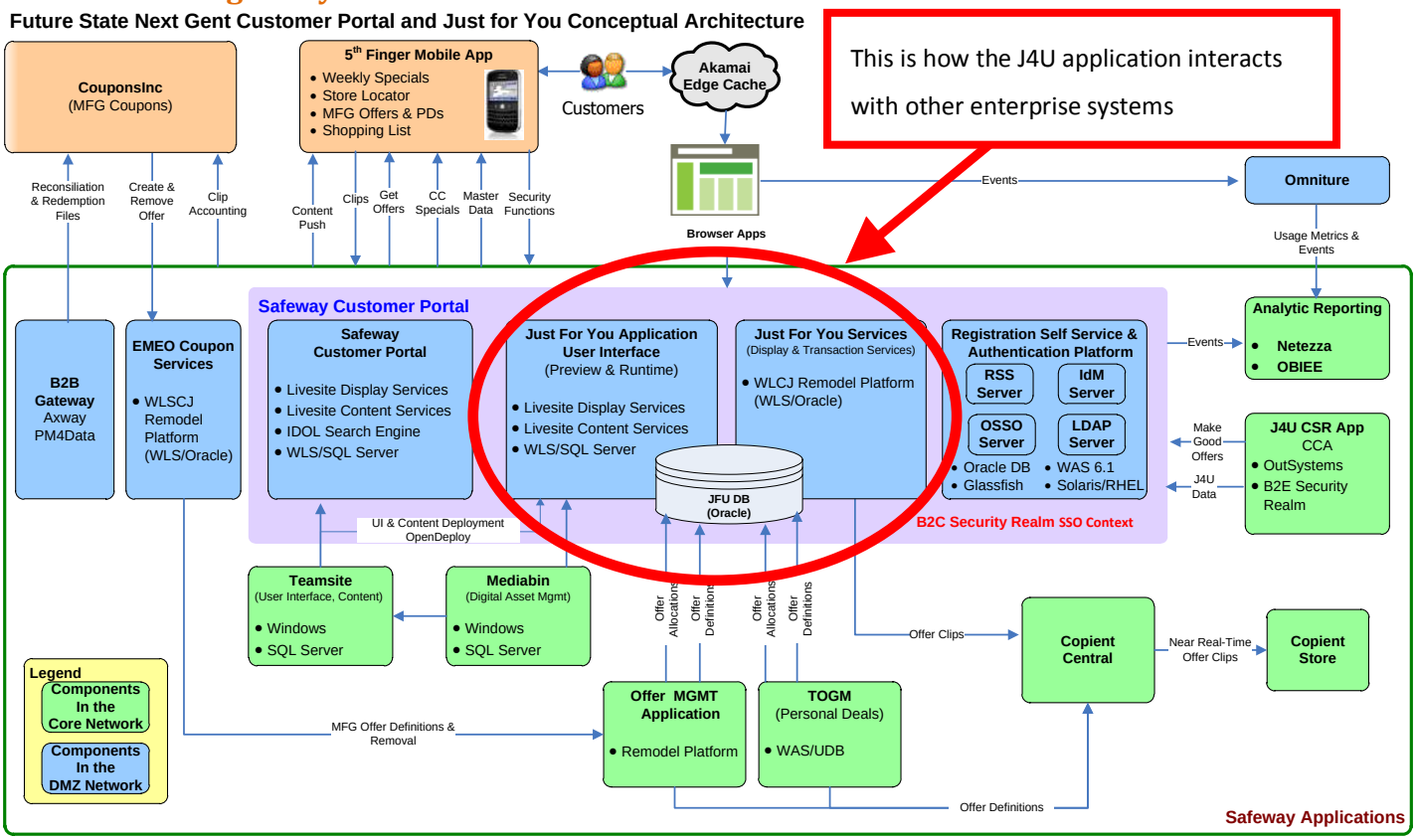


Figure 6: J4U Next Generation Logical System Interaction Diagram

5 Assessment of the Overall Architecture

UST was asked to perform an architectural assessment of the J4U Next Generation application to confirm that Safeway is proceeding appropriately to ensure the success of the application. Safeway's success for this project is defined by the answers to three questions:

1) Will the project deadlines be met?
2) Will the system scale to handle the anticipated load?
3) Will the system be available 99.999% of the time?

Since the J4U Next Generation application is just starting development, we investigated these questions by reviewing the design documents and interviewing the teams to understand how the system is going to be built. **Our overall assessment of the architecture is that with a few minor exceptions, the design is sufficient to fulfill the requirements. The majority of our recommendations will therefore focus primarily on immediately actionable recommendations to minimize risks that could potentially derail a successful implementation.**

5.1 Project Deadlines

Project timelines are made up of many interrelated dependencies and tasks. Projects require a careful, well-planned orchestration between resources, requirements, design, implementation and deliverables. Delays in one area can cause delays in several others. In our assessment we focused on three key areas related to project timelines:

1) Environments
2) Implementation
3) Communication/Coordination

In this section we will review and highlight areas which may potentially cause delays in meeting project deadlines, and suggestions to mitigate these risks.

5.1.1 Environments

An environment consists of all the hardware, software, networking, storage and external systems needed to operate J4U. J4U is executing a major port from an IBM technical stack to an Oracle stack, which is a significant undertaking. In addition, an entirely new network is being built to handle the anticipated load and to move toward an emerging industry standard, Pods. Significant changes in technology such as these are risky. In order to minimize the potential for impact to the timeline, one must be able to:

1. Quickly bring up, test, reconfigure, and tear down environments.
2. Capture configuration changes for replication into other environments.

3. Ensure consistent and dependable Development, QA, Staging, and Production environments.

Observations

Two areas that could potentially negatively impact the project timelines are Initial setup and environment replication. Due to the large number of new technologies involved, no existing environments could be cloned for the J4U core systems (WebLogic Application Server, and Oracle Database). This leads to long lead times of the initial setup and configuration of the environment.

Initial setup and configuration

During the interviews with the team, the potential for environments to miss their timeline was consistently mentioned as a huge risk to the project. The core issue is that the infrastructure team is not delivering to development team's expectations. In addition to this, the deliverables were perceived to be completed behind schedule. We discovered that these two issues were due to several challenges:

1. **Requirements timeliness and stability**
   The requirements for the environments were delivered piecemeal, changed frequently, and ultimately were not even finalized until after the environments were due. As a result, the infrastructure team was forced to scrap several setup/configurations through the build process.
2. **Communication**
   The expectations for the environments were not clearly set or communicated. The infrastructure team was using an iterative approach to bringing up environments, but this was not being clearly communicated to development. Development thought they were getting complete environments. The miscommunication resulted in additional lost time when development was waiting for a complete environment instead of using what was being delivered.
3. **Learning curve for new technologies**
   The project doesn't have any allowance for slippage of deadlines, which is not considered best practice because it is difficult to accurately estimate a task that has never been done before. Setting up environments can be a complex, interrelated web of configurations.

4. **Lack of documentation**
   There is little to no documentation on how to set up, configure, and run existing systems. This requires relearning of these systems, which takes time and leads to inconsistencies in the end product, leading to further confusion and inefficiency.

Cloning

In a virtualized environment, once a reference environment is set up, one can save the images of each of the servers (including the fully configured tools and middleware) in an image library. These images can be used to recreate clones of the original environment many times – for development, testing, staging, and production. Note that there will be some manual effort in terms of changing the IP addresses, URLs, domain relationships, and access rights configurations in the clones to separate the individual images / environments and avoid network conflicts.

For physical servers, though the same concept of saving images cannot be used, one can capture the server and OS configurations in scripts, copy the configuration files for the tools and middleware, capture the keystrokes during installation of various software elements, and document the entire process. This would be the most effective way of ensuring that the physical clone will be exactly the same as the reference server.

Once the systems are set up, they need to be cloned and replicated to other types of environments (Development, QA, Staging, and Production). We observed some concerns outlined below. Some of these concerns are similar to those raised previously with respect to initial setup and configuration.

1. **Lack of documentation**
   There is a lack of documentation on how to set up and configure existing systems. More complete documentation could have served as a baseline, reducing the time to set up new environments.
2. **Manual configuration**
   Configuration setup is currently performed as a manual process. It is very difficult to ensure consistency with manual processes. As a result, each environment type (Dev, QA, Stage, and Production) will have a slightly different configuration.
3. **Long Provisioning Time**
   There is a relatively long lead time to provision and configure systems. SLA takes approximately 30 days to set up and configure a server. Cloning a system should take hours to a few days (3-4 days).

Recommendations

1. **Use Automated Provisioning Tools**

Impact: High	**ROI:** Short-term

Use an automated system for provisioning systems (Build and configure). Between J4U and Remodel, over one hundred plus systems will be needed. Using automation would save a considerable amount of time, produce a stable, well understood system, and eliminate last minute troubleshooting of human errors.

VmWare has tools that will allow for saving images that can be deployed as "new" virtual servers in less than an hour. Pre work with regard to ensuring availability of SAN space and standard images plus post work with regard to connectivity and networks is still required. Note that the image can include middleware, application server software and tools preinstalled.

For hardware installations, tools from HP can facilitate the software side of provisioning. However, the racking, stacking and cabling tasks will remain. The network and connectivity tasks will also have to be executed.

2. **Use Configuration Management**

Impact: High	**ROI:** Short-term

Use Configuration Management to store deployment configuration and documents, knowledge base (Troubleshooting, Support issues, One-offs).

For every component in the environment, data regarding its provenance, age and configuration is stored in a tool (the Configuration Management Data Base). Scripts and agents can collect the configuration data from the target components and save them in the database. Additional scripts or manual intervention can ensure that the data is current, relevant and comprehensive.

3. **Use Existing Tools**

Impact: High	**ROI:** Short-term

Safeway has already purchased the necessary tools. Their usage needs to be encouraged.

4. **Document Systems and Processes**

Impact: High	**ROI:** Mid-term

Documentation is a critical part of any enterprise. Without good documentation people make mistakes, duplicate effort to solve known issues, and waste time looking for solutions. Safeway should hire technical writers and/or business analysts to capture:

1. Systems configurations
2. Work arounds
3. Task workflows
4. Systems processes
5. Known issues

It is import to develop an in-house culture of documentation as well. Without such a culture, documents will quickly become stale and out of date. In order to make documenting easier, more analysis needs to be done to understand Safeway's processes and to find a tool that best fits them.

5.1.2 Implementation

Implementation is the software development and systems integration of applications that run on application, database, and web servers/containers. The best designs can fail during implementation if appropriate precautions with checks and balances are not established. UST found several areas for improvement that will reduce to the project timeline. We also provide recommendations for addressing these issues.

Observations

OSSO

Open SSO (OSSO) is a key external system that handles user authentication and authorization. We identified three challenges with OSSO.

1. Originally the implementation of OSSO was supposed to take only a few weeks, but instead the rollout took almost nine months. OSSO was deployed to production in April 2011.
2. When the team originally brought up the OSSO environment for the J4U application, it was pointing to the development environment instead of to the staging environment. This error added to the Safeway's team perception that OSSO is unstable.
3. Maximum session problem. OSSO was exceeding the number of allowed sessions.

Through our interviews, OSSO was identified by team members as the largest single risk in the project. Our view is that further evaluation is needed. It is difficult to determine based on the

available information whether the current instability in OSSO is due to some systemic flaw, or due to pointing to the wrong environment (human error).

Development Methodology

The current development methodology is a modified waterfall approach with two development "tracks." Track one targets PD, Coupon Center SWY coupons, and Shopping list. Track two handles: CC MFG, PD, YCS, SL (Integrated), DM, and Mobile Preview. The tracks are being developed in parallel and are scheduled to last 25 and 40, days respectively. The tracks will be completely developed before they are handed off to QA for testing. Using this development cycle means that integration, performance, and scalability issues will not be discovered for approximately 30 days. If there are issues in any of these areas, significant delays to the project schedule could result.

As previously noted, several different types of risks exist that could potentially cause significant delays in the project timeline. The main areas of concern are:

1. **Environment**
   Due to the manual configuration issues raised in 5.1.1: Observations, it is highly likely that there will be configuration differences from environment to environment.
2. **Systems integration**
   Distributed systems rely on two key pieces to successfully exchange information. First, the public interface must be stable. The public interface defines how one system will talk to another. Public interfaces can change during development. Second, the functionality of the interface must be stable. Small, seemingly simple changes, can cause issues with interconnected systems, especially in enterprises with few automated unit tests.
3. **Performance**
   There are aggressive performance criteria the project must meet in order to be successful. Poor implementation choices can be difficult and costly to track down at the end of the development cycle.
4. **Scalability**
   Scalability and performance are closely related. So, the concerns are the same as the performance.

Recommendations

OSSO

1. **Use Stable Environments**

Impact: High	**ROI:** Immediate

Create and use an OSSO staging environment. Updates to staging need to be coordinated with all dependent projects, e.g., J4U. Development environments should not be used by external projects as changes and glitches are to be expected daily or weekly.

2. **Use Mock Objects**

Impact: Medium	**ROI:** Immediate

Mock objects simulate system components (database, external systems, etc.) so developers can write code and run tests without relying or configuring those system components. Mock objects keep development moving forward by isolating the component being developed. They are used with automated unit and limited integration tests. By leveraging a Dependency Injection framework, Spring, mock objects can be turned on and off with configuration files or command line arguments. Mock objects are a developer level construct that leverages the Spring frameworks' ability to inject behavior at run time.

3. **Hire Experts**

Impact: High	**ROI:** Immediate

Bring in expert resources for key technology stacks:

   a. OSSO (Resource started)
   b. WLS/MQ Messaging (Resource started)
   c. Datapower (In progress)

4. **OSSO Viability**

Impact: Low	**ROI:** Mid-term

Perform an in depth analysis on OSSO to determine its long term viability as part of the overall enterprise. It should be compared against other options such as Oracle's newer and supported SSO product.

Development Methodology

The sooner issues are discovered, the quicker and easier they are to fix, which reduces the risk of the project timeline slipping.

1. **Expand Unit Tests**

Impact: High	**ROI:** Immediate

Developers need to expand their unit testing to include: environment, integration, performance and scalability tests. Once the tests are written, they need to be run as often as possible. Below is a table of suggested run times.

Test Type	Frequency
Unit	Every code commit
Environment	Every deploy
Performance	Nightly
Scalability	Nightly

2. **Generate Health Checks**

Impact: Low	**ROI:** Immediate

In addition to the automated tests, each build should run several tools (like PMD, and Findbugs) to validate the health of the code. The health reports should be reviewed weekly.

3. **Use Simple Continuous Integration**

Impact: High	**ROI:** Immediate

The purpose of Continuous Integration (CI) is to mitigate risk. The core idea behind CI is frequent repeatable builds in a neutral environment, i.e., a location other than a developer's machine. It is important to compile and run the code in a neutral environment, not a developer's machine. Developers tend to have highly customized environments that are not well documented. CI tools rely on build automation scripts to actually compile the code and build the artifacts (deployable unit). Automation scripts are typically used to run unit tests as well. It is common practice to run unit tests every time the code is built. Notification alerts should be sent to the team on build or test failure.

In order to take advantage of CI, J4U needs to:

- **Use Maven for build automation**
  A CI server uses build automation scripts to compile the code and build the artifacts (deployable unit). Maven is a common build script tool for Java. Safeway is embracing Maven as a standard, but J4U is not using it yet**.**
- **Use Subversion**
  Maven does not support Serena's ChangeMan products. Maven has built-in support for Subversion. Subversion is a widely used Source Code Management

(SCM) tool. Code will be checked out each time a developer commits code to the SCM.

- **Use Jenkins or Hudson**

  Hudson and Jenkins, continuous integration tools, are currently being used in Safeway. Either tool will work for Safeway's needs.

CI Best practices:

- Compile the code and build artifacts every time code is committed to the SCM. Build code with every code check-in.
- Check-in code daily.
- Run unit tests with every build. See test frequency above.
- Notify (email or SMS) the team on build or test failure.

4. **Add Additional Services to build process**

Impact: Medium	**ROI:** Mid-term

In addition to the basic build process outlined above, additional tasks and services can be performed while building the code. Additional Services to perform:

- Run static code analyzers like PMD and Findbugs
- Configure the build to fail if health check reach a certain threshold
- Automatic release and deploy
- Generate release notes
- Generate developer documentation and post to wiki

5. **Perform Addition Analysis of Development Methodology**

Impact: Unknown	**ROI:** Mid-term

Safeway is moving towards the AFP-J framework. We did not have enough time to assess the framework and how well it will fit with J4U and Safeway's long term vision. Additionally, we did not analyze various SDLC related items like:

- Code versioning
- Code branching and merging
- Code check in process
- QA processes

5.1.3 Communication/Coordination

Ensuring that people have the right tools and information at the right time is a key element in project success. J4U Next Generation involves a number of different teams and consumes information from several enterprise applications.

Observations

The teams on J4U next generation consist of smart and dedicated associates who want to see the project succeed. They are heads down and working hard, however, they are not communicating as effectively as they could. The teams we interviewed believe that tight time lines are causing groups to focus only on their own issues, which leaves gaps and misunderstandings in the project, thereby costing time and increasing the risk of slippage. The teams also cited timelines as the reason that defined processes are being ignored.

The following is a list of examples of poor communications/coordination that has affected the project:

- Poor expectation setting and clarification caused teams to wait. See: section 5.1.1: Observations.
- The J4U environment was configured to point to a development instance of OSSO instead the more stable staging environment.
- Several groups are solving similar problems (e.g. OSSO), but not coordinating with each other to share and leverage knowledge.
- Major technology decisions were made late in the project lifecycle (JMS Messaging, Datapower, authorization checkpoints).

J4U Next Generation has many different groups that require a coordinated effort and cross functional collaboration to successfully complete the project. Although the tasks are defined at the program level, the work is being performed in a siloed manner. Each group completes its own portion of the task and hands it off to the next group in the work chain. Through our interviews and assessment it became apparent that there are issues with:

1) **Workflow**

   The task workflows are not clearly defined. Many well defined processes are not being followed and it is difficult to communicate ad-hoc processes that frequently change. Processes are not being followed due to the perception that there is insufficient time to do so.

2) **Roles and Responsibilities**

The roles and responsibilities for the various project tasks and processes are not clearly defined. Safeway recently underwent an organizational restructuring and many of the roles and responsibilities have gaps that are assumed to be filled by "someone else."

3) **Direction**

The teams' direction is not completely aligned. Different groups have competing priorities and expectations are not being clearly set. Resources are often and quickly moved from project to project.

4) **Working as one**

The team is not acting as a cohesive unit. The words "us" and "them" were often used when describing task coordination and work lists. The groups are using email as their primary task completion notification tool, in a one-way "fire it off and forget it" communication. This leads to a lack of follow-through in pursuing next steps and causes unnecessary delays in determining whether expectations for the project were met.

5) **Duplicate effort**

Different groups are often working in isolation on the same problems. This increases the risk of downstream issues, potentially leading to wasted effort, conflicts during integration, inconsistent environments, and missed expectations and timelines.

6) **"White Knight" syndrome**

Issues are regularly being solved through individual (small groups') heroic effort and not through collaboration or processes. As a result, these issues and their solutions are not being documented, perpetuating a culture of continually re-inventing the wheel.

7) **Decision Indecision**

Key stakeholders are not being included in design considerations that affect them. Support, development/QA and infrastructure groups expressed concern that decisions were being made without their input. As a result, when a decision is made and communicated to the project team, one key stakeholder or another raises valid concerns and the decision is put on hold or reversed. The issues are worked though, tweaks applied, and the decision is made again, only to have another group raise a new concern, and the decision is again put on hold. All of this back and forth leads to:

- Confusion among the teams,
- An erosion of confidence in decisions
- A loss of valuable project time.

While on site at Safeway we watched two important decisions go down this path: Datapower, and JMS/MQ messaging.

8) **Information Organization**

The information that is available for the J4U program is spread across multiple document repositories. These repositories have detailed information on the standards, guidelines, architecture and details on the implementation. However, the context of information, meaning, what information pertains to J4U and the currency of information, is not evident.

Recommendations

As previously noted, though the course of our assessment we did not discover any serious issues that would cause the project to fail. However, we found some areas that, if addressed, will reduce the likelihood of project delays.

1) **Create Program Project Plan**

Impact: Medium	**ROI:** Immediate

Create a program level project plan that shows all the dependencies and resources used. Currently it is difficult for one group to know if another is missing its milestones. For example, the development project plan has a single line item that is 112 days for environment delivery. There are no milestone points for development to know if the environments are on track until it is too late.

2) **Use Matrixed Resources**

Impact: Medium	**ROI:** Short-term

Create a matrixed organizational structure where resources are held accountable to their project outcomes. In a matrix structure the PM has authority over the assigned resources for the life of the project. The current non-matrixed structure tends to encourage the development of silos and a lack of coordination across the project teams. A matrixed structure would also help mitigate the "Us versus Them" type of thinking discussed previously.

3) **Use Project Portal**

Impact: Medium	**ROI:** Short-term

The team has a wiki set up but is not using it. The wiki needs to be used to foster better communication and improved project collaboration. Usage of the wiki can be encouraged through regular reviews of usage levels. Safeway should identify an internal champion to spearhead and oversee internal adoption. See section 6 below for recommendations on structure and content.

4) Create RACI Matrix

Impact: Medium	**ROI:** Immediate

A RACI matrix identifies who are Responsible, Accountable, Consulted, and Informed during tasks or processes. The issues we found centered on task hand-off between different teams. These processes need to be identified and documented. Once documented, create a RACI matrix assigning roles and responsibilities to each step in the process.

Code	Name	Project Sponsor	Business Analyst	Project Manager	Technical Architect	Applications Development
Stage A	Manage Sales					
Stage B	Assess Job					
Stage C	Initiate Project					
C04	Security Governance (draft)	C	C	A	I	I
C10	Functional Requirements	A	R	I	C	I
C11	Business Acceptance Criteria	A	R	I	C	I
Stage D	Design Solution					

Figure 7: Sample RACI Matrix

5) Coordinate Efforts

Impact: Medium	**ROI:** Immediate

There are two areas where coordination needs to be improved:

- Task hand-off

  Team silos are fostering a "throw it over the fence" mentality that wastes precious project time at each step in a task/process. The teams seem to behave as if their responsibility ends when they send an email stating the task is complete. From our interviews, there doesn't appear to be much cross team coordination, expectation setting or follow-up. As a result, the teams end up waiting for more information or are caught off-guard and not able to start critical tasks immediately. In addition to Recommendation 2: Use Matrixed resources above, the project managers need to drive the coordination across teams by proactively communicating with the various steps in the task. Also, Recommendation 4: RACI above will help the teams to understand what the expectations are.

- Solution to enterprise problems

  We observed several groups solving the same problems (e.g. Continuous Integration and Loyalty and J4U OSSO), but they were not coordinating their efforts. As a result, each group was solving the same issue in slightly different

ways. Part of the problem is that the different projects feel that they cannot rely or wait for another group to solve "their" issues. This could lead to integration issues later in the project cycle. Project managers and architects need to communicate and coordinate their efforts across projects.

6) **Build Information Contexts for J4U – Readme First**

Impact: Low	**ROI:** Long-term

There are baseline items that must be established across the programs to ensure there is a homogenous understanding of the J4U Next Generation Project. To build information contexts J4U needs to:

- **Define a Single Repository**
  Define the single source of all information pertaining to the J4U program. All key sub sources, versions must be available at the established locations.
- **Create help artifacts**
  Each directory, virtual directory must have a "Read Me First", containing, the subsystem name, a brief description, the version (if any), the artifacts in the folder , brief description of the artifact, associated references to collab, other ShareIT, DiscoverIT references and context of association (meaning, why did we reference Notification framework).
- **Assign ownership**
  Use a technical resource to aggregate the artifacts by interviewing and reviewing artifacts posted.

5.2 Scalability

Scalability is the ability of a system or group of systems to handle increased loads in a predictable manner. J4U has aggressive scaling requirements, which we believe the current architecture has been designed to handle. However, scalability can only be measured in the real world. So we focused our recommendations on how to ensure that the implementation meets the design. In our assessment we focused on three key areas related to scalability:

1. Implementation
2. Integration
3. Monitoring

In this section we will review and highlight areas which may potentially cause delays in meeting project deadlines and how to mitigate these risks.

5.2.1 Implementation

Implementation is the software development and systems integration of applications that run on application, database and web servers/containers. The best designs can fail during implementation if appropriate precautions with checks and balances are not established. UST found several areas for improvement that will help to ensue the application meets the scalability requirements. We also provide recommendations for addressing these issues.

Observations

Scalability of J4U Next Generation will rely on three factors:

- Architecture
- Implementation
- Scalability of dependent systems

Architecture

We focused our review on the J4U next generation components and did not evaluate the external systems that J4U relies on. In addition to reviewing the diagrams, we interviewed the team to determine the thought process and rationale behind the architecture decisions. **Our conclusion is that the architecture is designed to be scalable and that the architects are thinking and planning for scalability.** However, true scalability must be measured in the real world against the developed code.

Implementation

In J4U, these performance tests are to be carried out at the end of the development cycle. Waiting until the end of development means issues will not be discovered for approximately 30 days. Scalability issues often require rework or replacing core pieces of the application and sometimes require re-architecting pieces of the application. Any of these issues could result in significant delays to the project.

Scalability of dependent systems

J4U relies on a number of external systems to deliver its full functionality. Therefore, the scalability of J4U is dependent on these external systems. For a complete list see: Figure 4: High level Business Dependency Map

Recommendations

1. **Conduct Design Reviews**

Impact: Medium	**ROI:** Immediate

The architects should be responsible for the quality of the code developed and put in checks and processes to that end. Frequent design reviews ensure that the implementation adheres to the design. A design review should be conducted by the application architect and focus on:

- Determining and validating the differences between design and implementation.
- Interfacing with external systems or major points of demarcation within the application.
- Articulate the reasons behind the design to developers.

A typical review last an hour and involves the architect, the team lead, and the developers. Design reviews are not code reviews, but often go into the code to determine the severity of the problem. For the J4U project, these reviews should be weekly and focus on performance and algorithm efficiency. It will not be possible to review all architecture topics; but instead, select the critical components that are used the most. There are a wide variety of tools that can generate sequence diagrams, activity diagrams and object model from existing code. These artifacts can help narrow the review to points of concern. Another technique is to use health check and metrics to zoom in on likely trouble spots.

2. **Conduct Code reviews**

Impact: Medium	**ROI:** Immediate

In order to ensure that good coding practices are used, and to flush out issues as early as possible, code reviews should be conducted by the team lead and focus on:

- Enforcing good coding practices.
- Review of the health checks and static analysis tool results.
- Validate the day to day decisions that developers need to make in order to satisfy the requirements.
- Ensure unit test code coverage.

A typical review lasts an hour and is made up of the team lead and development team. It is important to review the code in a group setting and to encourage peer level comments and analysis. For the J4U project, these reviews should be weekly and focus on performance and algorithm efficiency. Use health check and static analysis tools like PMD to focus only on likely trouble spots.

3. **Create component scalability tests**

Impact: High	**ROI:** Immediate

To mitigate the risk of waiting until the end of the development cycle to test for scalability, scalability tests should be part of the daily tests that developers run see section: 5.1.2 for the recommended test schedule. In order to write unit level scalability tests, component level thresholds must be defined. Typically, as the application evolves, these thresholds are revisited. It is important to note that the thresholds need to take into account the environment on which the tests are performed. Environment specific text or property files can be used. The tests should be JUnit tests. The JUnit framework is already being used by development. The tests should fail if they don't meet the established thresholds. Continuous Integration should be used to execute the scalability automated tests.

5.2.2 Integration

As noted in Figure 4: High level Business Dependency Map J4U Next Generation relies on a number of external system for communicating with the stores, authentication and authorization, targeted offers, etc. Each of these systems will have an effect on the scalability of J4U. UST found several areas for improvement that will help to ensure the application meets the scalability requirements. We also provide recommendations for addressing these issues.

Observations

1. **New technologies**

   J4U is introducing many new technologies that will need to be integrated together. Each technology has a learning curve associated with it, which results in modications to integration techniques and strategies throughout the project lifecycle (e.g. Datapower and WLS/MQ Messaging). In addition, each of these technologies will need to be fully tested to ensure that Safeway understands the performance characteristics of each new piece. However, the current plan is to test for scalability at the beginning as an integrated POC and then again at the end of the project. As previously stated, waiting until the end to test introduces large risks. Also, it is unclear if the integration performance tweaks will need to wait until the final testing.

2. **Heavy Reliance on OSSO**

   We are concerned about the number of touch points requiring calls to OSSO. In the coupon flow, each server request from the client and each service call between servers require an extra call to OSSO to validate credentials. This centralized authentication/authorization technique is often called "Mother May I" due to the need to constantly ask a central authority for permission to perform a task. Although very secure, it places a tremendous burden on a single, albeit load balanced, system. We did not have time to review the security requirements for J4U Next Generation, but there are several techniques to reduce the number of calls to OSSO while maintaining the same level of security.

Recommendations

1. **Leverage JUnit for Scalability Tests**

Impact: High	**ROI:** Immediate

   Create integration tests that leverage the scalability test that were created in recommendation 3 above (Implementation:3). These tests should allow the infrastructure teams to quickly figure out the best configuration for the given scenario.

2. **Use Credential Caching**

Impact: High	**ROI:** Short-term

   The current architecture is going to cache LDAP data within OSSO to speed up validation/verification of credentials. We do not think this is aggressive enough. Although there are many techniques to reduce the number of calls to OSSO, a simple approach is to use credential caching. The cache would be used on the local system, avoiding a call to OSSO, and each credential would be valid for some amount of time. After that time has expired, the credentials must be revalidated from OSSO. Using a

credential cache will significantly reduce the number of calls to OSSO and speed up individual requests/transactions.

3. **Decouple System Dependencies**

Impact: Medium	**ROI:** Mid-term

Decouple process flow where possible using asynchronous processes or queues. Decoupling will allow different parts of the system to scale independently of each other and thereby increase throughput.

4. **Test for Failover and Error Scenarios**

Impact: High	**ROI:** Immediate

The currently scalability and integration testing is focused on happy day scenarios. A system is not scalable if it comes to a halt due to errors or exceptions. Error and exception handling must be able to scale to the same level as happy day scenarios. High availability/failover testing must also be incorporated into the plan. If the systems cannot meet the scalability requirements when key systems are off line (one server in balanced pair) then it doesn't scale.

5.2.3 Monitoring

Our understanding based on the review of the monitoring is that many of the Loyalty components will be monitored by the Safeway ESM team. There is a comprehensive plan that has been defined that lists the components in the Loyalty stack that are to be monitored. The ESM team has a 30-60-90 day plan to address J4U and Loyalty monitoring. The plan details out the roadmap and certain high level tasks and activities to enable monitoring. There are some key core items and non-core items that have been identified for the J4U re-launch, the J4U Next Generationas well as the types of resources needed.

There is a high level execution strategy for Nor Cal J4U – this highlights key areas that need to be identified but specific details in each area are not specified. There are the high risk areas as identified by the team. There are details on who would be the lead responsible for successful execution of the sub level activities and focus to complete details on what needs to be monitored.

The components identified specifically for J4U Next Generation are the New Loyalty Network, the Load Balancers, the J4U custom written components, the COTS components used in the J4U application including Autonomy MediaBin, Autonomy IDOL, Autonomy OpenDeploy, Autonomy LiveSite, DataPower, Cheetah Mail, Axway File Exchange servers, WebLogic Application Server,

Omniture, MQ to mention certain important components on the stack. Apart from these there are a number of components, frameworks and containers that are significant to J4U and Loyalty as a whole.

The monitoring components that are identified to monitor the application are the BMC SIM, SiteScope and the introspection components of Autonomy, DataPower

Observations

1. **No details on monitoring**

   The Loyalty application components appear to be monitored based on the detailed strategy and plan for Loyalty. However, the J4U Next Generation components are not document with the specific details that would provide what exact counters need to be monitored.

   From a network perspective, it appears that standard network parameters are being monitored, such as, throughput on key VLANs, core routers, firewall, and latencies between networks. From a general application perspective, the frameworks, the containers, COTS, server and software have parameters identified by the software vendor/provider that enables the tool to monitor the state. Though these are identified the right set of parameters to monitor for J4U has not been identified.
2. **Thresholds not established**

   There are no thresholds that are defined for the applications. The team has identified certain areas that the application components will be monitored by BMC, some areas by SiteScope and some areas by the different components introspection monitors. The aggregation of composite monitors is not defined.
3. **Unclear Rollout Plan**

   It appears that the NorCal loyalty strategy has been defined, but is unclear how that will be rolled out to the other divisions.
4. **Media Bin Monitoring**

   Autonomy MediaBin monitoring has been identified but the value of monitoring needs to be evaluated. Typically, MediaBin is used by the content producer in an interactive mode. If any faults occur, then the content producer will take corrective actions. If this is not the manner it is being used, then the use case must drive what is it that is required to be monitored.

Recommendations

1. **Detail Monitoring Specifications**

Impact: Medium	**ROI:** Near term

Monitoring of the components must be detailed. The detailed plan must include how the business will determine issues with the J4U when there are faults. A good monitoring plan also plans for remediation, defines corrective measures and determines how to differentiate between the false positives and the genuine faults.

2. **Establish Metrics and Thresholds**

Impact: Medium	**ROI:** Immediate

Metrics (aka. Counters) to determine application health for significant components on the J4U stack must be defined. Monitors must be identified for each metric where needed. Each parameter that provides state of the component on the J4U stack needs to have defined thresholds. These thresholds are calibrated to monitor for the specific watermarks to notify the appropriate teams to take corrective actions. The thresholds are important to determine as this is a factor of how quickly the support team could react to take corrective actions.

3. **Do Not Monitor MediaBin**

Impact: Low	**ROI:** Long term

Most uses of the MediaBin components appears to be during content production time, but if content production is automated (with no human intervention) then it will be required to be monitored.

4. **Define Scope of Monitoring**

Impact: Medium	**ROI:** Near term

The scope of the end to end monitoring for J4U is from the point of entry of external systems into the Safeway Network to the Loyalty; for internal systems it is the entry point from that process to J4U Next Generation. The software that pertains to Loyalty (specifically J4U) will be monitored along with the extensions and customizations that are developed by the J4U and Loyalty team. It must include the real time access of the J4U system via Web Services; the near real time access/interface and the batch interfaces.

Since there are a number of hardware components that will be used to run the J4U Next Generation application it is assumed that the Safeway ESM will provide the monitoring guidelines to the implementation team.

The Hardware and the Infrastructure Service Level Agreements (SLA) and Key Performance Indicators (KPI) will be managed by the Safeway ESM in accordance to J4U Non Functional Requirements as defined by the business teams.

There would be different SLAs for each of the consuming systems. Each of the consuming systems will provide expectations on the performance which will be converted to acceptable thresholds for the monitoring system.

In the event that acceptable thresholds cannot be established due to lack of an NFR specification then an acceptable threshold is defined and the monitoring is managed to that threshold.

5. **Define Alerts and Notification Systems**

Impact: Medium	**ROI:** Near term

The alerts and notifications will be sent via Safeway's email system and messages will be posted only for Safeway domain Ids. The alert and notification framework must route the message to the appropriate individual/group to the associated delivery device (email, pager, phone, terminal etc).

The details of the alert or notification can only be accessed via the centralized aggregation system and would require credentials and authorization to access. Access to individuals and departments will be managed by the business and network center.

6. **Define System Monitors**

Impact: Medium	**ROI:** Near term

In order to accomplish the goal of the end to end monitoring it is important to define the system where the monitoring is to be performed. As multiple systems will be integrating with J4U/Loyalty are both internal and external, it requires J4U to define the boundaries of monitoring control. The typical monitors with composite monitors that would be needed are:

- Ping Monitors
- Port Monitors
- Mail Monitors
- Log File Monitors
- File Monitors
- Directory Monitors
- Web Service Monitors

- Application Server Monitors
- J4U Clip Monitors – Composite Monitor
- J4U Transaction Monitors – Composite Monitor

It is important to follow a comprehensive structure that will help determine the counters that will be measured. A simple to understand structure is provided below that illustrates the type of detail to be covered.

Parameter	Value/Guidance
Server	<<name of host file>>
Counters	MemoryFree, MemoryTotal, MemoryUse, ActiveThreads, TotalThreads, PercentTimeMaxed, ConnectionWaitTime, ConnectionTime, ConnectionPercentUsed, NumTransactions, ActiveTransactions, TransactionRT, RolledBack, Timeouts, TransactionSuspended
Target	<<logical server>>
Update every	60 seconds
Title	J4U Application Server Trace
Timeout	5000 milliseconds
App Server Directory	<<directory of Application Server location>>
Verify Error	No
Update error (on error)	15 seconds
Schedule	7x24
Monitor Description	<<server name, if there is a cluster, or DR server name>>
Report Description	J4U Trace:<<servername>>
Depends on	None
Depends Condition	None
List order	None
Error if	<<depends on the counter selected>>
Warning If	<<depends on the counter selected as thresholds>>
Good If	<<depends on the counter selected and good>>

5.3 Availability

Availability is the measure of a system or group of systems to continue functioning over time without interruption. Availability is a combination of infrastructure and software. The systems must continuously perform to their required functions, i.e., it is not enough that the hardware is still running. The application must be correctly responding to user requests.

J4U has very aggressive availability requirements to be available 99.999% of the time. That allows the system to be down 5.26 minutes per year. We do not believe the project will achieve this level of availability; see below for specific reasons and recommendations to minimize their impacts. In our assessment we focused on two key areas related to availability:

1. Infrastructure
2. Application

5.3.1 Infrastructure: 99.999% Availability

This sections looks at availability from the perspective of hardware. It includes information on systems provision, environment stability, availability consideration and costs, and how to achieve 99.999% availability.

Observations

1. **Long Lead Time in Building Environments**

   The application development (APPDEV) team has stated that it takes too long for the infrastructure (INFRA) team to create an environment for them to work in (development or test environment). The reasons are several – new technologies, requirements received over time, limited resources; and some of it could also be perception related, as the infrastructure team releases the environment in stages, instead of a single release.

   One example that was proffered was that it took a month to obtain the WebLogic Server (WLS) environment. The explanation is that WLS with the default configuration can be generated in a day – but the WLS with the configuration that APPDEV needs can be done in a few days, perhaps a week, if the requirements are known and fixed. However, even this configuration will need to be tweaked and tuned as the application development activity progresses. The requirements are not fixed because since this is a

technology product that is new to the organization, there is a learning period that both APPDEV and INFRA teams are going through.

In general, APPDEV states that it takes 4 to 6 weeks to obtain a physical server and up to 2 weeks to obtain a virtual server. If this is related to just the server + OS per se, then these durations should be compared to such activities at other large IT organizations. Four to six weeks from ordering to racking is normal for a physical server; even 3 days for a virtual server in an existing virtualized environment would be considered excessive.

2. **Environment Stability**

   Stability of the environment has also been questioned – especially the OSSO (Open Source Single Sign On) component. The suggestion is that the APPDEV team has been linking to the development (DEV) instance of OSSO and not the more stable QA (Quality Assurance) instance.

   Perusal of the capacity planning documents reveal that the inputs for the model have been increased by a factor (by 300%) and that has resulted in a conservative sizing that is more than adequate for the National rollout. This allows for Scalability and also affects performance.

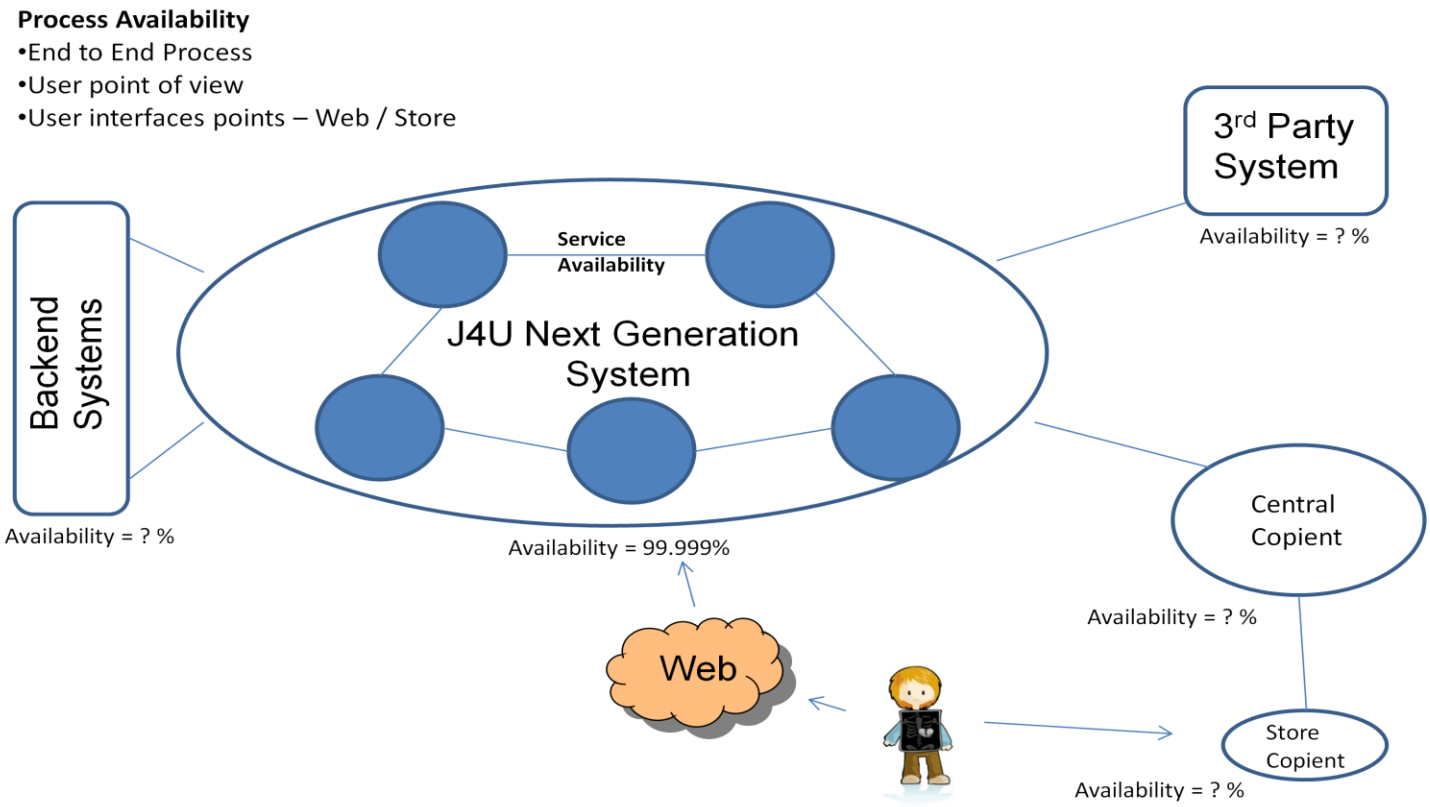
3. **Five-Nines Availability**

   Safeway has stated that it would like to have five-nines availability. However, neither the definition of availability (system, process, services, infrastructure), nor the measurement of five-nines (average, probability) is consistently expressed by the team. It is also very important to consider both the system not being available for a certain amount of time and the time period during which it was unavailable. For example, unavailability at 1 am, when the store is closed, may not be critical, but even a 1 sec unavailability of the system at 10 am during the Thanksgiving break could have an impact on the business.

   From a business customer perspective, five-nines availability would mean that on a given day, the worst that can happen to their interaction is a delay of 0.88 seconds (or a loss of data for 0.88 seconds) – which in practical terms means that there is no perceptible disruption to the customer at all.

   In the diagram below, the customer can interact with the Loyalty system through the web or directly at the store (and in the future both simultaneously using the mobile application). The J4U Next Gen system could provide 99.999% availability, but the other

systems (since they are legacy or 3rd party systems) may not be able to offer 99.999% availability and so the entire process has less than 99.999 % availability.

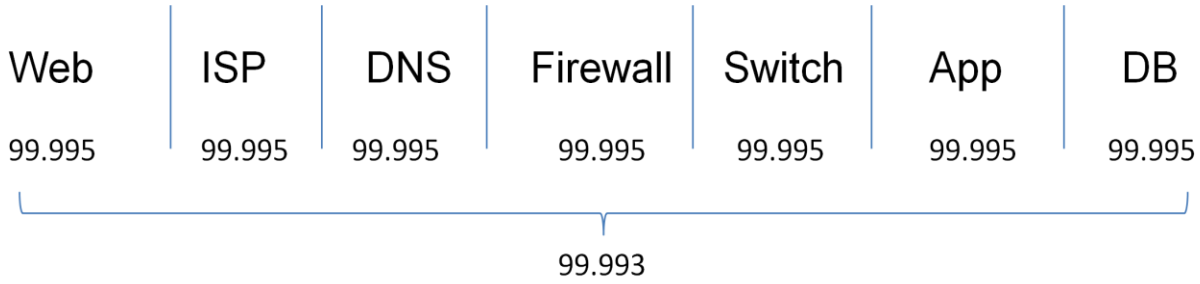


One can also define five nines in terms of the business process being disrupted at the most for only 0.88 seconds in a day. This is very similar to the business customer's perspective.

From an infrastructure point of view, if all the components necessary to access the application are available (with an exception of 0.88 seconds for a given day), then five nines availability has been achieved.

4. **Availability Calculation**

   The measurement or calculation of five nines affects the end result. The probabilistic view states that if a system has multiple components and each component has a probability of being available for 99.999% of the time, then the availability of the system is a product of the individual probabilities and mathematically this product will be less than 99.999%. This means that all components of the system have to be available 100% of the time, except for one component. The chain is only as strong as its weakest link.



The other mathematical approach to availability is to calculate availability (uptime and downtime statistics):

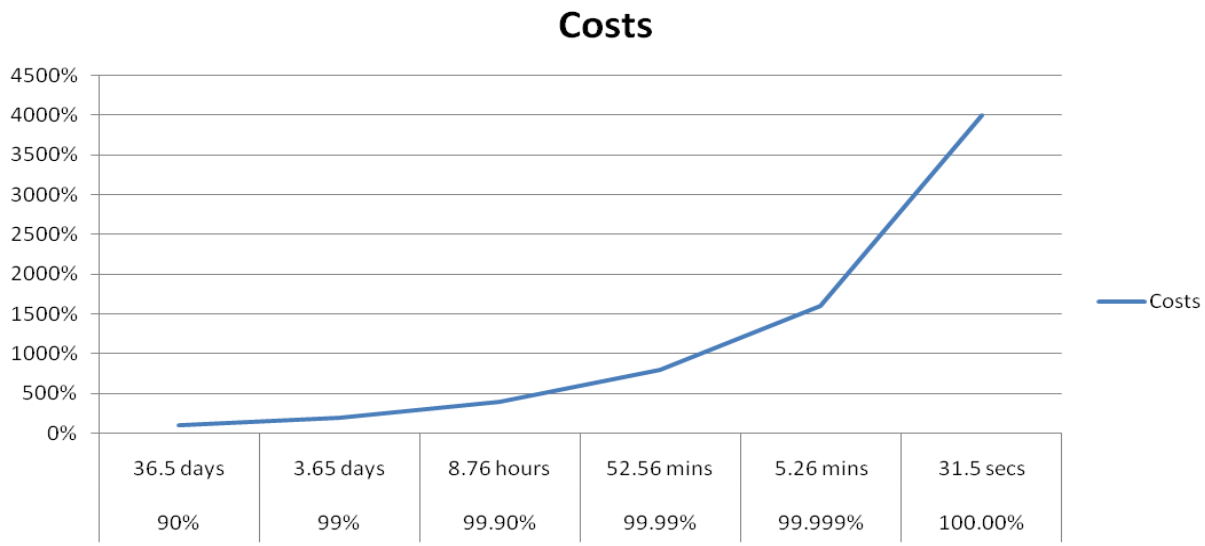
Availability % = 100* ((Uptime)/(Uptime+Downtime))

Use this calculated availability and average these values over a period of time. Longer the period of time, more closely the average will tend towards the stable value.

5. **Five-Nines Costs**

   The infrastructure team states that it is building the data center "capable" of five nines.

   Five nines availability is expensive, as indicated by the diagram below:



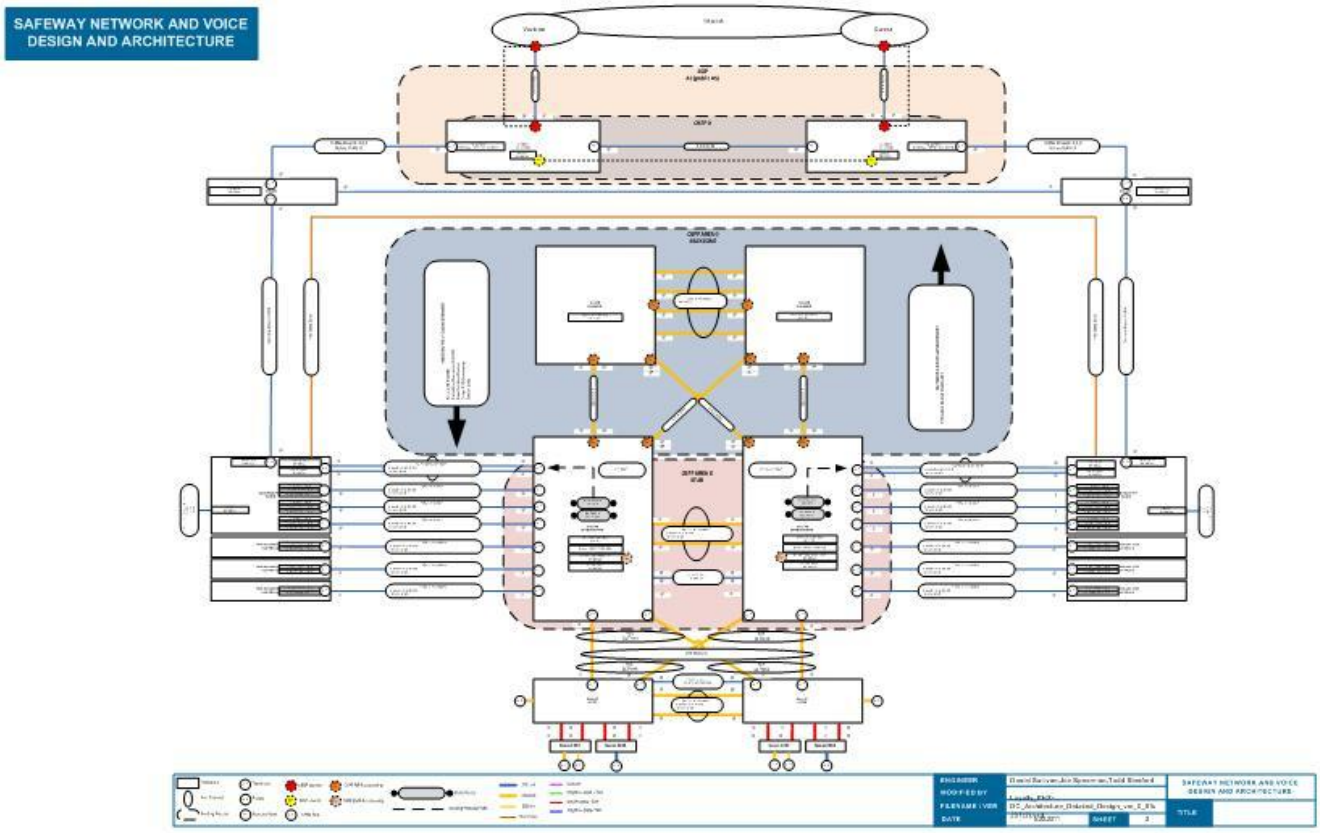
Note that five-nines would mean that the system can suffer downtime of only 5.26 minutes per year – which would be 0.88 seconds per day!! and would cost at least 20 times that of achieving 90% availability.

6. **Achieving Five-Nines**

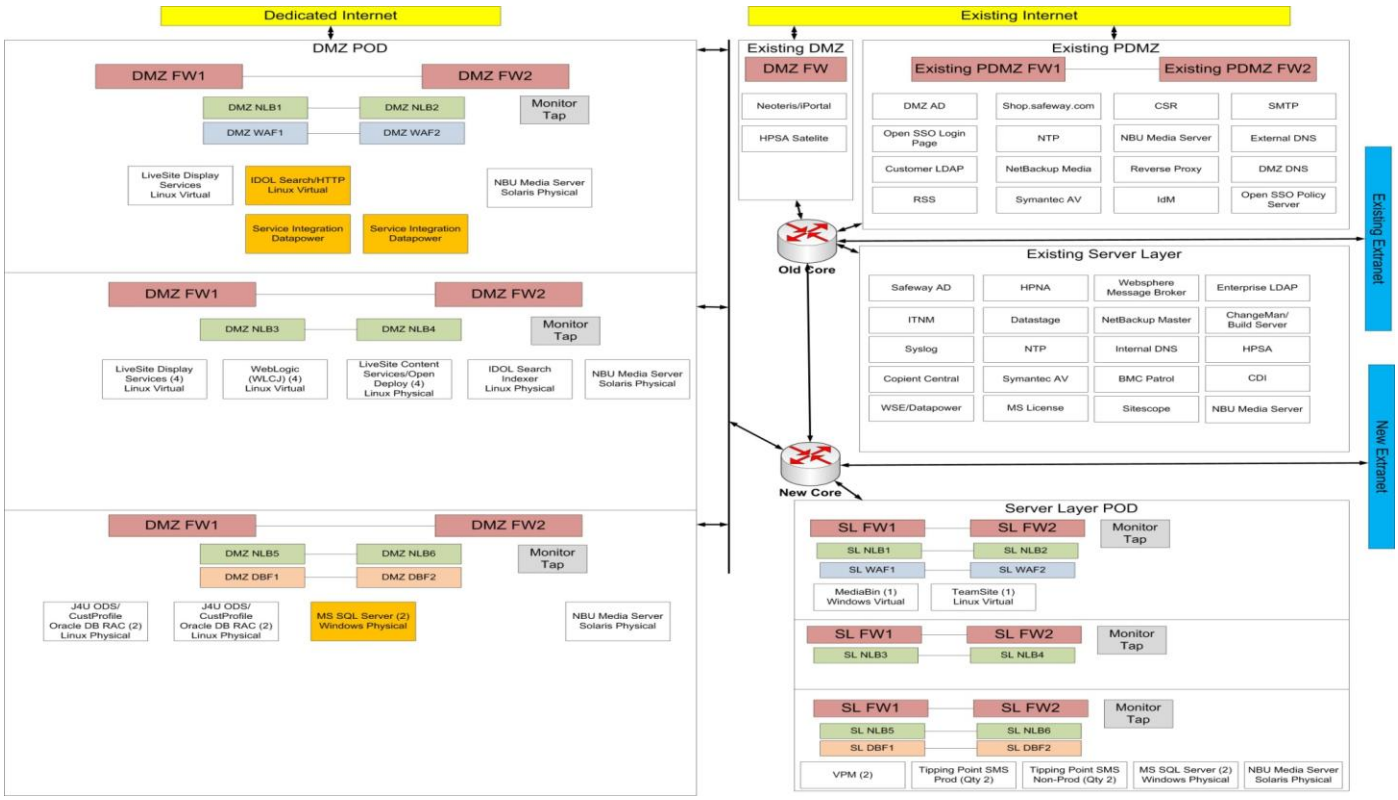
   Five nines is achieved through redundancy. The entire system is built for peak capacity and then it is replicated in the same data center to provide complete redundancy. (This is not to be mistaken with disaster recovery, which involves replication of the entire system in a different data center located in a distant geography). Note that redundancy

means more than just having duplicate hardware/equipment. It means interconnecting redundant components such that the failover from one to the other is automatic (no manual intervention) and is seamless.

The network architecture has been designed to be redundant as seen in one of the architecture diagrams:

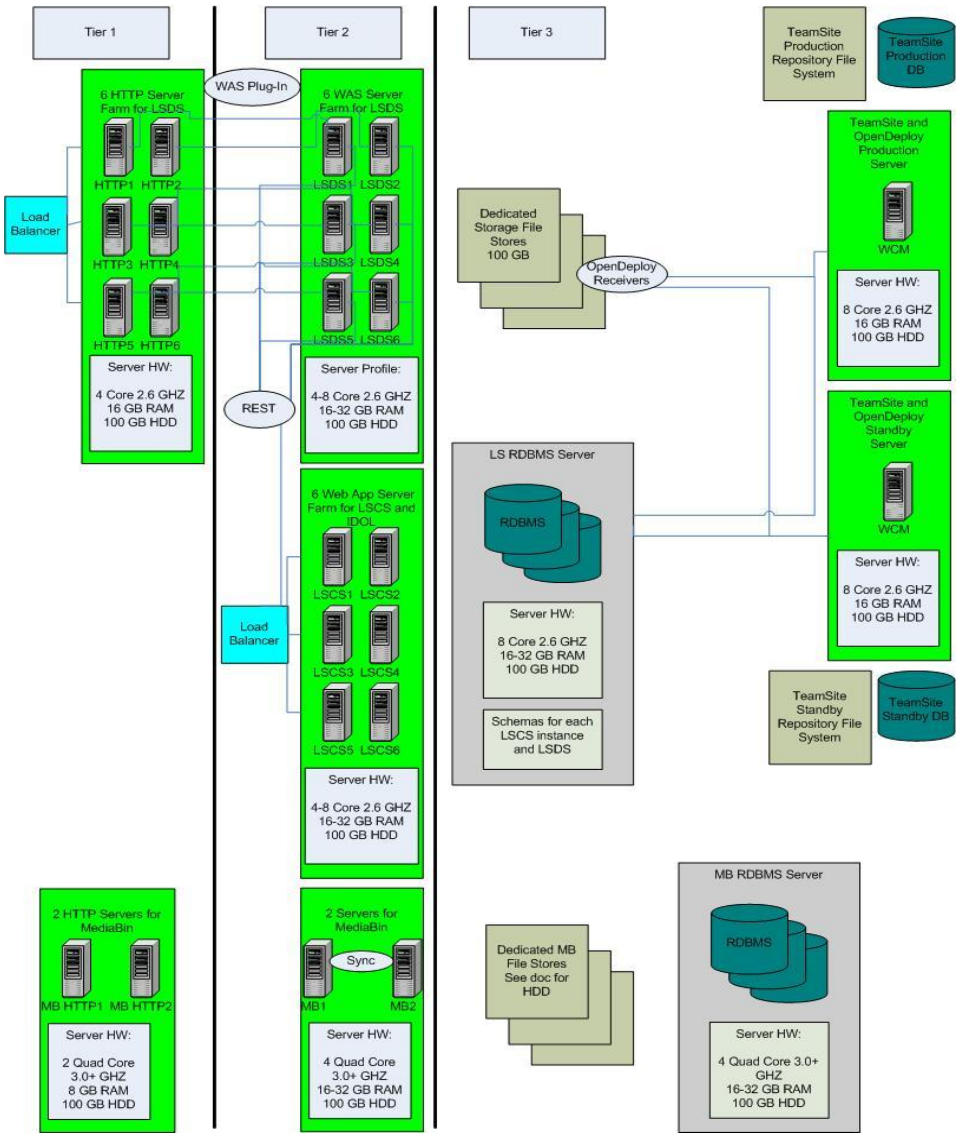


This same level of redundancy is also depicted in the diagram below which also emphasizes the multiplicity of firewalls, the most important of which provide security while the others provide isolation.



J4 U deployment diagram

The diagram below shows the use of server clustering to ensure redundancy. The database servers are physical servers and though this diagram does not depict database clustering and replication, in interviews, it was clearly stated that such failover/redundancy has been defined in the detailed architecture diagrams.



Safeway Loyalty Architecture (draft)

7. **Configuration Management Tools**

   Safeway possesses the HP Suite which has components that will capture Configuration Management (CM) data using agents and store it in the Configuration Management Database (CMDB). Safeway does not leverage these tools.

8. **Infrastructure Team Organization**

   Safeway is changing the way the Infrastructure team is organized with a view to ensuring that the sufficient care and attention is given to the collection of accurate requirements and delivering to those as per desired schedules. However, there is a gap that needs to be addressed. The infrastructure team is also responsible for installing and configuring middleware and application server components. This is because such

configuration sometimes needs administrative privileges on the server where the software is to be configured. But the configuration needs inputs from the application development teams. There could be a standard configuration for Safeway (such a standard is yet to be defined), but there will still be tweaks on a per application project basis. Additional tuning and tweaking will be required as the project progresses and either some requirements change or one learns more about the software (or there is a new release of the software).

9. **Disaster Recovery**

   Disaster Recovery (DR) is very important. Safeway has decided to allocate resources to develop a DR plan, but this plan and its execution will not happen before Just4U goes live. Safeway seems to understand and accepts the risks of this approach.

10. **Technology Selections**

    Safeway has made technology selections with regard to individual components of the infrastructure. These selections have been made after sufficient consideration and prima facie seem to be appropriate – based on Safeway's prior experience, capacity planning and input from the software/hardware vendors.

Recommendations

1. **Freeze and Deliver Approach**

Impact: High	**Timeframe:** Immediate

   Follow the iterative approach for delivering environments to the APPDEV team with a slight variation. Freeze requirements for the environment every day, at the end of the day. The updated environment should be made available to the APPDEV team by noon the next day for all tweaks that are immediately doable. Long term tweaks (Changes to the environment that would take more than 8 hours) should be documented separately and scheduled with INFRA and APPDEV teams.

2. **Organize Structure and Environment "spin-up"**

Impact: High	**Timeframe:** Short-term

   There are 3 options for shaking the organization up to reduce the concerns around "environment spin up" time.

   - Give the APPDEV team the privileges to change the system configurations (including middleware, application servers and tools) in the development environment. After the environment stabilizes, freeze and save the image,

Replicate this environment using the CMDB and, development team notes. If everything works, store image in image library. Use a similar process for physical servers.

- Make infrastructure resources available to the Loyalty team on an "On Call, drop everything else" mode till the environment stabilizes.
- (Recommended Option) Create a separate "Tools" team that will bridge the APPDEV and INFRA teams. This team will focus on configuring the middleware, the application servers, and the tools (development and testing). This team can report into the Program Manager and will be a short duration team created for each large project. This team's responsibility is to ensure that APPDEV has a stable working environment – DEV/TEST/QA/PROD as the application development process moves through its cycle.

3. **Document Requirements**

Impact: High	**Timeframe:** Short-term

Document all infrastructure requirements and make this a living document until Go Live. Then this document is frozen and edits to this will be made in conjunction with the change management process.

4. **Leverage Existing Tools**

Impact: High	**Timeframe:** Medium-term

Set up and use HP tools (that Safeway already possesses) for capturing configuration data auto-magically. The tool can also populate the Configuration Management Database (CMDB). Any manual changes to the data in the CMDB should come only through the Change Management process.

5. **Create Disaster Recovery Plan**

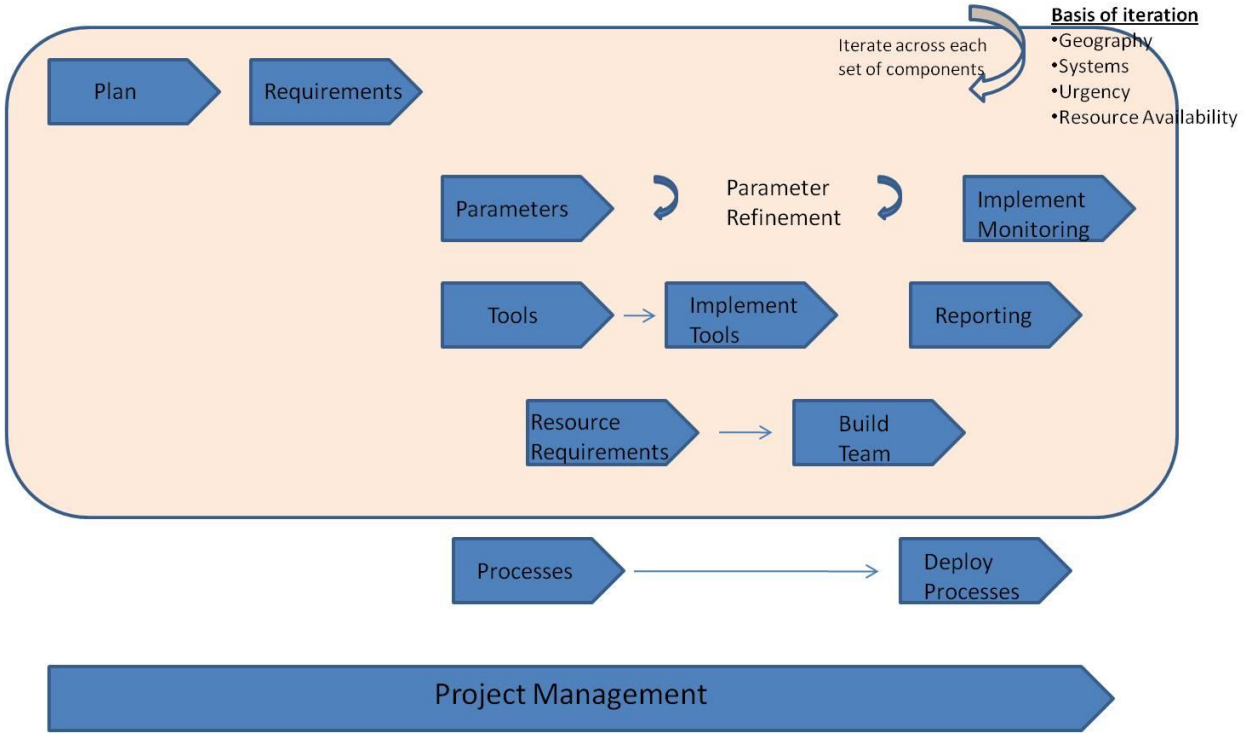
Impact: High	**Timeframe:** Medium-term

Develop and implement a comprehensive Disaster Recovery (DR) Plan. Start with determining the criticality of the business processes and the applications associated with them. For each of these, define the service levels required and along with that the Recovery Time and Recovery Point objectives (RTO and RPO). Based on these, one can plan the scope, size, connectivity , bandwidth, redundancy and operational aspects of the DR center.

6. **Establish Enterprise Monitoring**

Impact: High	**Timeframe:** Medium-term

Install and utilize enterprise monitoring tools. UST can help develop a long term plan (over the next 3 years) to implement an enterprise wide monitoring system. A sample approach is as below:



7. **Create Service Management Processes**

Impact: High	**Timeframe:** Long-term

Implement ITIL (Information Technology Infrastructure Library) standard Service Management processes for Support and Maintenance. ITIL is a Framework that allows the organization to institute robust processes tuned to specific needs. This is a long term project that involves changing the way the people work. It calls for a sustained effort over multiple years, but the value delivered will be visible within six months of the start of the implementation.

8. **Implement Full Redundancy**

Impact: High	**Timeframe:** Long-term

Implement full redundancy. For 99.999% availability, each of the components needs to have availability greater than 99.999%. This is possible only if full redundancy with automated switchover is implemented.

So, implement a system that caters to peak load. Now replicate this system and interconnect both the systems – original and the replica, so that failover is automatic (no manual intervention). Note that one would execute the replication and interconnection component by component.

For Just4U, sufficient redundancy has been built into the infrastructure. However, unless the other systems Just4U interfaces with also have similar redundancy, the overall availability will be less than what is desired.

9. **Establish Availability Reporting**

Impact: High	**Timeframe:** Long-term

The formula for calculating availability is well known. It is important to define availability reporting pertaining to the relevant process/component/application/system. It is recommended that the availability calculation

- Is based it on yearly average
- Is only calculated for relevant time periods (e.g.. 6 am through midnight)
- Excludes planned periods of unavailability

5.3.2 Application: 99.999% Availability

An application is made up of various software pieces that integrate together to perform a required set of functions. Application availability is difficult to measure and assess. **UST found several areas of concern that we believe will prevent J4U Next Generation from reach its availability goals.** We also provide recommendations for addressing these concerns.

Observations

There are three concerns that we have regarding availability with J4U Next Generation:

1. **Only planning for happy path**

   During our interviews, all the teams mentioned that planning for error and exception handling had not started. Without extensive error/exception handling paths and logic it will be impossible to reach 99.999% availability.

2. **OSSO Issues**

   Currently if OSSO fails, even in a load balanced scenario, all the systems that had sessions open will need to be rebooted. If this occurs, this catastrophic cascade failure will push the application beyond the 5.26 minutes per year.

3. **Legacy Systems**

Existing legacy systems that J4U depends on are not 99.999% available.

Recommendations

1. **Document all Error and Exception Paths**

Impact: High	**ROI:** Short-term

Availability is worst case scenario planning ("Begin with the end in mind"). Without error/exception case handling, it will be impossible to reach the desired availability. The errors and exceptions need to be documented for each hardware and software system. Each path must have a graceful and recoverable solution.

2. **Randomly Test System Outages**

Impact: High	**ROI:** Immediate

Document the failover scenarios, but remember to test random outages as well. One easy way to do this it to walk through the data center and random unplug and power down systems.

3. **Leverage Scalability Tests**

Impact: High	**ROI:** Immediate

The scalability tests above need to run in conjunction with availability testing. Leveraging the scalability test recommended above will reduce and possible eliminate writing availability test. See Scalability Recommendations above.

4. **Create Continuation Strategies**

Impact: High	**ROI:** Short-term

Create and implement strategies for complete tier outages. For example, the application cannot reach the database:

- How does the application handle that in a graceful way?
- How does it continue once the system is restored?

Strategies need to be created for Copient, database, OSSO, LDAP and network outages.

5. **Run Load Balancers at 45% Capacity**

Impact: Medium	**ROI:** Immediate

The industry standard for server load is usually 75% of peak capacity. However, in that configuration if one server in a load balanced pair fails during peak traffic, the remaining server needs to handle 150% of capacity, which is not possible. As a result, performance and availability will suffer as the system is not able to handle the load and starts timing out requests.

6. **Create an Emergency Demand Capacity Environment**

Impact: Medium	**ROI:** Mid-term

An emergency demand capacity environment would be used if the transaction volumes are higher than current capacity plans or worst case scenario disasters. We recommend creating a virtualized environment to handle an emergency demands. Cloud computing would be an easy way to achieve the additional capacity.

5.4 Additional Observations/Recommendations

5.4.1 Services

Observations

Due to time constraints, we were limited in our ability to analyze the service and their interaction with the enterprise and the domain.

1. **Not following REST naming convention**

   The J4U REST services are not following the standard REST naming or usage conventions. J4U is treating REST services like heavy weight web services. A shift in how REST services are perceived is required to take full advantage of their abilities. REST can be thought of as a coordinate system for resources. Each resource is pointed to by a unique URL. Also, REST services are discussed in different terms than other types of services. REST defines interactions in terms of nouns and verbs. Nouns represent the resource you need, e.g., coupon. Verbs are actions that are performed on the Nouns. REST has a fixed vocabulary of verbs (GET, PUT, POST, and DELETE) that correspond to web request types of the same name.

2. **Safeway is just starting their SOA initiative**

   Due to other priorities, UST was not able to look into Safeway's SOA initiative. Below are some recommendations regarding SOA best practices for your consideration as the organization moves forward with its SOA initiative.

Recommendations

1. **Use Lightweight Services**

Impact: Medium	**ROI:** Mid-term

   Replace heavy weight services Web Services with REST or HTTPinvocation services.

2. **Use Domain Models**

Impact: Medium	**ROI:** Long-term

   In Service Oriented Architectures there is often a tradeoff between service granularity and performance. Single purpose, finer grained services allow you to create composite services by aggregating fine grained services into a complex service. However, too many fine grained services results in a large network overhead that degrades performance. Ideally, Safeway will have a blend of high performing coarse grained services and highly reusable fine grained services. Use domain models to quickly bring up new services while maximize re-use. Domain models with Spring allow for the creation of new services through configuration file changes, i.e. no new code needs to be written.

5.4.2 EA

Observations

Through our interviews with the architects, there was a good sense of direction and what needed to be done. They were clearly thinking several steps ahead, but there was no plan or roadmap they were operating against.

Recommendations

1. **Create an Enterprise Architecture Roadmap**

Impact: Medium	**ROI:** Long-term

Establish tactical and strategic enterprise architecture roadmaps. The strategic roadmap should look out over the next 3 years and contain the major initiatives. A tactical roadmap should present a 3 to 6 months view and break down the major initiatives into smaller projects, see: 2 below.

2. **Create Enterprise Architecture Projects**

Impact: Medium	**ROI:** Short-term

Break up the EA roadmaps above into discrete projects lasting one to three months. It is very important that these projects are treated and funded like other projects. To be effective EA projects must:

- Be able to articulate to business and technical stakeholders the value of the project.
- Be run as any other projects with a PM, dedicated resources and a budget.
- Show real world value. Each project needs to establish real world metrics for ROI calculations. These calculations should be reported back to key stakeholders.

3. **Perform Additional Analysis**

Impact: Unknown	**ROI:** Short-term

We had limited time to work with the architecture group. More analysis is required for further recommendations.

5.4.3 Continuous Process Improvement

Observations

As stated previously, Safeway's processes are not well understood or are not consistently followed. Also, Safeway tends to wait until a process is bulletproof before proceeding with it, which causes unnecessarily delays.

Recommendations

4. **Establish a Process Framework**

Impact: High	**ROI:** Long-term

Establish a process framework like ITIL. ITIL has good metrics, and focuses on repeatable processes and incremental improvement. Appoint an ITIL champion with influence and power across the process and organizational lines.

5.4.4 Database

Observations

Due to time constraints, we were unable to look at the logical and physical data models or analyze enterprise data needs. However, the following issues came up as part of our other interviews and analysis.

1. Disconnect between strategic and tactical direction
2. No one is responsible for data policies or enterprise standards
3. No systems of record
4. Mixing of transactional and operational data
5. Maintenance of data is an afterthought

Recommendations

1. **Perform an in-depth analysis of Data Practices**

Impact: Unknown	**ROI:** Short-term

A comprehensive analysis needs to be performed on the data models, policies and procedures. The analysis should include:

   a. Deep dive on data models by non-biased players
   b. Validate data needs will be met near and long term
   c. Data quality practices and impact to organization.

2. **Establish Data Tzar Role**

Impact: Unknown	**ROI:** Long-term

Create a new position of data tzar (typically this position is Director of Information Architecture) The data tzar will have influence and power across organizational lines to ensure:

- Correct and consistent use of data
- Creation and enforcement of enterprise data and quality standards
- Creation and enforcement of data retention policies