Systems Engineering Management Plan

System Engineering Methods: CS672

Aidan Polivka

June 2nd, 2024

Executive Summary

This document is a proposal for a software engineering method plan for a bookstore called “Heartland Escapes”. It explores the bookstore’s technological need and provides a software solution to fulfill the bookstore’s high-level requirements. With these needs and requirements defined, this proposal then provides a roadmap of major project milestones and explains the project staffing structure.

After the staffing structure is identified, we then identify major decision-making processes and which individuals are charged with making those decisions. Then we outline the process environment and move into the system engineering model.

The system engineering model is broken into two parts, the conceptual model (Waterfall) and the iterative process model (Agile). Using Waterfall for the conceptual model allows us to get clear functional requirements for the entire system and build a road map and function/component matrix. It will also allow us to create a cost analysis for the customer and give the customer a good sense of what the product is going to be. This should also help us build trust with Heartland Escapes. Using Agile for our iterative development process should give us flexibility to pivot if requirements change.

The functional analysis section gives us insight into using functional requirements to better understand the product we’re developing for Heartland Escapes. This section took a lot of inspiration from the Journal of Emerging Technologies and Innovative Research. We then delve into the development process, and how we will go about implementing the functional requirements and gather the non-functional requirements. Then we discuss the validation and verification processes.

Then we discuss specialty engineering roles like UX designers and acceptance testing, then the system deployment process for Heartland Escapes. Finally, we discuss the maintenance team handoff after project completion.

Document History

Date Modified	Modification Details
May 5, 2024	Creation and outlining of initial release
May 12, 2024	Introduction: Purpose, System Overview & Project Schedule
May 19, 2024	Systems Engineering Model
May 26, 2024	Specialty Engineering, System Deployment, & Product Support
June 2, 2024	Executive Summary

Table of Contents

Introduction

Purpose

Heartland Escapes is a (fictional) Nebraska based family-owned bookstore that has been in business for a very long time. They have been set in their ways as far as store processes go and are far behind the curve for integrating technology into their business. As a result, the store owners have put many hours into paper-based inventory keeping, product ordering, employee timecard management and accounting. Frankly, the owners would be happy to continue handling all this work manually, but they are finding that with age they want to spend less time in the store. With how much knowledge is required to handle these processes, it will cost them a lot of money to hire additional hands to continue to support their paperwork.

System Overview

The system we develop for Heartland Escapes will be an administrative webapp. This application will be a single service to handle all their needs. It will support inventory management, employee time management, sales, accounting, and product re-ordering. This application will act as an e-commerce site for general users, and we will use role-based access permissions to support administrative functionality like order tracking, sales reporting, manufacturer information and inventory management. This new system will also support an employee role that allows employees to punch in and out using a custom time management system. This system will also support connections to cash registers for administrative and employee roles that are on the same network for an integrated point of sale system.

Project Schedule

Here is a list of key milestones we plan to accomplish throughout the course of this project:

1. Project directors & architecture staff introduction to major stakeholders
2. Onboarding of senior development staff
3. Client driven event storming sessions
4. High-level development roadmap
5. Solution Architect proposal for major technologies
6. Project management proposal for story development, bug remediation process, and technical debt management
7. DevOps Architect development and rollout of infrastructure as code
8. Solution Architect development and rollout of shell applications
9. Solution Architect rollout of project software development guidelines
10. Rollout of Development environment & CI/CD Pipeline
11. Rollout of Staging environment & CI/CD Pipeline
12. Onboarding of large-scale development effort (engineers, UI/UX designers, data engineers, scrum masters, acceptance testers, etc.)
13. Major Feature rollouts (preliminary ideas before client driven roadmap)
    a. User roles, authorization & authentication
    b. Inventory system
    c. Point of Sale system
    d. Product re-ordering system
    e. Employee time management
    f. Reporting system
14. Rollout of Production environment & CI/CD Pipeline
15. Deployment of in-store system to production environment, implementation of new system to Heartland Escapes
16. Rollout of e-commerce Development environment & CI/CD pipeline
17. Rollout of e-commerce Staging environment & CI/CD pipeline
18. E-commerce system development
19. E-commerce system production rollout
20. Establishment of maintenance plan with stakeholders
21. System handoff to maintenance team

System Engineering Processes

Project Organization

Because our organization is fully remote, we'll expect all engineering staff to work from home. With this in mind, we'll have to make additional network security considerations.

With the scale of this project, it would be best to have a hierarchical technical structure. We plan on having a solution architect, who oversees the technical leaders in data, dev ops, and two application development tech leaders. Then we'll have engineering staff ranging from entry level to senior engineers. The project management team will also be hierarchical, with one project director, two product owners, and four scrum masters. Finally, we will have a specialty team for user experience and accessibility.

The entirety of the project management team, the UX and accessibility individuals, and the solution architect will all be in frequent contact with the client. Technical leaders and engineering staff will have very limited contact with the client to reduce scope creep and distractions from development velocity. It's expected that the product owners and scrum masters will get enough information about the domain and expected feature requirements that the engineers should not need to have a direct relationship to Heartland Escapes.

Decision-Making Process

Final enterprise level technical decisions will be made by the client with the support of the solution architect and project director. Software architecture and technology decisions that don't have enterprise level impact will be made by the solution architect. Low-level day-to-day technical decisions will be made by tech leads.

Technical Decisions

Any time a third-party software or non-standard framework library is desired, an architecture design review (ADR) must be written to display options and pitch the desired library as the preferred option. This ADR must be approved by the solution architect to be implemented, but the ADR can be written by any engineer. If the ADR has enterprise level impact, it must be approved by both the solution architect and the client.

This process ensures that sufficient research is conducted before subscribing to a new technology. It also offers documented reasons why technological decisions were made during the process of developing the product for the maintenance team. Finally, it inhibits engineers from making unilateral architectural decisions that may be outside of the best interest of the client.

Project Decisions

Project level management decisions will be made by the project director. This individual will be responsible for final decisions on onboarding new engineers, scrum process, and what scrum ceremony will be conducted throughout the course of the project. They will also be responsible for determining sprint lengths, dictating required metrics to capture on a sprint-by-sprint basis, and development of process for capturing and displaying those metrics.

The project director and product owners will be responsible for event storming sessions with the client. It may be desired that the solution architect be a part of those discussions as well. These individuals will be expected to fully understand the clients request for feature development and the domain need that this work satisfies. As a result of these sessions, these individuals should be able to stand in for the client in project backlog population meetings with the scrum masters. When the backlog is refined, a session between the scrum leaders and tech leads will determine the level of effort of each item in the back log. This level of effort will be used in the final decision for quarter goals, and sprint commitments. This decision will be made by product owners and the project director.

Other Decisions

Once the solution architect, technical leaders, and project management team are onboarded to the project, decision-making processes will need to be defined:

1. Risk Management Decisions: Identifying, assessing, and mitigating project risks, involving the solution architect, project director, tech leads, and product owners.
2. Quality Assurance Decisions: Establishing quality standards and testing protocols, handled by the QA lead, tech leads, and project director.

3. Change Management Decisions: Handling changes in project scope or requirements, involving the project director, product owners, client, and solution architect.
4. Resource Allocation Decisions: Managing resource allocation and budget, involving the project director, client, and product owners.
5. Continuous Improvement Decisions: Conducting retrospectives and implementing improvements, involving the project director, tech leads, scrum masters, and solution architect.
6. Incident Management Decisions: Addressing unexpected issues like critical bugs or system outages, involving tech leads, project director, and solution architect.
7. Governance and Compliance Decisions: Ensuring project adherence to policies and regulations, involving the project director, solution architect, and compliance officer.

Environments

Much of the process environment has already been defined in previous sections. To reiterate the major points above:

1. We will have a hierarchical technical, and project management environment.
2. The solution architect role will be the highest order role of the technical team.
3. The project director role will be the highest order role of the project management team.
4. Engineers that are closest to the product will be encapsulated away from the client.

The requirements engineering handoff will be conducted at the beginning of every sprint from the scrum masters to their development teams. The scrum masters and product owners will be the major point of contact for domain questions for the technical teams.

Tooling and Technologies

GitHub will tentatively be our primary tool for source control and project management. Once the project management team is onboarded, this decision may be changed based on concerns raised by product owners and scrum masters. Other options include Jira and Azure DevOps.

Microsoft Teams will be used for professional meetings with the client, like feature demos between the product owners and the client and user acceptance testing sessions. This will serve as our project's gateway to the client.

Gather will be used as a virtual office for the project team during normal work hours, and it will be expected that all project staff be in the Gather office for the entirety of their workday.

Lucid Chart will be the primary technology used for design diagrams (Entity Relationship Diagrams, Event Storming Flowcharts, etc.). These diagrams will then be stored in source control in SVG format once completed.

System Engineering Model

Considering the scale of this project, an iterative (or life cycle) system engineering process will be the best option for Heartland Escapes. Additionally, with how far apart the engineer is from the client, it would be best to implement model-based system engineering to make sure all important information is captured.

For our iterative system development, we'll use the most common system engineering model in the industry: Agile. Ideally, we will use a waterfall approach for the conceptual design and an agile approach for product design. The project roadmap and high-level event storming will cover the entirety of the client's product. This will give us our initial list of feature development priorities, and a starting point for the iterative detailed design approach that takes place during the production of the product.

Every step of design should have a model outcome. The roadmap should be a flowchart of event paths. The detailed design efforts should result in higher detail events with technical context.

Configuration Management

We'll expect most configuration management to be handled through some form of "infrastructure as code" management system. Whether that's helm or terraform, the infrastructure and configurations will be contained within source control. Any sort of secure configurations will be controlled by a secrets management tool.

Requirements Engineering – Conceptual Design

The requirements engineering process will take place in two steps. As stated in the decision-making section, the project management team will create a preliminary road map using event storming with the client. As an outcome of this process, a feasibility study will be conducted for each event. Additionally, preliminary requirements will be generated. If this discussion about requirements becomes too detailed (which may be the case depending on how involved Heartland Escapes wants to be), requirement elicitation will be conducted in the subsequent step instead.

The next step is the agile backlog population process. Every quarter the requested features from the roadmap will be further refined by defining requirement specifications. This includes defining functional requirements, non-functional requirements, constraints, and acceptance criteria. These requirements will be used to define user stories created by the scrum leaders, which will be assigned a level of effort by the technical leaders. (02DCE, 2017)

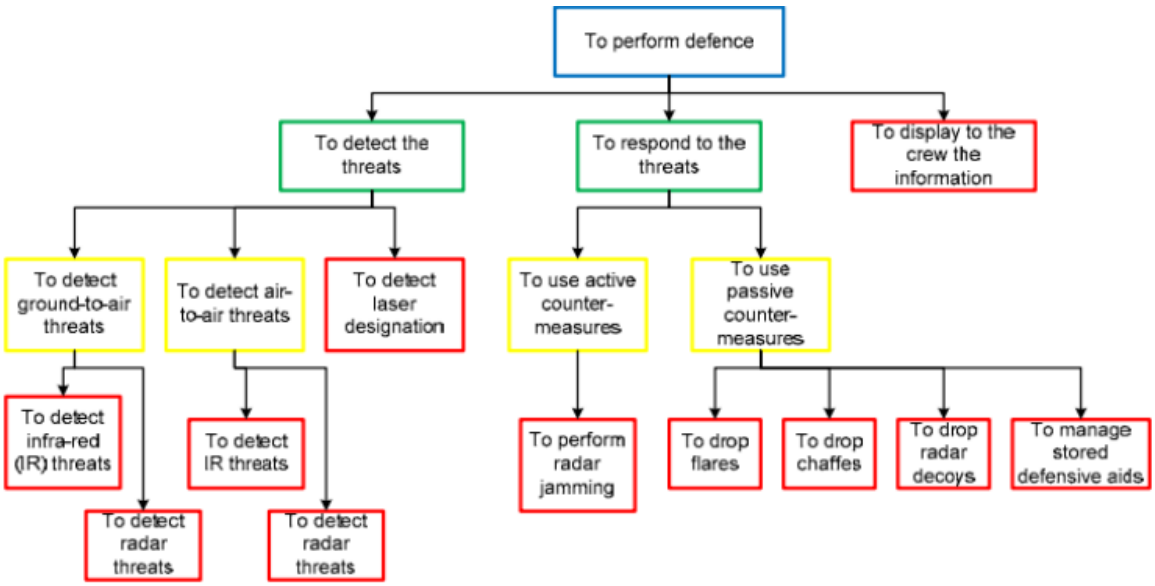
Functional Analysis – Preliminary Design

Client Driven System Roadmap

We'll start by developing the eventing architecture roadmap with the client. This will drive the creation of the subsequent diagrams for the functional analysis. The road map should be a simple process flow chart.

Functional Tree

Using the event roadmap, we'll develop a functional tree that supports all of the defined events within the roadmap. From these events we should be able to derive clear functional requirements for the system. These requirements should go from broad requirements in the higher order of the tree, to more detailed requirements in the branches. The example that Journal of Emerging Technologies and Innovative Research (JETIR) uses is a defense system for a fighter jet. Here is the functional tree diagram that JETIR generated:



Functions/Product Matrix

After the function tree is developed, a function/product matrix should be created. This matrix creation will be a great exercise for the solution architect to determine what infrastructure and tools are needed to create the product desired by Heartland Escapes. This will also allow us to visualize what the Heartland Escapes administrative tool will look like.

Products Tree and Cost Matrix

After the product matrix is defined, the solution architect will now have a list of tools required to deliver Heartland Escapes new product. From this list of tools, the solution architect will develop a cost breakdown of the tools/components required to support the Heartland Escapes administrative tool.

Connection Matrix and Functional Block Diagram

Finally, a connection matrix will be generated to demonstrate what tools need to communicate with one another. This will give the solution architect an understanding of what protocols and support is needed by each tool. Then a functional block diagram can be created showing these relationships in a more visual way. (Kumar, 2015)

Design Processes

As stated in previous sections, the preliminary and conceptual system design will take place before the major development effort begins. This will save the client money and give them good faith in the project before it starts.

During the larger scale development effort, non-functional requirements, constraints, and acceptance criteria will be generated in the iterative process of backlog population and refinement. When new user stories are brought into each sprint, the development teams will create high level designs showing how the user story feature should interact with the system. These diagrams should capture technical requirements of feature implementation, and will support the maintenance team after product completion and feature handoff.

Development Processes

Software

The software development process has been hinted at throughout the course of this proposal. The team will work in sprints, and the length of sprints will be defined by the project director. Every sprint, each team will pull items off the backlog and put them into their sprint queue. They'll discuss the user stories with their scrum master and ensure that they have a proper understanding of the acceptance criteria, requirements, and domain impact. Before development on each user story, the development team will conduct a Story Design and Tasking process which will result in a Lucid Chart design for the user story, and tasks that can be worked on by members of the team.

Once the story is complete, the team will notify the product owner of the completed feature. The product owner will coordinate with the quality assurance team on testing the feature. If the feature passes quality assurance, the product owner will schedule a demonstration for the client. If the client approves of the feature, the user story will be closed, and the cycle restarts.

Hardware

Typically, we will be working in a cloud environment. This is expected to be the case for this Heartland Escapes project as well. As stated in the configuration section, DevOps will handle most of the infrastructure as code development, which is the cloud architecture

equivalent to hardware development. The Hardware development process will act very similarly to the software development process. (

System Integration & Build Management

Before the large-scale development effort, continuous integration and continuous deployment pipelines will be built to support system integration. These CI/CD pipelines will be supported by the DevOps team just like the other infrastructure code. They will also manage software contracts between systems like authentication and authorization configuration management and inter-system security mechanisms.

Verification

The verification process will be conducted by the quality assurance team and the product owners. This testing will ensure that the user story fulfils the requirements defined by the product owners and the client. It will also ensure that no new bugs are introduced.

Validation

The validation process will be the demonstration delivered by the product owners to the stakeholders, and the end user testing by the users of the new system.

Specialty Engineering

Five major areas of specialty engineering are reliability, useability, maintainability, safety, and security. To achieve excellence in these areas on this project, we'll need to create processes and assign individuals.

Reliability and useability are strongly covered by our validation and verification processes. With experienced testers covering boundaries and edge cases of our system, we should be able to ensure the reliability of our system. Additionally, having user acceptance testing will also help us catch any useability issues. Also, having a UX design team will help us catch any useability issues before development.

For maintainability, we've ensured that documentation is captured at every level of the development process. The solution architect will also research and employ development practices that will make the system loosely coupled and more easily maintained. With the ADR process, research will be conducted to vet the best option for third party software and packages outside of the standard framework packages. A criterion for these software decisions should be long-term support and coupling.

As for safety and security, the solution architect will be responsible for ensuring secure development practices. The solution architect and the dev ops tech lead will work together to research and develop secure practices for secrets management and secure message passing. The solution architect and the data tech lead should configure the data to be secure at rest and in transit from the database. It's important to note that security choices will be highly dependent upon the architecture of the system, so many of the security decisions will be made after the project starts.

Additionally, all employees and contractors will be expected to participate in regular learning modules to protect the organization against phishing attacks, flash drives, and other best security practices.

System Deployment

Site Preparation

Site preparation will take place before the large development effort. The Solution Architect along with the tech leads will build the shell application, CI/CD pipelines, and compose the infrastructure as code parts of the repository.

System Installation

The solution architect and a scrum master will work together to task and outline the required processes for system installation. This will be a part of the preliminary work before initial deployment of the shell application.

System Checkout

As stated before, the dev ops tech lead and solution architect will configure the infrastructure as code and CI/CD pipelines to support system checkout. This work will be captured and created before development by the solution architect and a scrum master.

User Training

User acceptance testers will need to be properly onboarded with user access credentials (like VPN access) and training processes. User acceptance testers will be expected to attend demos to the client to understand new features. After major feature completion, release notes will be documented by product owners. User training will be centered around these major feature releases, using the event roadmap as a template. Users will shadow existing users until they understand are comfortable with the system so that questions can be answered in real time.

Support Engineer Training

Training for support engineers will include user access credentials, and a walk through of the technologies. They will also be given a walk through of the system, and the technologies used for authentication and active directory access. They'll start by

shadowing an existing support engineer to get a grasp of the position's day-to-day responsibilities, and slowly phase into a position with less oversight after they understand the position.

Product Support

Maintenance

Feedback Mechanism

Users will participate in surveys quarterly to gauge pain points and user experience. These surveys will allow users to provide how they're feeling about application features from 1 to 10, and an explanation in free-form text. This feedback will be reviewed by product owners and considered for future development of the system by the maintenance team.

Additionally, users will be able to submit issues with the product which will result in a bug ticket process. The ticket will be thoroughly researched and resolved, resulting in a response back to the user from the maintenance team.

Communication Workflow

When an issue is raised by a user, a bug ticket will be created. This ticket will be prioritized based on the impact to the system using a risk assessment matrix. The risk assessment matrix is a five-by-five matrix, whose x axis is severity and y axis is likelihood of occurrence. It will be up to the maintenance team to assign criticality levels to each box within the five-by-five matrix. After a prioritized bug is picked up by a developer, research into the bug will be conducted to recreate the issue. After the bug is resolved, the resolution will be reported back to the user who identified the issue. Depending on the severity of the issue, a user-wide announcement may be needed to convey the resolution to all users.

For the user surveys, if work is identified to better the system that the maintenance team wants to take on, the new feature will be analyzed and tasked out. This new feature work will be prioritized alongside existing bugs. When the feature is developed and rolled out, an announcement will be made to all users broadcasting the new feature's availability.

Logistics Support

Most of the logistics support will come from the maintenances team's cloud engineer(s). With the system being mostly supported within the cloud, it will be necessary to identify how to add backup components to the cloud system. If more RAM or long-term storage space is needed, that process will need to be identified for the maintenance team. Additionally, geographic redundancy will need to be considered in the case of an outage.

Disposal

The system should be built in a way that allows for sub-component disposal without breaking the rest of the system. It will be the responsibility of the solution architect to enforce this development practice throughout the development process. Feature flags and their responsible components will need to be well documented for the maintenance teams. If disposal of the entire system is required, the cloud developers should be able to easily shut down the system using their cloud providers portal.

References

02DCE. (2018, June 17). *Software Engineering | Requirements Engineering Process - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/

AGILE PRACTICE GUIDE. (2017). Project Management Institute Inc. https://www.agilealliance.org/wp-content/uploads/2021/02/AgilePracticeGuide.pdf

Bowman, A. (2023, July 26). *Appendix J: SEMP Content Outline - NASA*. NASA.gov. https://www.nasa.gov/reference/appendix-j-semp-content-outline/

Kumar, V. (2015). Functional Analysis in Systems Engineering: Methodology and Applications. *Journal of Emerging Technologies and Innovative Research*, *2*(10). https://www.jetir.org/papers/JETIR1701952.pdf

Blanchard, B. S., & Fabrycky, W. J. (2016). Systems Engineering and Analysis (5th ed.). Pearson Learning Solutions. https://coloradotech.vitalsource.com/books/9781323417522