Systems Engineering for Intelligent Transportation Systems

An Introduction for Transportation Professionals

January 2007

U.S. Department of Transportation
Federal Highway Administration
Federal Transit Administration

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Purpose

This guide is intended to introduce you to systems engineering and provide a basic understanding of how it can be applied to planning, designing, and implementing intelligent transportation systems (ITS) projects. The guide leads you step by step through the project life cycle and describes the systems engineering approach at each step. It describes how to begin implementing the systems engineering approach on your next ITS project and incorporate it more broadly into your organization's business processes and practices.

Reading this guide will make you conversant in systems engineering and familiar with the way that it is being applied to ITS projects today. It won't make you a systems engineering expert. Many excellent and comprehensive resources are available that describe every aspect of systems engineering in detail. These resources are identified throughout the guide in case you want more information.

This document is a resource and a learning tool on the topic of systems engineering. **It is not formal guidance from US DOT on how to meet the systems engineering requirements in FHWA Rule 940 and the FTA National ITS Architecture Policy**. Compliance with the Rule/Policy is actually established by each FHWA Division and FTA Regional Office. It is strongly recommended that you contact your federal representative for the specific requirements in your state.

1.2 Intended Audience

This guide is designed for ITS project managers, system owners, operators, maintainers, and anyone else in need of a quick, approachable primer on the basics of systems engineering for ITS. We assume you have a transportation background and know something about ITS, but you don't need any previous knowledge of systems engineering to benefit from this guide.

You might have noticed that systems engineers are not included in the above list. The intended audience is not systems engineers, since they should already be familiar with the information in this guide. The guide is intended for all the other transportation professionals who are involved in ITS project development and will need to know something about systems engineering to ensure that it is correctly and productively applied to their projects.

1.3 Navigating the Document

This document includes seven chapters that are organized to introduce you to systems engineering and then to describe how systems engineering can be applied to your ITS projects.

Here is a breakdown of the six remaining chapters and what you will find in each:

Chapter 2: Why Use Systems Engineering? provides some motivation for reading the rest of the document. It briefly explains why systems engineering should be used for ITS projects and gives some background on the FHWA Rule and FTA Policy requirements for systems engineering.

Chapter 3: What is Systems Engineering? sets the stage for the following chapters by defining some key terms and explaining the guiding principles behind systems engineering. The "V" model that adorns the cover of this document is introduced here.

Chapter 4: ITS Technical Processes follows an ITS project from initial project identification all the way through retirement of the implemented system. The systems engineering approach is described as you traverse the "V" model and step through topics like Concept of Operations, requirements, and design. This is the heart of the document.

Chapter 5: ITS Project Management Processes describes project planning, risk management, project monitoring and control, and configuration management. These processes are used to manage the ITS project so that it is completed on time and on budget. They complement the technical processes that are described in the previous chapter. If you are not familiar with these project management processes, you should briefly familiarize yourself with this chapter as you encounter references to the processes in Chapter 4.

Chapter 6: Applying Systems Engineering shows how the systems engineering process can be applied to your next ITS project. This chapter discusses procurement, development approaches, and tailoring the systems engineering process to fit the needs of your project. It also describes how organizations build systems engineering into their business practices.

Chapter 7: Resources lists many excellent books, reports, training courses, and other systems engineering resources that you can use to learn more about any of the systems engineering topics that are introduced in this document.

1.4 About the Icons

Icons are used to highlight different kinds of information throughout this document.

This "lightbulb" icon identifies suggestions that may improve the systems engineering analysis or the quality of the systems engineering products that are created. Usually based on actual experience, these are ideas that have worked in the past.

This "exclamation point" icon flags warnings. In contrast to tips, these are problems that have been encountered that you should avoid. Also frequently based on actual experience, these are ideas that have NOT worked in the past.

This "*i* for information" icon highlights resources that offer additional information related to systems engineering, including books, reports, presentations, and other documents. Chapter 7 includes a list of all the resources that are identified in this document.

This "scissors" icon identifies ways that the systems engineering process can be tailored for smaller ITS projects. Many ITS projects are relatively low risk and low complexity, and the systems engineering process should be tailored accordingly. Section 6.2.2 provides a more comprehensive discussion of how to tailor the systems engineering approach.

This "monitor" icon identifies software tools (programs, databases, spreadsheets, etc.) that support some aspect of the systems engineering or ITS project development processes. The information provided is not intended to endorse or recommend any particular tool.

This "scales" icon identifies references to the FHWA Rule and FTA Policy on ITS Architecture and Standards. These are normally references to the portion of the rule/policy related to systems engineering analysis (Sections 940.11 of the Rule and VI of the Policy). (See http://www.ops.fhwa.dot.gov/its_arch_imp/policy.htm regarding the Rule/Policy.)

This "book" icon is used where ITS and systems engineering terminology is defined. Terminology is one of the first hurdles to overcome in any new subject area. Readers can skip quickly past the definitions of familiar terms.

2 WHY USE SYSTEMS ENGINEERING?

2.1 Value of Systems Engineering

Although ITS projects come in many shapes and sizes, they all use technology (computers, communications, sensors, etc.) and frequently include the exchange of information, either within a system or between systems. The technology and integration that sets ITS projects apart also creates challenges for the ITS project manager. What every ITS project manager wants is a successful result at the end of the project, with "success" measured by:

- how well the implementation satisfies the needs of the people who use it, and
- how closely the project stayed within the budgeted cost and schedule.

Systems engineering reduces the risk of schedule and cost overruns and increases the likelihood that the implementation will meet the user's needs. Other benefits include:

- improved stakeholder participation
- more adaptable, resilient systems
- verified functionality and fewer defects
- higher level of reuse from one project to the next, and
- better documentation.

These assertions have been supported by several studies that have shown that good systems engineering results in better cost and schedule performance. Studies have been performed by the International Council of Systems Engineering (INCOSE)[1], Boeing[2], and IBM[3], among others. Figure 1 shows the results of an INCOSE study that collected both planned and actual project and systems engineering cost data for 44 projects. The survey indicated that investing in systems engineering did improve project cost performance. The responses indicated a 50% overrun on average without systems engineering and a clear trend toward better cost performance results with systems engineering.

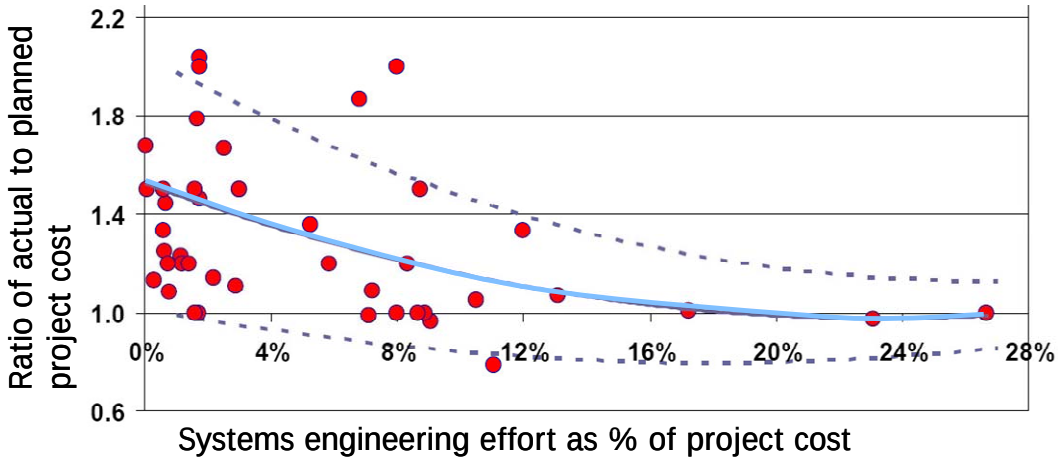


Figure 1: Systems Engineering Improves Project Cost Performance

[1]Eric Honour, "Understanding the Value of Systems Engineering", 2004.

[2]John D. Vu, "Software Process Improvement Journey: From Level 1 to Level 5", 2001.

[3]Bruce Barker, IBM Commercial Products, 2003.

2.2 US DOT Policy

US DOT recognized the potential benefit of the systems engineering approach for ITS projects and included requirements for a systems engineering analysis in the FHWA Rule/FTA Policy that was enacted on January 8, 2001. The Rule/Policy requires a systems engineering analysis to be performed for ITS projects that use funds from the Highway Trust Fund, including the Mass Transit Account. As shown in an excerpt from the Rule in Figure 2, the Rule/Policy actually specifies seven requirements that the systems engineering analysis must include at a minimum.

The Rule/Policy allows each project sponsor to use a systems engineering approach that is tailored to fit the needs of each ITS project. As you will see in the following chapters, the systems engineering approach is actually broader than the seven specific requirements identified in the Rule/Policy. If you implement a good systems engineering process, you will meet or exceed the specific systems engineering analysis requirements identified in the Rule/Policy.

The FHWA Division and FTA Regional Offices determine how the systems engineering analysis requirements in the Rule/Policy should be applied to ITS projects in each region and how compliance should be demonstrated by each project sponsor. Federal oversight is provided based on oversight requirements defined in the stewardship agreements with each state. Several states have established checklists that prompt project sponsors to consider the systems engineering analysis requirements as part of the project development process, as shown in Figure 3.[4] Contact the ITS specialist in your FHWA Division Office or FTA Regional Office for more information.

Note that each organization may establish its own systems engineering process requirements that satisfy the requirements of the Rule/Policy, as Florida DOT did with its Systems Engineering Management Plan (see http://www.floridaits.com/SEMP).

§ 23 CFR 940.11 Project implementation.
(a) All ITS projects funded with highway trust funds shall be based on a systems engineering analysis.
(b) The analysis should be on a scale commensurate with the project scope.
(c) The systems engineering analysis shall include, at a minimum:
(1) Identification of portions of the regional ITS architecture being implemented (or if a regional ITS architecture does not exist, the applicable portions of the National ITS Architecture);
(2) Identification of participating agencies' roles and responsibilities;
(3) Requirements definitions;
(4) Analysis of alternative system configurations and technology options to meet requirements;
(5) Procurement options;
(6) Identification of applicable ITS standards and testing procedures; and
(7) Procedures and resources necessary for operations and

Figure 2: FHWA/FTA Systems Engineering Analysis Requirements

In **California**, a Systems Engineering Requirements Form, or SERF, must be completed by the project sponsor at project initiation. This form, included in the Caltrans Local Assistance Procedures Manual, includes one question for each of the seven systems engineering requirements in Rule 940.11. The SERF checklist is a streamlined form that is only one or two pages long, but it is enough to ensure that each project sponsor will address the systems engineering requirements of the rule. **Virginia DOT** has also implemented a Systems Engineering and Architecture Compliance (Rule 940) checklist for use in Northern Virginia. Many other states are implementing similar forms.

Figure 3: Examples of Systems Engineering Analysis Requirements Checklist

[4]The SERF is available at www.dot.ca.gov/hq/LocalPrograms/lam/lapm.htm. The Virginia DOT checklist is available at www.vdot-itsarch.com/nova/docs/rule940checklist.doc.

3 WHAT IS SYSTEMS ENGINEERING?

3.1 A Few Definitions

In true systems engineering fashion, let's begin with a few basic definitions before we jump into the details of the systems engineering discipline.

What is a System?

Everyone uses the term and has an intuitive notion of what a system is, but there is a formal definition. INCOSE defines a system as:

> *"A combination of interacting elements organized to achieve one or more stated purposes."*

This general definition covers almost everything you can think of – household appliances, transportation management systems, the latest weapon system – all of these are systems.

What is Systems Engineering?

Since the term was coined in the 1950s, systems engineering has evolved from a process focused primarily on large-scale defense systems to a broader discipline that is used in all kinds of project development. Systems engineering can be applied to any system development, so whether you are developing a household appliance, building a house, or implementing a sophisticated transportation management system, systems engineering can be used. INCOSE defines systems engineering like this:

> *Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem.*
>
> *Systems Engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems Engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.*

Note that this definition is very broad – it covers the project life cycle from needs definition to system disposal. It includes technical activities like requirements and design, as well as project activities like risk management and configuration management. Systems engineering provides a systematic process and tools that directly support project management.

What is an ITS Project?

In order to apply systems engineering to ITS projects in accordance with the FHWA Rule/FTA Policy, it is important to define an *ITS project*. Rule 940 defines ITS projects quite broadly:

> *ITS Project means any project that in whole or in part funds the acquisition of technologies or systems of technologies that provide or significantly contribute to the provision of one or more ITS user services as defined in the National ITS Architecture.*

This definition encompasses a wide range of projects. Smaller ITS projects might be limited to the purchase and installation of field equipment – controllers, ramp meters,

signals, etc. Larger ITS projects support integration of multiple systems and development of custom software – for example, transportation management centers and 511 traveler information systems. These ITS projects are vastly different in complexity and in the amount of systems engineering that is needed. The FHWA Division/FTA Regional Offices establish and monitor how systems engineering analysis requirements are levied on specific ITS projects.

3.2 Key Principles

There are a handful of fundamental challenges and important concepts that shape and drive the systems engineering discipline.

3.2.1 Project Development Challenges

Project Initiation Euphoria

In the first days of any new project, the mood is optimistic and expectations are high. Just over the horizon, reality is looming, and technology, schedule, and funding constraints may ultimately cause the project to fall short of goals that were established in its early days. The need to balance these natural inclinations and real-world constraints is an important driver for implementing a systematic systems engineering approach at the outset to guide the team and manage expectations.

Cone of Uncertainty

At the beginning of a high-technology project, there may also be significant uncertainty in the project cost and schedule estimates. The less experience the project team has with similar projects, the more uncertainty there will be. The estimates naturally get better as work progresses and the project team gains a better understanding of the system they are building. At project completion, all the uncertainty has been removed – the team knows exactly how much was spent. When you plot the uncertainty against time (see Figure 4), it looks like a cone, which is why Barry Boehm called this challenge the "cone of uncertainty".

Systems engineering focuses on resolving uncertainty early in project development by establishing the project scope and defining good requirements. Incremental development strategies also help to mitigate the risk of unreliable estimates early in the project.

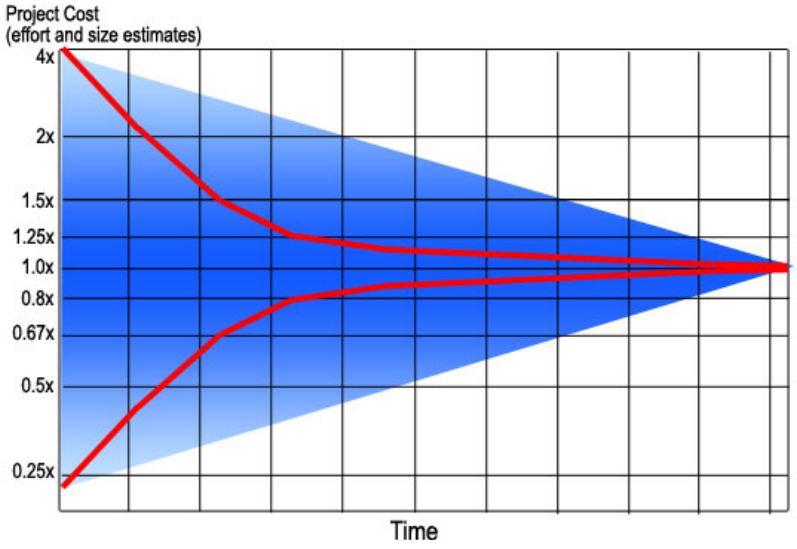


Figure 4: Cone of Uncertainty

The Wrong Procurement Method Can Tie Your Hands

The traditional procurement methods that have been used for decades in highway construction are often not suitable for ITS projects. For example, the Low Bid method uses a consultant to prepare a design specification that is then implemented by a contractor who submits the lowest bid. This method works well for building roads, but experience shows that it does not work well for many ITS projects that frequently require collaboration and iteration between the design and implementation phases. It is vital to select the right procurement method so that you can implement the right systems engineering approach for your project. (See Section 6.2.1 for more information on procurement.)

Late Changes Drive Project Costs

There is no such thing as a mistake-free project development. In the transportation industry, experienced construction managers will tell you that every project has change orders. The problem is that change orders during construction are more expensive to the project. A mistake or missed system feature that is not recognized until after project closeout will be even more expensive to address.

Studies[5] of software development projects have shown that this "latency cost" can increase the cost of fixing a mistake dramatically. As shown in Figure 5, for example, a bad requirement will be relatively cheap to fix while you are still in the requirements phase (1x) but increasingly expensive to fix later in project development. This is because you not only have to fix the bad requirement later in the project, you also have to fix the design and implementation problems that were caused by the bad requirement. The problems compound themselves if they are left uncorrected.

In systems engineering, verification and validation of the evolving project documentation is performed early and often to maximize the chances of identifying defects as early in the project development cycle as possible.

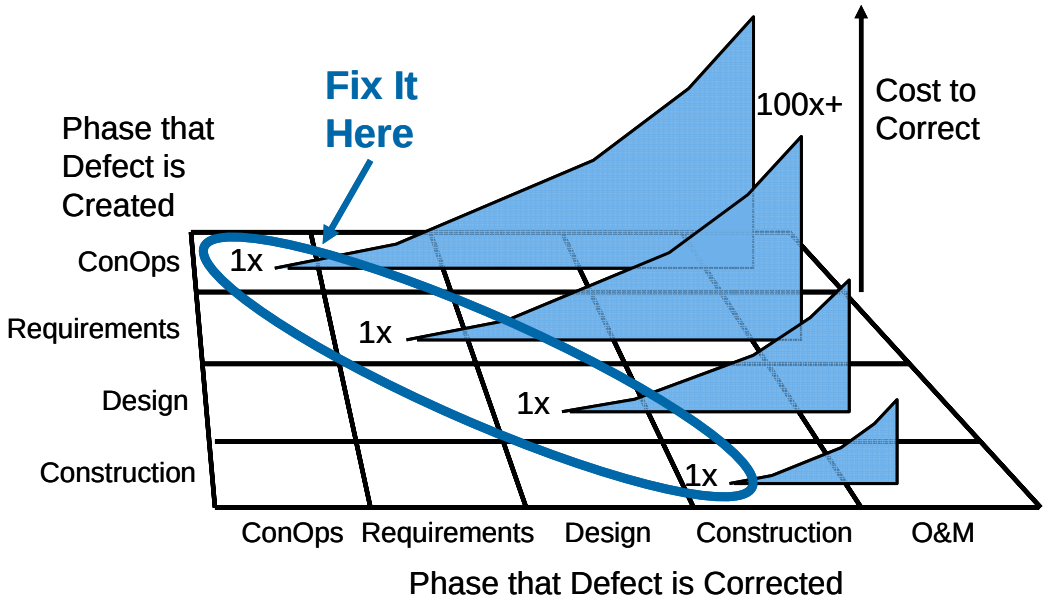


Figure 5: Late Changes Drive Project Costs
(Adapted from Steve McConnell, *Code Complete*)

[5]See, for example, Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, 1992.

The Odds Are Against Success

The Standish Group has done more than 10 years of research, collecting statistics on information technology projects, and their findings have consistently painted a dismal (albeit slowly improving) picture. For example, in 2004, as shown in Figure 6, only 34% of the projects surveyed met the criteria for success – completed on time, on budget, and with all the features originally specified. Of the 280,000 projects surveyed that year, more than 142,000 were late or over budget and another 42,000 failed outright.

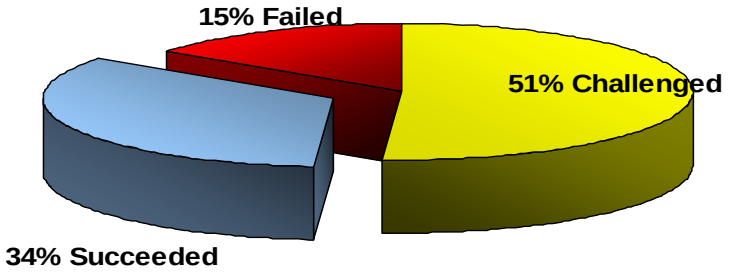


Figure 6: Standish Group: 2004 CHAOS Report Project Success Rate

While the infamous failure rates are the most often repeated information, the report also identifies success factors that are identified through the same project surveys. Many of these success factors (including user involvement, minimized scope, and firm basic requirements) are related to the systems engineering process. Systems engineering won't guarantee success, but it will help you to identify issues earlier in the project schedule and will improve your chances for a successful project in the end.

3.2.2 Systems Engineering Principles

Start with Your Eye on the Finish Line

You should reach consensus at the very beginning of the project on what will constitute success at the end. This means that the stakeholders should start with an agreement of what the project should accomplish and the metrics that will be used to measure the success of the project. This initial focus on the finish line must be sustained by project management as project development progresses and competing interests and project complexities begin to dominate the day-to-day work.

Stakeholder Involvement is Key

Successful projects involve the customer, users, operators, and other stakeholders in the project development. Systems engineering is a systematic process that includes reviews and decision points intended to provide visibility into the process and encourage stakeholder involvement. The systems engineering process includes stakeholders through all stages of the project, from initial needs definition through system verification and acceptance. The stakeholders who are involved in any particular step will vary, providing managers, operators, and technical personnel with an opportunity to contribute to the steps in the process where their input is needed.

Define the Problem Before Implementing the Solution

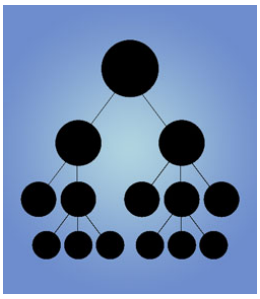
Very often, you'll have a solution already in mind at the start of a project and may even find yourself "backing into" requirements to match your solution. Resist this temptation and instead use the systems engineering process to first define the problem. You'll find that there are actually multiple ways to solve the problem, and a good trade study will help you to determine the best solution on the basis of a clear understanding of the requirements.

Delay Technology Choices

Technology is constantly changing. The choices available when a project is initially conceived may well be replaced by better technology by the time the project is implemented. Specifying technology too early will result in outdated technology or constant baseline changes as you try to keep up with technology advancements. It's best to follow the systems engineering process by defining the needs, requirements, and high-level design without specifying technology. You'll have a stable baseline, and you'll be able to make the most appropriate technology choices when it is time to implement.

Baseline is a frequently used term in systems engineering. A baseline is a reference point against which everyone on the project team works, so you want to control the changes that are made to the baseline. The process of establishing and controlling project baselines is *configuration management*, which is discussed in Section 5.4.

Divide and Conquer



Many systems are large and complex. A key systems engineering strategy is the decomposition of such a system into smaller subsystems and then of the subsystems into more manageable hardware and software components. These simpler components are easier to understand and define and ultimately are easier to build. Much of the systems engineering process is built around this approach – breaking down a big problem into many smaller components that can be individually solved and then recombined.

Connecting the Dots – Traceability

As you move from one step to the next in the systems engineering process, it is important to be able to relate the items in one step with those in another. The relationship between items is called *traceability*. For example, you use traceability to relate a requirement to the subsystem that will implement the requirement. Traceability connects many items together. The requirement will be related to a user need as well as to a test that will be used to verify the requirement. Traceability is a powerful concept that allows you to be certain that the system that is implemented at the end of the project is directly connected with the user needs that were identified at the beginning.

3.3 The "V" Systems Engineering Model

Many different process models have been developed over the years that specify a series of steps that make up the systems engineering approach[6]. Among these models, the "V" model, shown in Figure 7, is emerging as the de facto standard way to represent systems engineering for ITS projects.

Don't be surprised if you come across different spellings for the "V" model. Some books, guides, and other resources refer to the same V-shaped model as the "Vee" model. If it looks like a "V" and it sounds like a "V", then it is a reference to the same basic model, whether it is spelled "V" or "Vee".

3.3.1 Overview of the "V" Model

Since it was first developed in the 1980s, the "V" model has been refined and applied in many different industries. Wings have been recently added to the "V" as part of its adaptation for ITS to show how project development fits within the broader ITS project life cycle. The left wing shows the regional ITS architecture, feasibility studies, and concept exploration that support initial identification and scoping of an ITS project based on regional needs. A gap follows the regional architecture(s) step because the regional architecture is a broader product of the planning process that covers all ITS projects in the region. The following steps in the "V" are for a specific ITS project. The central core of the "V" shows the project definition, implementation, and verification processes. The right wing shows the operations and maintenance, changes and upgrades, and ultimate retirement of the system. The wings are a key addition to the model since it is important to consider the entire life cycle during project development.

As shown in the "V", the systems engineering approach defines project requirements before technology choices are made and the system is implemented. On the left side of the "V", the system definition progresses from a general user view of the system to a detailed specification of the system design. The system is decomposed into subsystems, and the subsystems are decomposed into components – a large system may be broken into smaller and smaller pieces through many layers of decomposition. As the system is decomposed, the requirements are also decomposed into more specific requirements that are allocated to the system components.

As development progresses, a series of documented baselines are established that support the steps that follow. For example, a consensus Concept of Operations supports system requirements development. A baseline set of system requirements then supports system design. The hardware and software are implemented at the bottom of the "V", and the components of the system are then integrated and verified in iterative fashion on the right. Ultimately, the completed system is validated to measure how well it meets the user's needs. (Each of the steps in the "V" are defined in detail in Chapter 4.)

[6]The "Waterfall", "Spiral", and "V" are the most common system development models. Among many sources, http://www.floridaits.com/SEMP/Files/PDF_Report/030220-TMI-V2.pdf provides a good description of the different models.

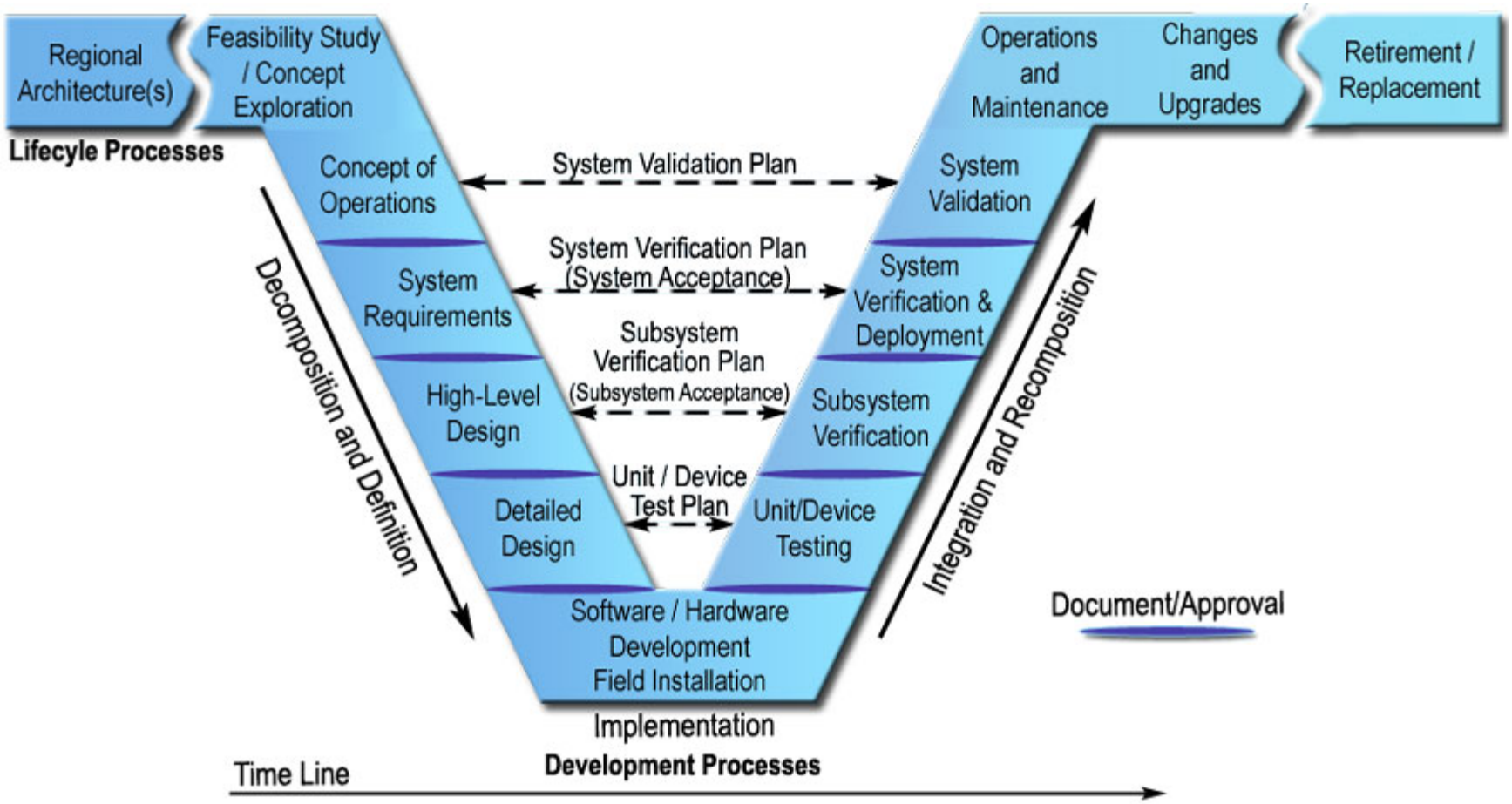


Figure 7: Systems Engineering "V" Diagram

3.3.2 Connecting the Left and Right Sides of the "V"

One of the first things that strikes you about the "V" is the symmetry between the left and right sides of the model. This symmetry reflects the relationship between the steps on the left and the steps on the right. The system definition that is generated on the left is ultimately used to verify the system on the right. For example, the user needs and performance measures that are identified in the Concept of Operations are the basis for the System Validation Plan that is used to validate the system at the end of project development. Similarly, a System Verification Plan is developed with the System Requirements so that the engineers consider how to verify each requirement as the requirements are written.

The connections between the left and right are indicated by the arrows that cross the "V", showing how plans developed on the left drive the process on the right. These connections provide continuity between the beginning and end of project development and ensure that the engineers are focused on the completion of the project from the beginning. The connections between the left and right sides of the model reflect one of the systems engineering principles – start with your eye on the finish line.

3.3.3 Decision Points

Projects have been managed for years using Gantt charts that identify tasks and major milestones. You don't start the next task until you have completed the previous supporting tasks and passed the intervening milestone. The "V" diagram is similarly punctuated by a series of major milestones (labeled Document/Approval in the figure) where the output of the previous step is reviewed and the customer and project team determine whether the project is ready to move to the next step in the process. The project moves forward only if the criteria for the decision point have been satisfied. Decision points are important milestones that provide visibility into the project development and allow for issue identification and course correction during development. (Decision-point reviews are covered in more detail in Section 5.2.2.)

4 ITS TECHNICAL PROCESSES

This document is intended to help you understand how systems engineering can be used throughout the ITS project life cycle. Chapters 4 and 5 present two different types of processes that support systems engineering:

1. **Technical processes,** such as system requirements, high-level design, integration, and verification, are described here in Chapter 4. These processes, depicted in the "V" systems engineering model, are performed to develop an ITS project that meets the user's needs. This chapter leads you step by step through each of the technical processes in the "V". Each process is summarized, key activities are identified, and outputs that should be generated are defined.
2. **Project management processes,** such as project planning, risk management, and configuration management, are described in Chapter 5. These cross-cutting activities are just as important to the success of the project, but they do not appear in the "V" diagram because they apply to many different steps in the "V". These processes are used to plan, monitor, and control the ITS project so that it is completed on time and on budget.

Relationship to Traditional Transportation Processes

ITS projects are identified and funded through transportation planning and programming/budgeting processes in each state, planning region (e.g., metropolitan planning area), and agency. The "V" diagram and the systems engineering process begin once a need for an ITS project has been identified. The early steps in the "V" define the project scope and determine the feasibility and acceptability as well as the costs and benefits of the project. These early steps actually support planning and programming/budgeting since they are intended to identify high-level risks, benefits, and costs and to determine if the ITS project is a good investment. The latter steps support project implementation, then transition into operations and maintenance, changes and upgrades, and ultimate retirement or replacement of the system. (The systems engineering "V" is placed in context with the traditional transportation project life cycle in Section 6.1.)

Technical Documentation

Each step of the process that is described in this chapter results in one or more technical outputs. This documentation is used in subsequent steps in the process and provides a critical documentation trail for the project. The documentation that is discussed in this chapter is identified in Table 1, which provides a bird's-eye view of where it fits in the "V". Several resources provide good descriptions and templates for this documentation.[7] Note that not every ITS project will require every document listed in the table. (More information on tailoring is provided later in this chapter and in Section 6.2.3.)

About the Examples

This chapter is illustrated with real examples that show how different agencies have used the systems engineering process for their ITS projects. These real examples aren't perfect and shouldn't be taken as the only approach, or even the best approach, for accomplishing a particular step. As time goes by and we gain experience using systems engineering on ITS projects, many more examples will become available.

[7]The California Systems Engineering Guidebook and the Florida Systems Engineering Management Plan both include good documentation descriptions and templates. See Chapter 7 for more resources.

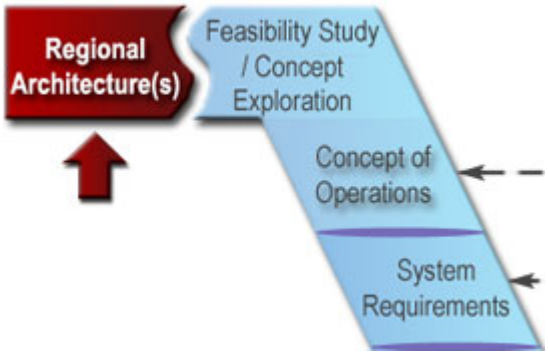
Table 1: Technical Documentation in the "V" Systems Engineering Process

Documentation / Chapter/Process Step	4.1 Using the Regional ITS Architecture	4.2 Feasibility Study/Concept Exploration	4.3 Concept of Operations	4.4 System Requirements	4.5 System Design	4.6 SW/HW Development & Testing	4.7 Integration & Verification	4.8 Initial Deployment	4.9 System Validation	4.10 Operations and Maintenance	4.11 Retirement/Replacement
Relevant portion of Reg ITS Arch	C		U	U	U						
Feasibility Study		C	U								
Concept of Operations			C	U							
System Validation Plan			C						U		
System Requirements Document				C	U	U	U	U	U	U	U
System Verification Plan				C			U				
Traceability Matrix				C	U		U			U	U
System Acceptance Plan				C				U			
High-Level (Architectural) Design					CU	U					
Detailed Design Specifications					C	U					
Interface Specifications					C	U	U				
Subsystem Verification Plans					C		U				
Integration Plan					C		U				
Subsystem Acceptance Plan					C		U				
Unit/Device Test Plan					C	U					
SW/HW Development Plans						CU					
Verification Procedures							CU				
Delivery & Installation Plan								CU			
Transition Plan								CU			
O&M Plan and Procedures								C		U	
System Validation Procedures									CU		
System Retirement Plan											CU

C: Create documentation U: Primary Use/Update of the documentation

4.1 Using the Regional ITS Architecture

In this step: The portion of the regional ITS architecture that is related to the project is identified. Other artifacts of the planning and programming processes that are relevant to the project are collected and used as a starting point for project development. This is the first step in defining your ITS project.



OBJECTIVES	▪ Define the project scope while considering the regional vision and opportunities for integration ▪ Improve consistency between ITS projects and identify more efficient incremental implementation strategies ▪ Improve continuity between planning and project development
INPUT ***Sources of Information***	▪ Relevant regional ITS architecture(s) ▪ Regional/national resources supporting architecture use ▪ Other planning/programming products relevant to the project
PROCESS ***Key Activities***	▪ Identify regional ITS architecture(s) that are relevant to the project ▪ Identify the portion of the regional ITS architecture that applies ▪ Verify project consistency with the regional ITS architecture and identify any necessary changes to the regional ITS architecture
OUTPUT ***Process Results***	▪ List of project stakeholders and roles and responsibilities ▪ List of inventory elements included in or affected by the project ▪ List of requirements the proposed system(s) must meet ▪ List of interfaces and the information to be exchanged or shared by the system(s) ▪ Regional ITS architecture feedback as necessary
REVIEW ***Proceed only if you have:***	▪ Demonstrated consistency with the regional ITS architecture and identified needed changes to the regional ITS architecture, if applicable ▪ Extracted the relevant portion of the regional ITS architecture that can be used in subsequent steps ▪ Reached consensus on the project/system scope

4.1.1 Overview

The regional ITS architecture provides a good starting point for systems engineering analyses that are performed during ITS project development. It provides region-level information that can be used and expanded in project development.

When an ITS project is initiated, there is a natural tendency to focus on the programmatic and technical details and to lose sight of the broader regional context. Using the regional ITS architecture as a basis for project implementation provides this regional context as shown in Figure 8. It provides each project sponsor with the opportunity to view their project in the context of surrounding systems. It also prompts the sponsor to think about how the project fits within the overall transportation vision for the region. Finally, it identifies the integration opportunities that should be considered and provides a head start for the systems engineering analysis.

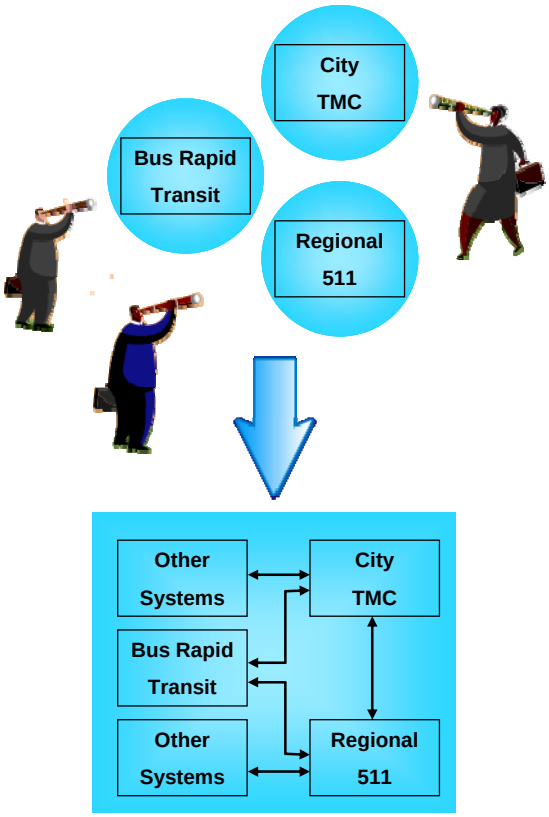


Figure 8: Regional ITS Architecture Framework for Integration

The *regional ITS architecture* is a tool that is used in transportation planning, programming, and project implementation for ITS. It is a framework for institutional agreement and technical integration for ITS projects and is the place to start when defining the basic scope of a project.

The regional ITS architecture is the first step in the "V" because the best opportunity for its use is at the beginning of the development process. The architecture is most valuable as a scoping tool that allows a project to be broadly defined and shown in a regional context. The regional ITS architecture step and the concept exploration step that is described in the next section may iterate since different concepts may have different architecture mappings. The initial architecture mapping may continue to be refined and used as the Concept of Operations and system requirements are developed.

The Regional ITS Architecture Guidance Document provides detailed guidance for regional ITS architecture development, use, and maintenance. (Version 2 of this document provides detailed guidance for using a regional ITS architecture to support project implementation.)

4.1.2 Key Activities

Initial use of the regional ITS architecture requires a few basic activities: locating the right architecture, identifying the portion of the architecture that applies to your project, and notifying the architecture maintainer of any required regional architecture changes. None of these tasks is particularly time consuming – the basic extraction of information can be done in an afternoon, even for a fairly complex project, if you are knowledgeable about the regional ITS architecture. Of course, it can be time consuming to climb the learning curve, and coordinating and building consensus on the scope of the project will require time and effort. Each of the key activities is described in the following paragraphs.

- **Identify regional ITS architecture(s) that are relevant to the project –** First, find the regional ITS architecture that covers the geographic area where your project will be implemented. In some cases, more than one regional ITS architecture may apply. For example, a major metropolitan area may be included in a statewide architecture, a regional architecture for the metropolitan area, and subregional architectures that are developed for a particular agency or jurisdiction. Coordinate with the ITS specialist at the FHWA Division/FTA Regional Office if necessary to sort this out.

  In the event that no regional ITS architecture exists at the time that an ITS project is initiated, coordinate with the FHWA Division/FTA Regional Office on starting a regional ITS architecture effort. In the interim, a project-level architecture should be developed based on the National ITS Architecture[8] to support the ITS project.

- **Identify the portion of the regional ITS architecture that applies –** Next, identify the portion of the regional ITS architecture(s) that are applicable to your project or identify the portion of the National ITS Architecture that applies if a regional ITS architecture does not exist. Document any constraints that the architecture may place on the project, including ITS standards that may be applicable.

The systems engineering analysis requirements identified in FHWA Rule 940.11/FTA Policy Section VI require identification of the portion of the regional ITS architecture that is implemented by each ITS project that uses federal funds. If a regional ITS architecture does not exist, then the portion of the National ITS Architecture that will be implemented by the project must be identified.

You should build consensus around the fundamental project scope decisions that are made as the relevant portions of the regional ITS architecture are identified. One good approach is to create a context diagram that shows the ITS system to be implemented in the middle of the diagram surrounded by all other potentially interfacing systems in the region. For example, Figure 9 is a context diagram for the MaineDOT Communications Center. A context diagram can be used to discuss integration opportunities that should be considered in this project and in future projects. A discussion like this puts the ITS project in context and raises awareness of future ITS integration opportunities. It also may highlight regional ITS architecture issues that should be addressed.

In almost every case, the regional ITS architecture will identify potential integration opportunities that will not be included in the current project. Specific integration options may be deferred for many reasons – agencies on both sides of the interface may not be ready, there may not be sufficient funding or time available to implement everything, supporting infrastructure may not yet be completed, a necessary standard may not be available, implementing too much at once may incur too much complexity/risk, etc.

Even if they are deferred, it is important to account for future integration options in the current project design. The ultimate goal is to make ITS deployment as economical as possible by considering how this project will support future projects over time. To support this objective, future integration options that may impact the project design should be identified and considered in the project development. For example, additional stakeholders may be involved in the current project to ensure that future interface requirements are identified and factored into the current project design.

- **Verify consistency with the regional ITS architecture and identify any necessary changes to it –** Confirm that nothing planned in your project would be counter to the

[8]Visit http://www.its.dot.gov/arch/ for more information on the National ITS Architecture.

regional ITS architecture. If you determine that the regional ITS architecture should be updated to more accurately reflect your ITS project, submit the required changes to the regional ITS architecture maintainer named in the architecture maintenance plan.

Each region should define a mechanism that allows the project team to provide comments on the architecture with minimal time investment. Project teams that use the architecture will be among the most significant sources for regional ITS architecture maintenance changes, and the region should strive to facilitate this feedback. If your region does not have such a mechanism, consult the Regional ITS Architecture Guidance Document for more information on facilitating architecture use and maintenance in your region.

4.1.3 Outputs

The first output of this step is the subset of the regional ITS architecture for the ITS project. While the Rule/Policy requires a subset of the regional ITS architecture to be identified, it does not define the components that should be included. You should consult local guidelines or requirements to help make this determination. In most cases, the following components will precisely define the scope of the project: (1) stakeholders, (2) inventory elements, (3) functional requirements, and (4) information flows.

These four components define the system(s) that will be created or impacted by the project, the affected stakeholders, the functionality that will be implemented, and the interfaces that will be added or updated. Other components may be identified, including market packages, roles and responsibilities, relevant ITS standards, and agreements. For very large ITS projects, this might be several pages of information. For a small ITS project, this might fit on a single page. The information that is extracted will actually be used in the concept exploration, Concept of Operations, requirements, and design steps that follow.

The Turbo Architecture software tool can be used to quickly and accurately define an ITS project architecture if the regional ITS architecture was developed with Turbo. Turbo Architecture can be used to generate diagrams and reports that fully document the portion of the regional ITS architecture that will be implemented by the project. Turbo Architecture can also be used to develop a project ITS architecture based on the National ITS Architecture if a regional ITS architecture does not exist. The Turbo Architecture software can be obtained from McTrans by visiting their website at http://www-mctrans.ce.ufl.edu/featured/turbo/.

If you don't find what you need in the regional ITS architecture, then you should add the missing or changed items to your architecture subset and highlight them so it is clear what you changed. For example, if there is a system in your project that is not represented in the regional ITS architecture, add it to your architecture subset and highlight it. The highlighted changes serve two purposes: they allow you to move forward with an augmented architecture subset that you can use in the next steps of the process, and they provide the basis for your feedback for regional ITS architecture maintenance.

The second output of this step – feedback to the regional ITS architecture maintenance team – is just as important as the first output. Submit any recommended changes using the mechanism defined for your region in the regional ITS architecture maintenance plan.

4.1.4 Examples

The subset of the regional ITS architecture that is included in the project can be shown in a series of simple tables and/or a diagram from Turbo Architecture, as shown in Figure 9. This figure identifies the inventory elements and interfaces that will be implemented by a MaineDOT Dynamic Message Sign (DMS) project in which several signs will be installed in

Portland, Maine, along with a central control system with interfaces to a number of other centers. Functional requirements that are relevant to the project were also extracted, as shown in Table 2.

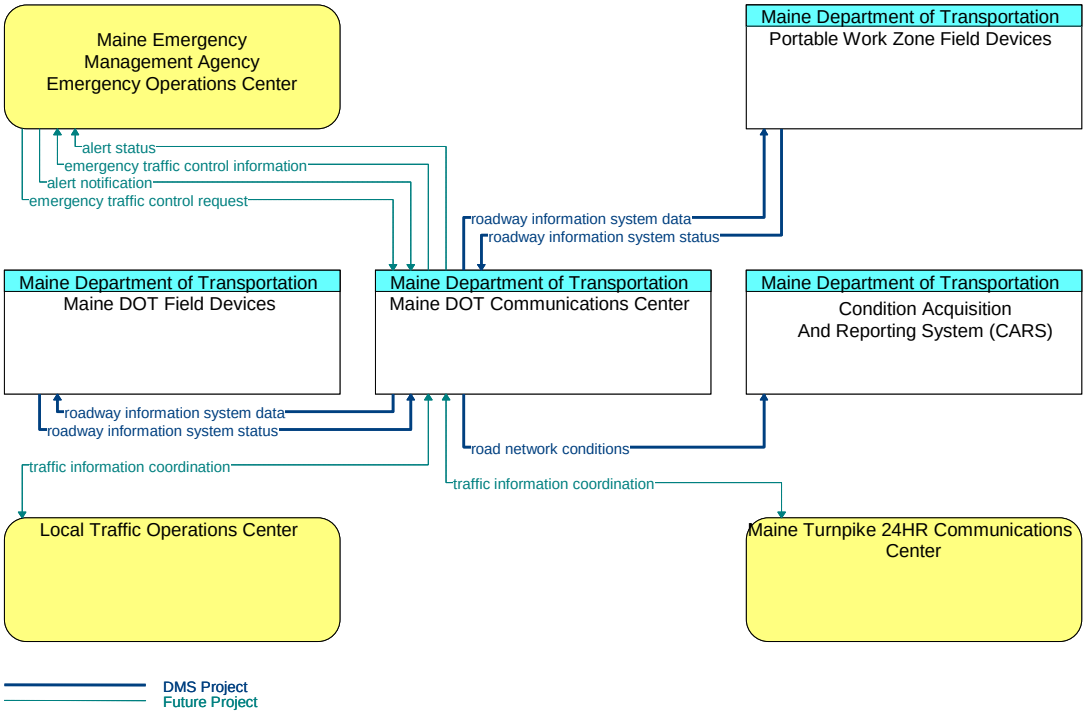
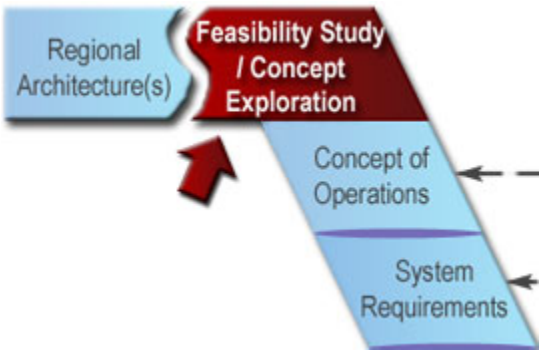


Figure 9: Example: MaineDOT DMS Project Architecture Subset

Table 2: MaineDOT DMS Project Functional Requirements (Partial List)

Element	Functional Area	ID	Requirement
MaineDOT Communications Center	TMC Traffic Information Dissemination	1	The Center shall remotely control DMS for dissemination of traffic and other information to drivers.
MaineDOT Communications Center	TMC Traffic Information Dissemination	3	The Center shall collect operational status for the driver information systems equipment (DMS, HAR, etc.).
MaineDOT Communications Center	TMC Traffic Information Dissemination	4	The Center shall collect fault data for the driver information systems equipment (DMS, HAR, etc.) for repair.

4.2 Feasibility Study/Concept Exploration



In this step: A business case is made for the project. Technical, economic, and political feasibility is assessed; benefits and costs are estimated; and key risks are identified. Alternative concepts for meeting the project's purpose and need are explored, and the superior concept is selected and justified using trade study techniques.

OBJECTIVES	▪ Identify superior, cost-effective concept, and document alternative concepts with rationale for selection ▪ Verify project feasibility and identify preliminary risks ▪ Garner management buy-in and necessary approvals for the project
INPUT ***Sources of Information***	▪ Project goals and objectives ▪ Project purpose and need ▪ Project scope/subset of the regional ITS architecture
PROCESS ***Key Activities***	▪ Define evaluation criteria ▪ Perform initial risk analysis ▪ Identify alternative concepts ▪ Evaluate alternatives ▪ Document results
OUTPUT ***Process Results***	▪ Feasibility study that identifies alternative concepts and makes the business case for the project and the selected concept
REVIEW ***Proceed only if you have:***	▪ Received approval on the feasibility study from project management, executive management, and controlling authorities, as required ▪ Reached consensus on the selected alternative

4.2.1 Overview

In this step, the proposed ITS project is assessed to determine whether it is technically, economically, and operationally viable. Major concept alternatives are considered, and the most viable option is selected and justified. While the concept exploration should be at a fairly high level at this early stage, enough technical detail must be included to show that the proposed concept is workable and realistic. The feasibility study provides a basis for understanding and agreement among project decision makers – project management, executive management, and any external agencies that must support the project, such as a regional planning commission.

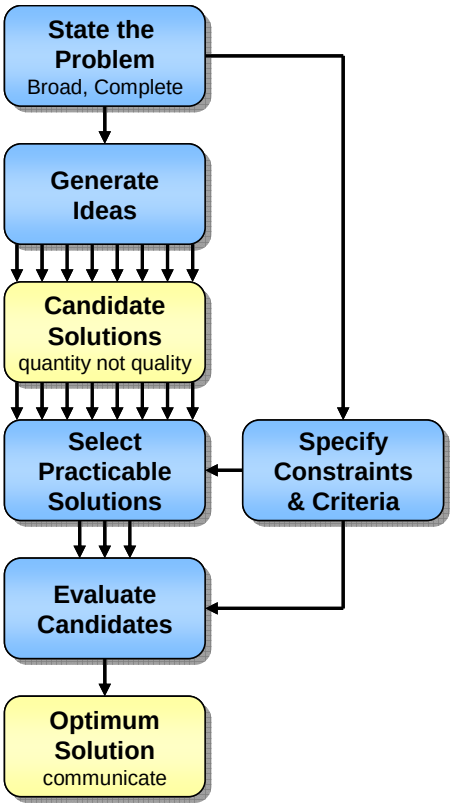


Figure 10: Concept Exploration Uses Basic Trade Study Techniques

The Rule/Policy requires the systems engineering analysis to include an analysis of alternative system configurations and technology options. The focus of this Rule/Policy requirement is on design decisions that are made later in the process, but a fundamental analysis of basic systems configurations is performed in this step.

It is easy to confuse the concept exploration that is performed in this step with the Concept of Operations that is developed in the next step. *Concept exploration* is a broad assessment of fundamentally different alternatives – for example, a new electronic toll facility versus additional conventional lanes. The alternatives would have dramatically different concepts of operations, so it is important to select a proposed concept before developing a Concept of Operations. Different alternatives may also have different regional ITS architecture mappings, so this step may iterate with the previous regional ITS architecture step.

The process is driven by the project vision, goals, and objectives, and by the needs for the project that were identified through the transportation planning process. It starts by identifying a broad range of potential concepts that satisfy the project need(s). The concepts are compared relative to measures that assess the relative benefits, costs, and risks of each alternative. Project stakeholders must be involved to establish the evaluation criteria, verify that all viable alternative concepts are considered, and make sure there is consensus on the selected alternative. The recommendations provide a documented rationale for the selected project approach and an assessment of its feasibility. The process is identical to a feasibility study done for large roadway and transit projects.

The alternatives analysis that is performed during a feasibility analysis uses a basic trade study technique, shown in Figure 10, that will be repeated many times during the project life cycle. At this early concept exploration step, the alternatives are fundamental choices, such as to maintain the existing facility ("do nothing"), build a new road, or add ITS technology to the existing facility. During design, the alternatives are design decisions, such as whether signs should be located at location A, B, or C. During construction, alternatives may have to do with optimizing closures while the work is performed. At each step, a set of alternatives is identified and analyzed from technical, economic, and operational perspectives.

A feasibility study should be conducted only when a broad analysis is needed before the commitment of development resources. Some states require a feasibility study for certain ITS projects. A feasibility study is typically not required for smaller, incremental ITS projects where there are not fundamentally different approaches for implementation and where feasibility is not in question – for example, a project that adds DMS to an existing system. In other cases, a broad exploration of alternatives is not warranted but a cost-benefit study is needed to make the business case for the project.

4.2.2 Key Activities

Here at the very beginning of project development, the unknowns will certainly outnumber the knowns. Without a Concept of Operations or requirements, many assumptions will have to be made. It is important to educate the group performing this assessment on the concept exploration process and to set a schedule – otherwise, this stage could be an open-ended process since there's always something new over the horizon. The process activities are:

- **Define evaluation criteria –** Work with stakeholders to clearly define the problem or opportunity that is to be addressed by this project, elaborating on the identified purpose and the need(s), goals, and objectives as necessary. These elements were originally defined through the transportation planning and programming process that identified the project. The inputs may be augmented by the portion of the regional ITS architecture that was identified in the previous step.

  Based on the statement of the problem, establish cost constraints and any other constraints that will be used to limit the acceptable alternatives. Determine how success will be measured – the degree to which the project will solve the stated problem or realize the identified opportunity. These measures should be included in the criteria that will be used to evaluate the alternative concepts. Also, do a preliminary risk analysis to identify issues and obstacles that may affect the project, and develop evaluation criteria that will measure the sensitivity of each candidate solution to each of the risks.

  It is a good idea to define evaluation criteria before alternative concepts are enumerated. By developing the criteria first, you reduce the risk of intuitively settling on an alternative and then subconsciously biasing the criteria toward the preferred alternative. It is important to develop the criteria so that they are not preferential to one of the concepts.

- **Identify alternative concepts –** Identify a broad range of potential concepts that will solve the identified problem. The alternative concepts should be defined in clear, technology-independent terms that all affected organizations will understand. At least two (and preferably more) alternatives should be defined. One alternative should always be "do nothing", which provides a basis for comparison with the other alternatives.

  If you find that you are identifying specific products or vendors as the alternative solutions, you are being too specific. A trade comparison of products or vendors occurs much later in the process based on defined requirements during design. The alternatives here should be high-level concepts – for example, instrumentation with traffic detectors versus use of traffic probes to support traffic data collection for a corridor. Alternatives may also reflect life-cycle options, such as leased versus owned equipment, contracted versus in-house staffing, etc. You may have to establish a basic architecture and a minimal strawman design to support the analysis, but do no more than is necessary to support the evaluation.

  A common pitfall in developing a concept exploration or any trade study comparison is the premature selection of an alternative early in the study process. Be sure to keep an open mind and spend enough time on all viable options. If only one of the alternatives is

defined in detail in a concept exploration, it creates the appearance that the other alternatives were not earnestly considered or explored.

- **Evaluate alternatives –** Perform a systematic analysis of the alternatives, applying the same criteria to each. The evaluation should measure the technical benefit, the economic impact, the operational feasibility, the life-cycle costs, and the risks associated with each alternative. A cost-benefit analysis is a key aspect of the evaluation.

  A number of tools support cost-benefit analysis for ITS projects:

  - The ITS Costs database contains estimates that can be used for policy analysis and cost-benefit analysis. It contains unit cost estimates for more than 200 ITS technologies as well as system costs for selected ITS deployments. (The unit cost database is available online and as an Excel spreadsheet at http://www.itscosts.its.dot.gov.)

  - The ITS Benefits database contains information regarding the impacts of ITS projects on the operation of the surface transportation system. The ITS Benefits website provides an online and Excel spreadsheet version of this database as well as several other documents pertaining to ITS benefits. (See http://www.itsbenefits.its.dot.gov.)

  - The ITS Deployment Analysis System (IDAS) is software developed by the Federal Highway Administration that can be used to estimate the benefits and costs of ITS investments, which are either alternatives to or enhancements of traditional highway and transit infrastructure. IDAS can currently predict relative costs and benefits for more than 60 types of ITS investments. (See http://idas.camsys.com/.)

  - SCRITS (SCReening for ITS) is a spreadsheet analysis tool for estimating the user benefits of ITS. It is intended as a sketch- or screening-level analysis tool for allowing practitioners to obtain an initial indication of the possible benefits of various ITS applications. (For more information, see http://www.fhwa.dot.gov/steam/scrits.htm.)

  While it is best to do a complete analysis of every alternative, sometimes the sheer number of alternatives makes this thorough approach impractical. One common practice is to apply the evaluation criteria in stages, weeding out the alternatives that don't meet the fundamental criteria so that the more detailed, time-consuming analysis is performed on only a few of the most viable alternatives. The evaluation should be validated by reviewing the analysis with stakeholders who may have reasonable objections to certain assumptions and alternatives.

- **Document results –** The feasibility study is documented to provide an overview of the project and determine if feasible solutions exist that should be funded and implemented. The study document should include sufficient information to support the decision makers who will make funding decisions based on the study. The cost-benefit analysis may be included as part of the feasibility study document or included in a separate cost-benefit analysis document, depending on the nature of the project and state/local documentation requirements.

  Remember your audience when writing a feasibility study; this study makes a business case primarily for a management audience. Any feature of the study that prevents the reader from assimilating the costs and benefits and the associated risks of each alternative solution as briefly, completely, and painlessly as possible reduces the effectiveness of the study for the audience.

Several review cycles may be required for the feasibility study. First, the document should be circulated among the project team to make sure that there is buy-in. Then an updated draft should be distributed to internal management and other organizations for approval.

4.2.3 Outputs

The feasibility study establishes the business case for investment in a project by defining the reasons for undertaking the project and analyzing its costs and benefits. Different organizations and different projects will have different requirements, but a feasibility study should contain, at a minimum, the following:

1. A description of the problem or opportunity that the project is intended to address.
2. The project objectives that must be achieved for an alternative to be an effective response to the problem or opportunity, and the evaluation criteria that were used.
3. Economic and risk analyses of each of the alternatives that meet the established objectives, and the reasons for rejecting the alternatives that were not selected.
4. A summary description of the selected alternative, including the major system features and resources that will be used.
5. An economic analysis of the funding sources, the life-cycle costs and benefits of the project, and the life-cycle costs and benefits of the current method of operation.

4.2.4 Examples

Identification of Alternatives – Transportation Planning Studies

Feasibility studies that examine alternative concepts are frequently done for large transportation projects as part of corridor studies, major investment studies, and environmental analysis reports. The ITS option(s) in these studies often compete with traditional capital improvement options; hybrid options, which include a mix of technology and traditional capital improvements, are also considered.

For example, a congested corridor in Collin County, Texas, was the subject of a feasibility study report (FSR)[9] that was prepared by representatives from the North Central Texas Council of Governments and affected agencies. This FSR examined the following alternatives: (1) do nothing, (2) build a new freeway, (3) build a toll road with electronic collection (two alternatives), and (4) build managed lanes. One summary table that compared the traffic volumes supported by the different alternatives is shown in Table 3. Supported traffic volumes, estimated capital costs, and potential revenue generation were used to compare the alternatives. The analysis favored the electronic toll alternatives.

Broad alternatives analyses like these are included in many planning studies.

Table 3: Comparison of Alternatives – Supported Traffic Volumes for 2025

	Alternative #1 (No-Build)	Alternative #2 (Freeway)	Alternative #3 (Tolls east of DNT)	Alternative #4 (Tolls east of Hillcrest)	Alternative #5 (Managed Lanes)
DNT to Hillcrest	118,171	161,069	139,565	156,920	149,921
Hillcrest to Custer	61,767*	146,283	118,835	121,604	144,736
Custer to Stacy	47,379*	120,694	72,280	72,297	115,304
Stacy to US 75	46,607*	94,198	58,762	59,369	92,880

[9]See http://www.nctcog.org/trans/corridor/SH121/.

Cost-Benefit Analysis

Minnesota DOT developed a guidance document[10] for cost-benefit analysis in 2005 that includes several illustrative examples. Generally, higher-level graphics that visually compare the costs and benefits of the alternatives, like the one shown in Figure 11, are used in the body of the cost-benefit analysis. More detailed computation that supports high-level graphics, like the table reproduced in Table 4, is included in appendices.

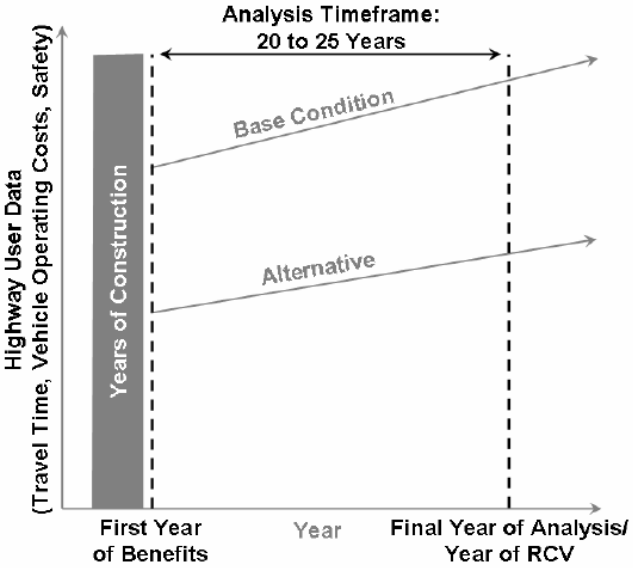


Figure 11: Example of High-Level Economic Comparison of Alternatives

Table 4: Example of Alternatives Benefit Estimation

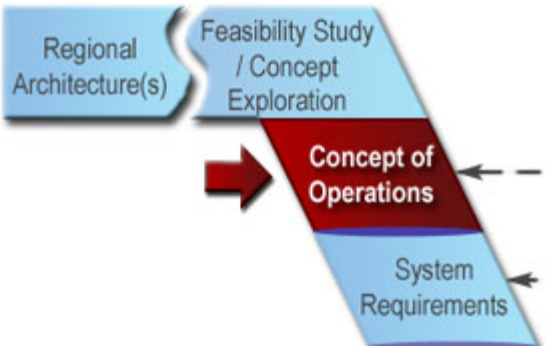
Column:	A	B		C	D	E
	Year	DAILY VHT [a] (veh-hours/day)		DIFFERENCE IN VHT (veh-hours/day)	Annual Savings in Constant Dollars [b]	Present Value of Savings (dollars) [c]
		Base Condition	Alternative			
2011	**1**	**2,643,108**	**2,635,761**	**7,348**	**$28,432,953**	**$22,348,044**
2012	2	2,687,035	2,678,788	8,247	$31,914,451	$24,236,203
2013	3	2,731,691	2,722,517	9,175	$35,502,929	$26,049,598
2014	4	2,777,090	2,766,960	10,130	$39,200,965	$27,790,302
2015	5	2,823,243	2,812,128	11,115	$43,011,196	$29,460,334
2016	6	2,870,164	2,858,034	12,129	$46,936,313	$31,061,668
2017	7	2,917,864	2,904,689	13,174	$50,979,070	$32,596,229
2018	8	2,966,356	2,952,106	14,250	$55,142,275	$34,065,893
2019	9	3,015,655	3,000,297	15,358	$59,428,800	$35,472,493
2020	10	3,065,773	3,049,275	16,498	$63,841,579	$36,817,816
2021	11	3,116,724	3,099,052	17,672	$68,383,607	$38,103,604
2022	12	3,168,521	3,149,642	18,880	$73,057,945	$39,331,558
2023	13	3,221,180	3,201,057	20,123	$77,867,720	$40,503,338
2024	14	3,274,714	3,253,312	21,402	$82,816,126	$41,620,560
2025	15	3,329,137	3,306,420	22,717	$87,906,424	$42,684,802
2026	16	3,384,465	3,360,395	24,070	$93,141,947	$43,697,603
2027	17	3,440,712	3,415,251	25,461	$98,526,098	$44,660,465
2028	18	3,497,894	3,471,002	26,892	$104,062,352	$45,574,849
2029	19	3,556,026	3,527,664	28,363	$109,754,260	$46,442,185
2030	**20**	**3,615,125**	**3,585,250**	**29,875**	**$115,605,448**	**$47,263,862**
	Total Benefits During Project Life (2011 - 2030)					$729,781,406

(a) All daily vehicle hours of travel (VHT) data was derived using a transportation forecast model of this region.
(b) Using the composite cost per hour (includes both auto and truck).
(c) Present value of savings during the life of the project in terms of 2004 dollars.

[10] See http://www.oim.dot.state.mn.us/EASS/

4.3 Concept of Operations

In this step: The project stakeholders reach a shared understanding of the system to be developed and how it will be operated and maintained. The Concept of Operations (ConOps) is documented to provide a foundation for more detailed analyses that will follow. It will be the basis for the system requirements that are developed in the next step.



OBJECTIVES	▪ High-level identification of user needs and system capabilities in terms that all project stakeholders can understand ▪ Stakeholder agreement on interrelationships and roles and responsibilities for the system ▪ Shared understanding by system owners, operators, maintainers, and developers on the who, what, why, where, and how of the system ▪ Agreement on key performance measures and a basic plan for how the system will be validated at the end of project development
INPUT ***Sources of Information***	▪ Stakeholder lists, roles and responsibilities, and other components from the regional ITS architecture ▪ Recommended concept and feasibility study from the previous step ▪ Broad stakeholder input and review
PROCESS ***Key Activities***	▪ Identify the stakeholders associated with the system/project ▪ Define the core group responsible for creating the Concept of Operations ▪ Develop an initial Concept of Operations, review with broader group of stakeholders, and iterate ▪ Define stakeholder needs ▪ Create a System Validation Plan
OUTPUT ***Process Results***	▪ Concept of Operations describing the who, what, why, where, and how of the project/system, including stakeholder needs and constraints ▪ System Validation Plan defining the approach that will be used to validate the project delivery
REVIEW ***Proceed only if you have:***	▪ Received approval on the Concept of Operations from each stakeholder organization ▪ Received approval on the System Validation Plan from each stakeholder organization

4.3.1 Overview

The Concept of Operations (ConOps) is a foundation document that frames the overall system and sets the technical course for the project. Its purpose is to clearly convey a high-level view of the system to be developed that each stakeholder can understand. A good ConOps answers who, what, where, when, why, and how questions about the project from the viewpoint of each stakeholder, as shown in Figure 12.

- Who – Who are the stakeholders involved with the system?
- What – What are the elements and the high-level capabilities of the system?
- Where – What is the geographic and physical extent of the system?
- When – What is the sequence of activities that will be performed?
- Why – What is the problem or opportunity addressed by the system?
- How – How will the system be developed, operated, and maintained?

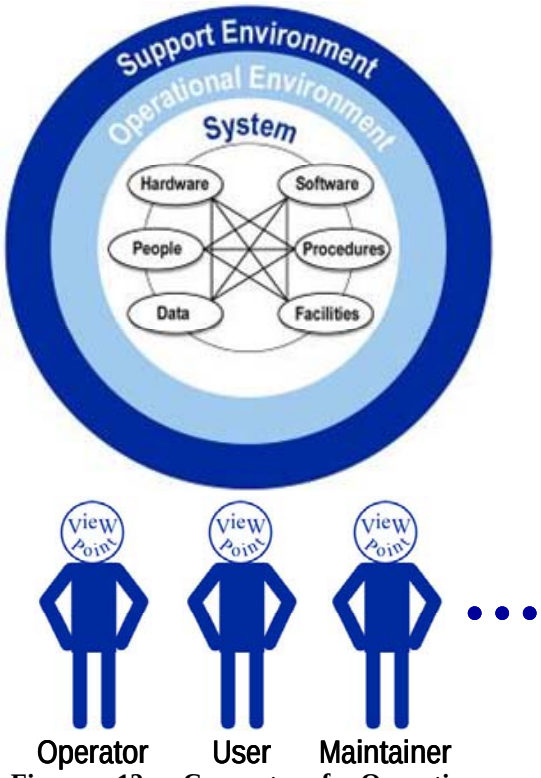


Figure 12: Concept of Operations
(Adapted from ANSI/AIAA-G-043-1992)

In ITS, we draw a distinction between an *Operational Concept*, which is the high-level description of roles and responsibilities that is included in the regional ITS architecture, and a *Concept of Operations*, which is the more detailed, multifaceted document described in this section.

Don't assume that a new ConOps is required for every ITS project. A single system-level ConOps can support many ITS projects that incrementally implement and extend a system. For example, a ConOps may be developed for a large transportation management system. This system may be implemented and expanded with numerous ITS projects over several years. Once the ConOps is developed, it may be reviewed and used with relatively minor updates for each of the projects that incrementally implement the transportation management system.

4.3.2 Key Activities

Although there is no single recipe for developing a ConOps, successful efforts will include a few key activities:

- **Identify the stakeholders associated with the system/project –** Systems engineering in general, and this effort in particular, require broad participation from the project's stakeholders. One of the first steps in developing a ConOps is to make sure that all the stakeholders involved in or impacted by the project – owners, operators, maintainers, users, and so forth – are identified and involved. You can start with the stakeholder list from the regional ITS architecture and then expand it to identify the more specific organizations – divisions and departments – that should be involved. One of the most

effective ways to involve the stakeholders is to create an *integrated product team (IPT)* that brings together the necessary expertise and provides a forum for all project stakeholders.

- **Define the core group responsible for creating the ConOps –** Although broad involvement is critical, you can't have 20 people on your writing team. Select a few individuals who are responsible for capturing and documenting the vision of the broader group. Depending on the size of the project and staff capabilities, this team might include a consultant or staff members with knowledge of the project and requisite writing and communications skills.

If you hire a consultant, don't assume that is the end of your responsibility for ConOps development. The stakeholders are the foremost experts on their needs and must be materially involved in the ConOps development. The consultant can provide technical expertise on what should be in a ConOps, facilitate the meetings and outreach activities, prepare the document, and coordinate the review, but the stakeholders' concept should be documented in the end. The stakeholders should consider the ConOps *their* document, not the consultant's document.

The best person to write the ConOps may not be the foremost technical expert on the proposed system. Stakeholder outreach, consensus building, and the ability to understand and clearly document the larger picture are key.

- **Develop the initial ConOps, review it with the broader group of stakeholders, and iterate –** Incrementally create the ConOps, review relevant portions with stakeholders, and adjust the concept as necessary to get buy-in. All stakeholders do not have to agree on every aspect of the project, but all must feel that they are achieving their major goals for the project.

  Portions of the ConOps can often be created from existing documents. For example, the regional ITS architecture identifies stakeholder roles and responsibilities that can be used. A feasibility study, project study report, or other preliminary study documentation may provide even more relevant information. A project application form used to support project programming will normally include goals, objectives, and other information that should be reflected in the ConOps for continuity.

Operational scenarios are an excellent way to work with the stakeholders to define a ConOps. Scenarios associated with a major incident, a work zone, or another project-specific situation provide a vivid context for a discussion of the system's operation. It is common practice to define several scenarios that cover normal system operation (the "sunny day" scenario) as well as various fault-and-failure scenarios.

- **Define stakeholder needs –** Actually, this is a key purpose of the ConOps – to capture a clear definition of the stakeholders' needs and constraints that will support system requirements development in the next step. Interviews, workshops, and surveys are some of the techniques that are used to perform this activity. The ConOps is a great tool for defining needs since it forces the stakeholders to think about the way the system will behave and how it will interact with users and other systems. The operational scenarios in the ConOps are among the best tools for discovering needs. The list of needs that is generated should be prioritized by the stakeholders. Once they start to compare and rank the needs, they will discover that some of their "needs" are really "wants" or "nice-to-haves".

- **Create a System Validation Plan –** The initial performance measures that are identified in the ConOps provide a foundation for the System Validation Plan. While expectations

for the system will change over time, the performance measures outlined in the ConOps force early consideration and agreement of how system performance and project success will be measured. Examples of performance measures include travel time, incident duration, and level of service. You should define a set of performance measures that will assess the effectiveness of the system you are implementing.

A System Validation Plan is prepared that defines the consensus validation approach and performance measures. As with the ConOps, all affected stakeholder organizations should formally approve the System Validation Plan at this early stage so that downstream, all will agree on when they can "declare victory" that the new system is the right system. The plan will be finalized during system validation (see Section 4.9.2).

4.3.3 Output

The ConOps should be an approachable document that is relevant to all project stakeholders, including system operators, maintainers, developers, owners/decision makers, and other transportation professionals. The art of creating a good ConOps lies in using natural language and supporting graphics so that it is accessible to all while being technically precise enough to provide a traceable foundation for the requirements document and the System Validation Plan.

The ConOps is not a requirements document that lists the detailed, testable requirements for the system, nor is it a design document that specifies the technical design or technologies to be used. Resist the temptation to predetermine the solution in the ConOps – you should not unnecessarily preclude viable options at this early step. You also want to "keep it simple" and refrain from using formalized, highly structured English that is more suitable for the requirements and design specifications that follow.

Done right, the ConOps will be a living document that can be revised and amended so that it continues to reflect how the system is really operated. Later in the life cycle, an up-to-date ConOps can be used to define changes and upgrades to the system.

Two different industry standards provide suggested outlines for Concepts of Operations: ANSI/AIAA-G-043-1992 and IEEE Std 1362-1998, as shown in Figure 13. Both outlines include similar content, although the structure of the IEEE outline lends itself more to incremental projects that are upgrading an existing system or capability. The ANSI/AIAA outline is focused on the system to be developed, so it may lend itself more to new system developments where there is no predecessor system. Successful ConOps have been developed using both outlines. Obtain a copy of both, and make your own choice if you need to develop a ConOps.

ANSI/AIAA-G-043 Outline

1. Scope
2. Referenced Documents
3. User-Oriented Operational Description
4. Operational Needs
5. System Overview
6. Operational Environment
7. Support Environment
8. Operational Scenarios

IEEE 1362 Outline

1. Scope
2. Referenced Documents
3. The Current System or Situation
4. Justification for and Nature of Changes
5. Concepts for the Proposed System
6. Operational Scenarios
7. Summary of Impacts
8. Analysis of the Proposed System

Figure 13: Industry-Standard Outlines for Concept of Operations

Graphics should be used to highlight key points in the ConOps. At a minimum, a system diagram that identifies the key elements and interfaces and clearly defines the scope of the

project should be included. Tables and graphics can also be a very effective way to show key goals and objectives, operational scenarios, etc.

The Rule/Policy requires identification of participating agency roles and responsibilities as part of the systems engineering analysis for ITS projects. It also requires that the procedures and resources necessary for operations and management of the system be defined. These elements are initially defined and documented for the project as part of the ConOps. In the ANSI/AIAA standard outline, most of these elements fit under Chapter 3 (User-Oriented Operational Description). In the IEEE outline, the current system information is included in Chapter 3 and the proposed system information is in Chapter 5.

The System Validation Plan that is created during this step should describe how the final system will be measured to determine whether or not it meets the original intent of the stakeholders as described in the ConOps. (For further details and examples, see Section 4.9.)

4.3.4 Examples

Many Concepts of Operations have been generated for all types of ITS projects in the last five years. Excerpts from a few examples are included here to show some of the ways that key elements of the ConOps have been documented for ITS projects following the sequence from the ANSI/AAIA outline.

User-Oriented Operational Description (Roles and Responsibilities)

Typically, roles and responsibilities are documented as a list or in tabular form. Table 5 is an excerpt of a table from the California Advanced Transportation Management System (CATMS) ConOps that is structured to show shared responsibilities and to highlight coordination points between the different system stakeholders. This early documentation of "who does what" grabs the stakeholders' attention and supports development of system requirements and operational agreements and procedures in future steps.

Table 5: Roles and Responsibilities (Excerpt from CATMS Concept of Operations)

L = Lead **S = Support** **Position**	**TMC Manager**	**TMC Operator**	**Maintenance Dispatch**	**TMS Engineer**	**IT**	**HQ Engineer**	**District Deputy of Operations**	**Program Manager**	**Director**
Contacting Executives	L	S					L	S	S
After incident briefing and analysis	L	S	S	S		S	S		
Signal timing adjustments				L					
Field data detection				L					
Weather monitoring		L	S						
Road condition monitoring and communication	S	L	S						
Special event traffic monitoring	S	L	S						
Workzone lane closure logging	S	L	S						
Monitor workzone duration and congestion	S	S	S						
Incident response management	L	L	S						
News monitoring	S	L							

System Overview

The system overview is typically supported by one or more diagrams that show the scope, major elements, and interrelationships of the system. Many types of diagrams can be used, from simple block diagrams to executive-level graphics-rich diagrams. Figure 14 is an example of a high-level graphic that includes basic process flow information, roles and responsibilities, and interfaces, providing an "at a glance" overview of the major facets of the system.

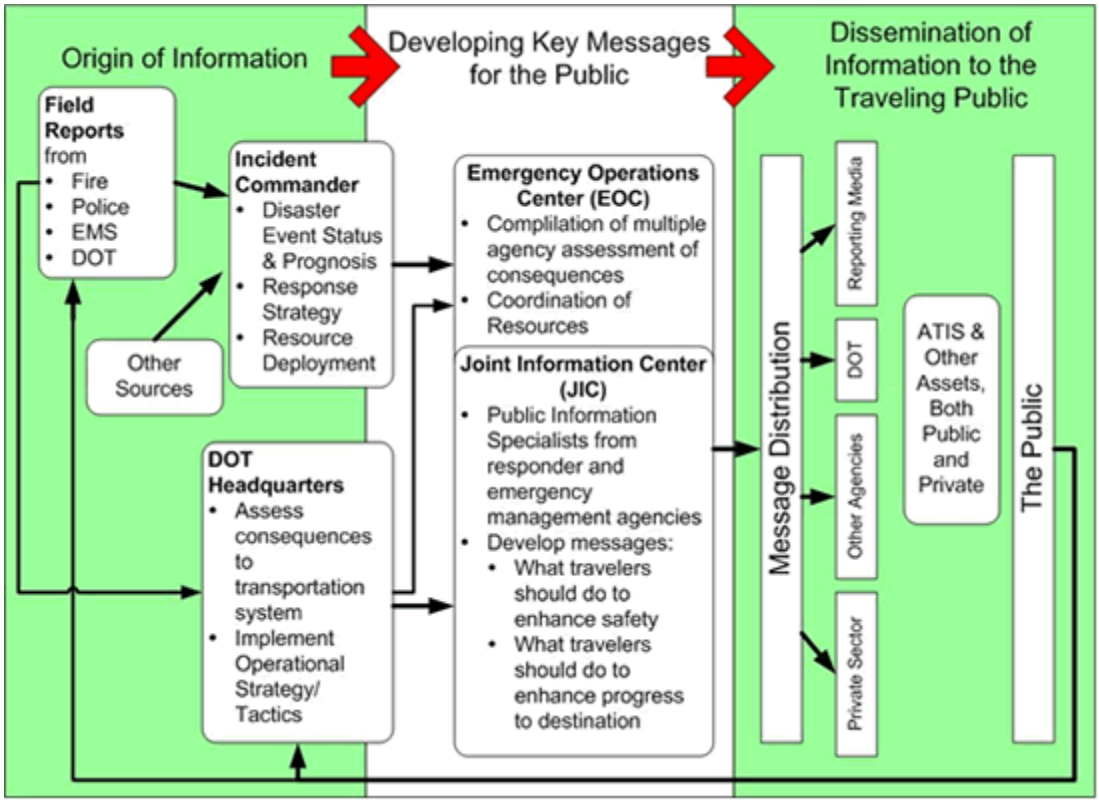


Figure 14: Example of System Overview Graphic (from *Communicating with the Public Using ATIS During Disasters Concept of Operations*)

Operational Scenarios

In operational scenarios, the ConOps takes the perspective of each of the stakeholders as different scenarios unfold that illustrate major system capabilities and stakeholder interactions under normal and stressed (e.g., failure mode) circumstances. The stakeholders walk through the scenario and document what the agencies and system would do at each step.

Figure 15 shows an example of a scenario that includes some realistic detail that help stakeholders immerse themselves in the scenario and visualize system operation. This is one of five scenarios that were developed for the City of Lincoln StarTRAN AVL system to show the major system capabilities and the interactions between the AVL system and its users and other interfacing systems.

Marcel, a StarTran bus operator, usually begins his work shift with administrative activities. After receiving supervisory direction, he boards the bus and prepares the AVL system. He begins by logging into the system.

The system then prompts Marcel for the route to be followed. He enters the planned route number, and the AVL system retrieves the appropriate route and schedule information from the AVL system server. The bus' AVL system then asks Marcel to verify the appropriate route and schedule information were properly retrieved.

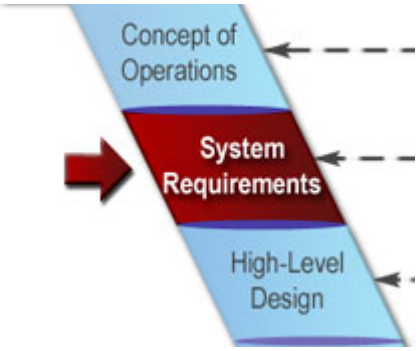
Once he provides verification, the bus' head sign is automatically updated to reflect the appropriate route information. The fare payment schedule is automatically adjusted to reflect the verified route, modified as necessary by the system clock to reflect any applicable time-differential rates.

The system then loads the appropriate bus stop announcements for the chosen route. These prerecorded announcements are consistent regardless whether Marcel or another bus operator is driving the route, and have been verified as ADA compliant. These announcements are then broadcast at the appropriate bus stop throughout the route.

Figure 15: Operational Scenario Description[11]

[11]From "StarTran Automated Vehicle Location System Concept of Operations", StarTran and City of Lincoln, NE.

4.4 System Requirements



In this step: The stakeholder needs identified in the Concept of Operations are reviewed, analyzed, and transformed into verifiable requirements that define *what* the system will do but not *how* the system will do it. Working closely with stakeholders, the requirements are elicited, analyzed, validated, documented, and baselined.

OBJECTIVES	▪ Develop a validated set of system requirements that meet the stakeholders' needs
INPUT ***Sources of Information***	▪ Concept of Operations (stakeholder needs) ▪ Functional requirements, interfaces, and applicable ITS standards from the regional ITS architecture ▪ Applicable statutes, regulations, and policies ▪ Constraints (required legacy system interfaces, hardware/software platform, etc.)
PROCESS ***Key Activities***	▪ Elicit requirements ▪ Analyze requirements ▪ Document requirements ▪ Validate requirements ▪ Manage requirements ▪ Create a System Verification Plan ▪ Create a System Acceptance Plan
OUTPUT ***Process Results***	▪ System Requirements document ▪ System Verification Plan ▪ Traceability Matrix ▪ System Acceptance Plan
REVIEW ***Proceed only if you have:***	▪ Received approval on the System Requirements document from each stakeholder organization, including those that will deploy, test, install, operate, and maintain the new system ▪ Received approval on the System Verification Plan from the project sponsor, the test team, and other stakeholder organizations ▪ Received approval on the System Acceptance Plan from the project sponsor, the Operations & Maintenance (O&M) team, and other stakeholder organizations

4.4.1 Overview

One of the most important attributes of a successful project is a clear statement of requirements that meet the stakeholders' needs. Unfortunately, creating a clear statement of requirements is often much easier said than done. The initial list of stakeholder needs that are collected will normally be a jumble of requirements, wish lists, technology preferences, and other disconnected thoughts and ideas. A lot of analysis must be performed to develop a good set of requirements from this initial list.

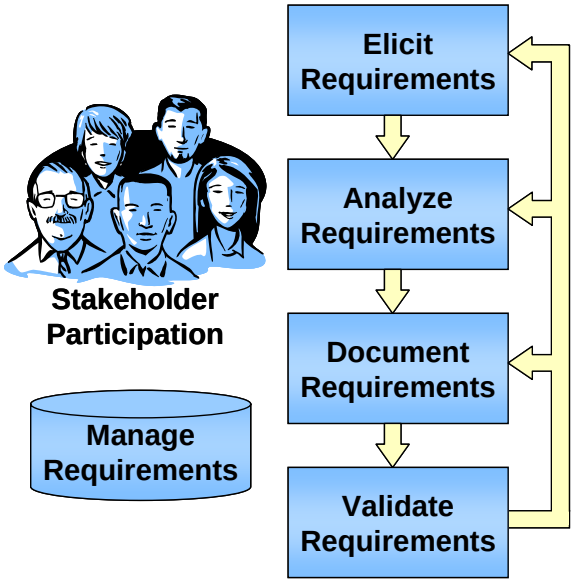


Figure 16: Requirements Engineering Activities

EIA-632[12] defines *requirement* as "something that governs what, how well, and under what conditions a product will achieve a given purpose." This is a good definition because it touches on the different types of requirements that must be defined for a project. Functional requirements define "what" the system must do, performance requirements define "how well" the system must perform its functions, and a variety of other requirements define "under what conditions" the system must operate. *Requirements engineering* covers all of the activities needed to define and manage requirements that are shown in Figure 16.

Specify What, Not How. Be sure to keep the definition of a requirement in mind as you develop your system requirements. Many requirements documents contain statements that are not requirements. One of the most common pitfalls is to jump to a design solution and then write "requirements" that define how the system will accomplish its functions. Specify what the system will do in the system requirements, and save how the system will do it for the system design step.

It is important to involve stakeholders in requirements development. Stakeholders may not have experience in writing requirements statements, but they are the foremost experts concerning their own requirements. The project requirements ultimately are the primary formal communication from the system stakeholders to the system developer. The project will be successful only if the requirements adequately represent stakeholders' needs and are written so they will be interpreted correctly by the developer.

In the effort to get stakeholders involved, make sure you don't sour them on the project by making unreasonable demands on their time or putting them in situations where they can't contribute. Many nontechnical users have been subjected to stacks of detailed technical outputs that they can't productively review. Sooner or later, the user will wave the white flag in this situation and become unresponsive. You must (1) pick your stakeholders carefully and (2) make participation as focused and productive as possible.

[12]EIA-632 is the Electronics Industry Association Standard "Processes for Engineering a System".

The Requirements step is an important one that you shouldn't skimp on. Every ITS project should have a documented set of requirements that are approved and baselined. Of course, this doesn't mean that a new requirements specification must be written from scratch for every project. Projects that enhance or extend an existing system should start with the existing system requirements. This doesn't have to be a particularly large document for smaller ITS projects. The system requirements specification for a recent website development project was less than 20 pages.

4.4.2 Key Activities

There isn't one "right" approach for requirements development. Different organizations develop requirements in different ways. Even in the same organization, the requirements development process for a small ITS project can be much less formal than the process for the largest, most complex ITS projects. The differences are primarily in the details and in the level of formality. All requirements development processes should involve elicitation, analysis, documentation, validation, and management activities. Note that each of these activities is highly iterative. In the course of a day, a systems engineer may do a bit of each of the activities as a new requirement is identified, refined, and documented.

- **Elicit requirements –** Building on the stakeholders' needs and other inputs, such as the functional requirements from the regional ITS architecture and any relevant statutes, regulations, or policies, define a strawman set of system requirements and review and expand on them, working closely with the project stakeholders. There are many different elicitation techniques that can be used, including interviews, scenarios (see discussion under Concept of Operations in Section 4.3), prototypes, facilitated meetings, surveys, and observations. These techniques can be used in combination to discover the stakeholders' requirements.

Elicit and *elicitation* are words you may not run into every day. *Elicit* means to draw forth or to evoke a response. This is the perfect word to use in this case because you will have to do some work to draw out the requirements from the stakeholders and any existing documentation. More work is implied by "elicit requirements" than if we said "collect requirements" or even "identify requirements", and this is intended.

Make sure that you have the right stakeholders involved. This means not only the right organizations but also the right individuals within them. For example, it isn't enough to engage someone from the maintenance organization – it should be an electrical maintenance person who has experience with ITS equipment maintenance for ITS projects. Furthermore, as we move through the steps in the process and the products become more technical, different stakeholders may be involved. Managers may be more involved in the Concept of Operations, while technical staff will be more involved in review of the system requirements and high-level design. Finding individuals with the right combination of knowledge of current operations, vision of the future system, and time to invest in supporting requirements development is one of the key early challenges in any requirements development effort.

There are many techniques for working with stakeholders to get to the fundamental requirements of the system. The Florida SEMP[13] highlights one of the best and simplest techniques – the "Five Whys" – that was popularized by Toyota in the 1970s. Using the Five Whys technique, you look at an initially stated need and ask "Why?" repeatedly, not

[13]The Florida Systems Engineering Management Plan is available at http://www.floridaits.com/SEMP/Index.htm.

unlike a curious four-year-old, until you find the real underlying requirements. The dialog in Table 6 is an example that is based on an actual conversation.

Table 6: The "Five Whys" Technique in Action

Stakeholder	Systems Engineer
I need irrigation channels on my keyboard.	Why?
I occasionally spill coffee on the keyboard.	Why?
I need to have three or four manuals open to operate the system and the coffee just gets knocked over.	Why do you need to have three or four manuals open?
…	…
The dialog continues as the systems engineer discovers several different underlying needs that will drive environmental requirements, human factors/workspace requirements, and user interface requirements, all by pursuing the initial stated need for "irrigation channels".	

Of course, you sometimes need to direct the conversation by asking more than "why" to use this technique effectively. In the example, the conversation could easily have veered off to a discussion of the user's love for Starbucks coffee. Five iterations is a good rule of thumb, but it may take fewer or more iterations – the idea is to be persistent until you get to the core issues. Note also that the dialog can be internal – the stakeholder could have sat down and asked herself "Why", using the same technique to get at her underlying needs.

As you gather the requirements, be sure to look beyond the operational requirements for the system and cover the complete life cycle (system development, deployment, training, transition, operations and maintenance, upgrades, and retirement) as well as requirements such as security and safety. More than one ITS project has failed because the security requirements of public safety stakeholders were not captured and reflected in the ITS project requirements. A good system requirements template can be used as a checklist to help ensure that all types of requirements are considered.

The best way to start writing requirements is to use just two words: a verb and a noun. For example, the user requirement "monitor road weather conditions" would yield system requirements such as "shall detect ice", "shall monitor wind speed", and "shall monitor pavement temperature". Performance requirements would define the different kinds of ice conditions and the range of wind speeds and pavement temperatures.

- **Analyze requirements –** The requirements are analyzed in detail, and the stakeholders negotiate to prioritize them. This is where the requirements are cleaned up: conflicts are resolved, gaps are identified and addressed, ambiguity and redundancy are removed, and the requirements are organized and decomposed into more detailed requirements. Several levels of requirements are developed, providing sufficient granularity so that they can be allocated to individual subsystems and components in the next step, high-level design.

Requirements are normally defined in a requirements hierarchy in which the highest-level "parent" requirements are supported by more detailed "child" requirements. A hierarchy allows you to start with high-level requirements and work your way down to the details. The highest-level requirements should trace to stakeholder needs in the Concept of Operations. A hierarchy is a useful organizational structure that makes it easier to write and review requirements and to manage the requirements development activity. An example of a requirements hierarchy is given in Figure 17.

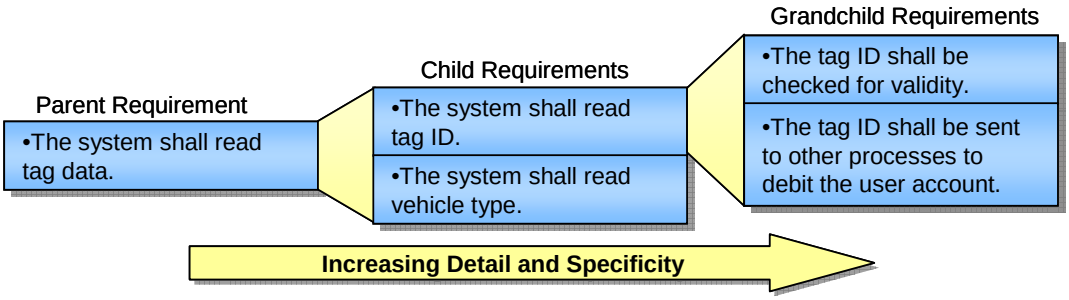


Figure 17: Example of Hierarchy of High-Level and Detailed Requirements

For larger systems, it can be very difficult to "get your arms around" all of the requirements. Requirements modeling tools provide a graphic way to define requirements so that they are easier to understand and analyze. These tools are particularly useful for more complex ITS projects. There are numerous requirements modeling tools and techniques available that can help you model the system as part of the analysis process. INCOSE maintains a data repository of available modeling tools that is available on its website[14].

A *model* is a representation of something else. There are physical models, like the scale model of a train, and more abstract models, like an architectural plan for a new building. Many different models of the system to be built can be created and used as part of the systems engineering process. During requirements analysis, logical models are used that describe what the system will do. Later, during system design, physical models are created that show how the system will be implemented.

Requirements modeling is an iterative process. Draft models can be developed early in the process based on the Concept of Operations and the regional ITS architecture. These models are refined as they are used to support requirements elicitation and walkthroughs, keeping bounds on the system and reducing requirements creep.

- **Document requirements –** The requirements are documented in a well-organized, approachable fashion so that the stakeholders and system development team can all easily understand and review them. Typically, a combination of plain language and diagrams are used to define the requirements.

The requirements documentation should include more than requirements. There are many different *attributes* that should be tracked for each requirement. A rich set of attributes is particularly important for large, complex projects. If you are developing such a project, consider specifying the following for each requirement: requirement number, source, author, creation date, change history, verification method, priority, and status. The historical and change-tracking attributes are particularly important since they allow management to measure and track requirements stability.

Traceability is another important aspect of requirements documentation. Each requirement should trace to a higher-level requirement, a stakeholder need, or other governing rules, standards, or constraints from which the requirement is derived. As the system is developed, each requirement will also be traced to the test case that will verify it, to more detailed "child" requirements that may be derived from it, and to design elements that will help to implement it. Establish and populate the Traceability Matrix at this stage, and continue to populate it during development. The Traceability Matrix is a

[14]See http://www.incose.org/ProductsPubs/products/toolsdatabase.aspx.

vital document that is maintained to the end of project development, allowing traceability from user needs to the system components, verification, and validation.

- **Validate requirements –** The documented requirements are carefully checked for consistency, accuracy, and completeness. This is a critical step that is intended to identify requirement defects as early in the process as possible, when correcting them is most economical. To support validation, requirements walkthroughs are held to review the requirements in a systematic way with the project stakeholders and project team.

You will see "validation" used in a few different contexts in systems engineering. Here in *requirements validation*, you make sure that the requirements are correct and complete. Later, in *system validation* (discussed in Section 4.9), you make sure that you have built the right system. In fact, the requirements validation that is performed here will ultimately help to make sure that the system validation is successful in the end.

A walkthrough is a technique in which a review team steps through a deliverable (e.g., requirements, design, or code) looking for problems. A walkthrough should be relatively informal and "blame free" to maximize the number of problems that are identified. A requirements walkthrough should be attended by the people that have a vested interest in the requirements. For a large project, this might include the requirements author, customer, user representative(s), implementers, and testers.

Table 7 identifies an oft-repeated list of attributes of good requirements. As part of the validation process, you do your best to make sure that the requirements have all of these desired attributes. Unfortunately, computers can do only a fraction of this validation and people have to do the rest. Techniques for validating a requirement against each of these quality attributes are also shown in Table 7. An attribute list like this can be converted into a checklist that prompts reviewers to ask themselves the right questions as they are reviewing the requirements.

Table 7: Validating Quality Attributes of Good Requirements

Quality Attribute	Validate by:
Necessary	Make sure that each requirement traces to either a stakeholder need in the ConOps or a parent requirement. A computer can check that the traceability is complete, but people have to verify that the identified traces are valid.
Clear	Some requirements management tools can help with this by looking for red-flag words and constructs in the requirements (e.g., "user friendly", "optimum", "real-time", pronouns, and complex sentences). Most of this aspect of validation relies on walkthroughs and other reviews to make sure the requirements aren't subject to different interpretations. The main culprit here is ambiguity in the English language.
Complete	Does every stakeholder or organizational need in the ConOps trace to at least one requirement? If you implement all of the requirements that trace to the need, will the need be fully met? A computer can answer the first question, but only stakeholder(s) can answer the second.
Correct	In general, it takes a walkthrough to verify that the requirements accurately describe the functionality and performance that must be delivered. The stakeholders must validate that the highest-level system requirements are correct. Traceability can assist in determining the correctness of lower-level requirements. If a child requirement is in conflict with a parent requirement, then either the parent or the child

Quality Attribute	Validate by:
	requirement is incorrect.
Feasible	Again, this must be determined by review and analysis of the requirements. A computer can help with the analysis and possibly even flag words like "instant" or "instantaneous" that may be found in infeasible requirements, but a person ultimately makes the judgment of whether the requirements are feasible. In this case, it is the developer who can provide a reality check and identify requirements that may be technically infeasible or key cost drivers early in the process. Since system performance is dependent on system design and technology choices, requirements feasibility will continue to be monitored and addressed as the system design is developed.
Verifiable	Does the requirement have a verification method assigned? (This is something a computer can check.) Is the requirement really stated in a way that is verifiable? (This much more difficult check can only be performed by people.) For example, ambiguous requirements are not verifiable.

- **Manage requirements –** Processes and tools are established to manage the requirements and associated information that is collected, track changes to the requirements, and provide facilities that support traceability, requirements retrieval and reporting, etc. (See Section 5.4 for more information on configuration management techniques that should be used on the requirements baseline.)

Every ITS project should have a tool that helps to manage the requirements baseline. More complex ITS projects will benefit from a tool specifically for requirements management such as DOORS or Requisite-Pro.[15] A professional requirements management tool is expensive, but it includes a long list of capabilities including change management, requirements attributes storage and reporting, impact analysis, requirements status tracking, requirements validation tools, access control, and more.

Like the other requirements engineering activities, the requirements management capabilities should be scaled based on the complexity and size of the ITS project. Requirements for smaller ITS projects can be managed easily and effectively by a single engineer using a general purpose tool like Microsoft Access or Excel.

- **Create a System Verification Plan –** As the requirements are documented, a plan for verifying the system based on the requirements is defined. This is extremely important because only verifiable requirements should find their way into the system requirements document. A verification method is identified for every requirement – normally, by one of four ways: Test, Demonstration, Inspection, or Analysis. The purpose of this early assignment of a method, long before the requirements will actually be verified, is to make sure that the requirements author thinks about how the requirement will be verified from the very start. (See Section 4.7 for more information on System Verification.)
- **Create a System Acceptance Plan –** A plan should be created that describes the functionality the new system must successfully display prior to acceptance by the customer; consensus by all parties on the contents of this plan should be reached early in the life cycle.

[15] INCOSE (www.incose.org) has a comprehensive list of requirements management tools in its tools database.

4.4.3 Outputs

No matter how you developed your requirements, you must document them in some consistent, accessible, and reviewable way. The requirements development process may result in several different levels of requirements over several steps in the "V" – stakeholder requirements, system requirements, subsystem requirements, etc. – that may be documented in several different outputs. For example, stakeholder requirements might be documented in a series of Use Cases; system requirements, in a System Requirements Specification; and subsystem requirements, in subsystem specifications. All of these requirements should be compiled in a single repository that can be used to manage and publish the requirements specifications at each stage of the project.

It is much easier to use a standard template for the requirements specifications than it is to come up with your own, and numerous standard templates are available. If your organization does not have a standard requirements template, you can start with a standard template like the one contained in IEEE Standard 830 (for software requirements specifications) or IEEE Standard 1233 (for system requirements specifications). Starting with a template saves time and ensures that the requirements specification is complete. Of course, the template can be modified as necessary to meet the needs of the project.

The system requirements specification should fully specify the system to be developed and should include the following information:

- System boundary with interfacing systems clearly identified
- General system description, including capabilities, modes, and users, as applicable
- External interface requirements for interfacing systems and people
- Functional requirements and associated performance requirements
- Environmental requirements
- Life-cycle process requirements supporting development, qualification (e.g., test, verification, validation, and acceptance), production, deployment, transition, operations and maintenance, change and upgrade, and retirement/replacement, as applicable
- Reliability and availability
- Expandability
- Staffing, human factors, safety, and security requirements; and
- Physical constraints (such as weight and form factors).

As you read through this list, you may recognize that some of this information has already been collected and documented in previous steps, and there is no need to recreate it here. Refer back to the Concept of Operations that already contains a description of the system boundary, the system itself, and other items in this list.

A System Verification Plan, describing the approach for verifying each and every system requirement, and a System Acceptance Plan, describing the capabilities that must function successfully for customer acceptance, should be created, reviewed, and approved.

4.4.4 Examples

Stakeholder Requirements

The Oregon DOT TripCheck project developed a User Functional Requirements Specification, which lists the user requirements for the redesigned TripCheck website. The excerpt from this document in

Table 8 shows several user requirements for the website autorouting function. As shown, every requirement is prioritized on a scale from 1 ("must have") to 4 ("don't implement") and is related to different types of end users – Commuters (C), Inter-City Travelers (ICT), Tourist Travelers (TT), ADA Travelers (ADA), and Commercial Truckers (CT). These prioritized user requirements were used by the contractor to support Use Case modeling and to define system requirements.

Table 8: ODOT TripCheck User Requirements (Excerpt)

REQ ID	ODOT PRIORITY	REQUIREMENT	AUDIENCE SEGMENTS					
			C	ICT	TT	ADA	CT	ALL
RR001	2	The system should allow the user to enter a multi-point route using a combination of the criteria specified in MP004c-f and h.	X	X	X	X		
RR0011	1	The system shall allow the user to select destination points by clicking on the map.	X	X	X	X		
		The system shall allow the user to specify the following when determining road routes (note: this functionality is for trip planning.):						
RR002a	2	- starting date AND/OR ending date if only one date is specified, the system calculates the other.			X	X		
RR002b	2	- starting time/ending time if only one time is specified, the system calculates the other.			X	X		
RR002c	2	- month of travel (instead of start/end)			X	X		
RR002d	4	- quickest route (by time)						X
RR002e	1	- shortest route (by miles)	X	X	X	X		
RR002f	2	- most scenic route (based on scenic byways within a user-specified mile radius of the direct (shortest) route			X	X		
RR002g	3	- routes most recommended by others			X	X		

Note that stakeholder requirements that are collected through the requirements elicitation process are likely to have a few imperfections. The key is to document the stakeholder requirements, make them as clear and succinct as possible, prioritize them, and then use them to develop more formally stated system requirements.

System Requirements

The Maryland CHART II system is a statewide traffic management system that has been operational since 2001. The CHART program maintains a website that provides all of the CHART documentation at http://www.chart.state.md.us, including a comprehensive system

requirements document. A few of the system requirements for the equipment inventory and report generation functions are shown in Table 9.

Table 9: CHART II System Requirements (Excerpt)

3.1.3 Equipment Inventory *The equipment inventory is a list of SHA equipment used in connection with CHART response to incidents. The system provides functions to maintain the inventory, equipment status, and to generate alerts for delinquent equipment.*
3.1.3.1 The system shall provide the capability to maintain the equipment inventory.
3.1.3.1.1 The system shall support the addition of new equipment entries to the inventory.
3.1.3.1.2 The system shall support the modification of existing equipment inventory entries.
3.1.3.1.3 The system shall support the deletion of equipment inventory entries.
3.1.3.1.4 The system shall support the allocation of equipment to events.
3.1.4 Report Generation *This section lists requirements for the generation of reports from the CHART system and archive data.*
3.1.4.1 The system shall provide the capability to generate reports from online and archived data.
3.1.4.2 The system shall support the generation of operational reports.
3.1.4.2.1 The system shall support the generation of a Center Situation report.
3.1.4.2.2 The system shall support the generation of a Disable Vehicle event report.
3.1.4.2.3 The system shall support the generation of an Incident event report.
3.1.4.2.4 The system shall support the generation of traffic volume reports.

Traceability Matrix

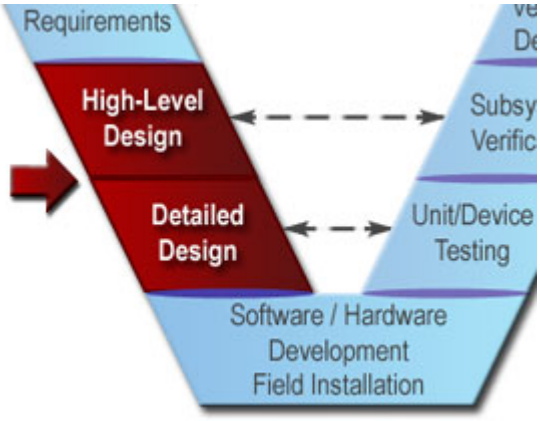
Table 10 is a typical traceability matrix that would be maintained and populated throughout the project development process. The matrix may be maintained directly in a database or spreadsheet for small projects or generated and maintained with a requirements management tool for more complex projects. Using either approach, the matrix provides backwards and forwards traceability between stakeholder needs (and other potential requirements sources), system requirements, design, implementation, and verification test cases. As shown, only the unique identifiers (e.g., UN1.1) are actually included in the traceability matrix so you don't have to keep many instances of the actual text up-to-date. Note also that the design and implementation columns would not actually be completed until later in the process.

Table 10: Sample Traceability Matrix

Requirement Source	System Requirement	High-Level Design Component	Code Unit	Test Case
UN1.1	R00220	7.2.3	SystemMonitor	UT 4.2
	R00330	7.3.1	CalcVolume	UT 5.5
	R00331	7.3.1	CalcCount	

4.5 System Design

In this step: A system design is created based on the system requirements including a high-level design that defines the overall framework for the system. Subsystems of the system are identified and decomposed further into components. Requirements are allocated to the system components, and interfaces are specified in detail. Detailed specifications are created for the hardware and software components to be developed, and final product selections are made for off-the-shelf components.



OBJECTIVES	▪ Produce a high-level design that meets the system requirements and defines key interfaces, and that facilitates development, integration, and future maintenance and upgrades ▪ Develop detailed design specifications that support hardware and software development and procurement of off-the-shelf equipment
INPUT ***Sources of Information***	▪ Concept of Operations ▪ System Requirements document ▪ Off-the-shelf products ▪ Existing system design documentation ▪ ITS standards ▪ Other industry standards
PROCESS ***Key Activities***	▪ Evaluate off-the-shelf components ▪ Develop and evaluate alternative high-level designs ▪ Analyze and allocate requirements ▪ Document interfaces and identify standards ▪ Create Integration Plan, Subsystem Verification Plans, and Subsystem Acceptance Plans ▪ Develop detailed component-level design specifications
OUTPUT ***Process Results***	▪ Off-the-shelf evaluation and alternatives summary reports ▪ High-level (architectural) design ▪ Detailed design specifications for hardware/software ▪ Integration Plans, Subsystem Verification Plans, Subsystem Acceptance Plans, and Unit/Device Test Plans
REVIEW ***Proceed only if you have:***	▪ Approved high-level design for the project ▪ Defined all system interfaces ▪ Traced the system design specifications to the requirements ▪ Approved detailed specifications for all hardware/software components

4.5.1 Overview

In the systems engineering approach, we define the problem before we define the solution. The previous steps in the "V" have all focused primarily on defining the problem to be solved. The system design step is the first step where we focus on the solution. This is an important transitional step that links the system requirements that were defined in the previous step with system implementation that will be performed in the next step, as shown in Figure 18.

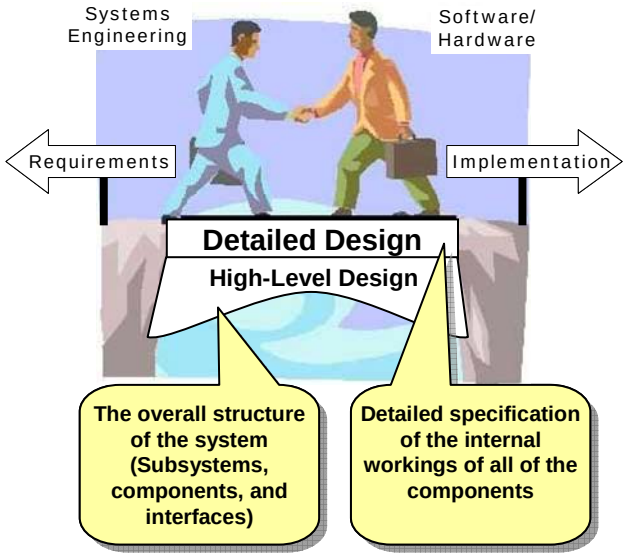


Figure 18: System Design is the Bridge from Requirements to Implementation

There are two levels of design that should be included in your project design activities:

High-level design is commonly referred to as *architectural design* in most systems engineering handbooks and process standards. Architectural design is used because an overall structure for the project is defined in this step. IEEE 610[16] defines architectural design as "*the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system*". Of course, ITS projects may include several computer systems, a communications network, distributed devices, facilities, and people. High-level design defines a framework for all of these project components.

Detailed design is the complete specification of the software, hardware, and communications components, defining *how* the components will be developed to meet the system requirements. The software specifications are described in enough detail that the software team can write the individual software modules. The hardware specifications are detailed enough that the hardware components can be fabricated or purchased.

Many consider design to be the most creative part of project development. Two different designs might both meet the system requirements, but one could be far superior in how efficiently it can be developed, integrated, maintained, and upgraded over time. Perhaps the most significant contributor to a successful design is previous design experience with similar systems. The latest car designs all build on 100 years of accumulated automotive design experience. Similarly, the design of a new transportation management system should build on existing successful transportation management system designs. In both cases, the system designer builds on knowledge of what worked before and, perhaps even more importantly, what did not.

It is extremely rare to find an ITS system that is truly "unprecedented", so many if not most system designs should be able to build on existing design information. This is particularly true for projects that are extending an existing system that already includes a well-documented design. In this case, the high-level design will change only to the degree that new functionality or interfaces are added. Similarly, much of the detailed design can be reused for projects that extend the coverage of an existing system.

[16]IEEE 610 is the *IEEE Standard Computer Dictionary.*

4.5.2 Key Activities

System design is a cooperative effort that is performed by systems engineers and the implementation experts who will actually build the system. The process works best when there is a close working relationship among the customer, the systems engineers (e.g., a consultant or in-house systems engineering staff), and the implementation team (e.g., a contractor or in-house team).

High-Level Design

High-level design is normally led by systems engineers with participation from the implementation experts to ensure that the design is implementable. Typical activities of high-level design are shown in Figure 19. Each of the activities are actually performed iteratively as high-level design alternatives are defined and evaluated.

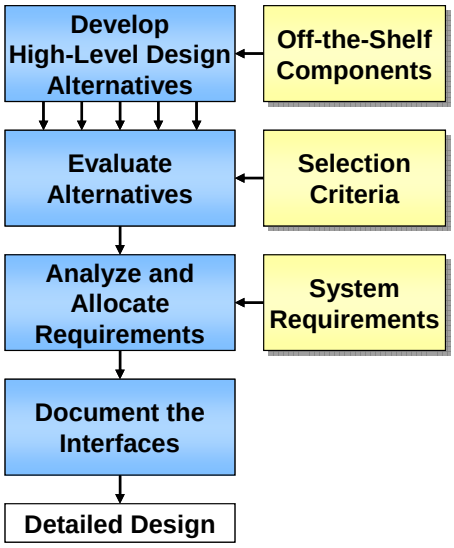


Figure 19: High-Level Design Activities

- **Evaluate off-the-shelf components –** One key aspect of high-level design is the identification of components that will be purchased, reused, or developed from scratch. The project may be required to use off-the-shelf hardware or software, or this may simply be the preferred solution. Specific design constraints may also require that a particular product be used. For example, a municipality that is expanding a signal control system that already includes 300 Type 170 controllers may constrain the design of the expansion to use the same controllers to facilitate operation and maintenance of the overall system. State DOTs and other large agencies often publish approved products lists that identify ITS-related products that meet agency specifications.

  When off-the-shelf components will be used, the high-level design must be consistent with the capabilities of the target products. The designer should have an eye on the available products as the high-level design is produced to avoid specifying a design that can be supported only by a custom solution. A particular product should not be specified in the high-level design unless it is truly required. When possible, the high-level design should be vendor and technology independent so that new products and technologies can be inserted over time.

  You should give off-the-shelf hardware and software serious consideration and use it where it makes sense. The potential benefits of off-the-shelf solutions – reduced acquisition time and cost, and increased reliability – should be weighed against the requirements that may not be satisfied by the off-the-shelf solution and potential loss of flexibility. If you have requirements that preclude off-the-shelf solutions, determine how important they are and what their real cost will be. This make/buy evaluation should be documented in a summary report that considers the costs and benefits of off-the-shelf and custom solution alternatives over the system life cycle. This report should be a key deliverable of the project.

  Also recognize that there is a large grey area between off-the-shelf and custom software for ITS applications. Every qualified software developer starts with an established code base when creating the next "custom solution", accruing some of the benefits of off-the-

shelf solutions. Many vendors of off-the-shelf solutions offer customization services, further blurring the distinction between off-the-shelf and custom software.

Resources

The FHWA report *The Road to Successful ITS Software Acquisition* includes a good discussion of software make/buy decision factors and a lot of other good information on software acquisition for ITS. The executive summary for the report is available at www.itsdocs.fhwa.dot.gov/jpodocs/repts_te/36s01!.pdf.

- **Develop and evaluate high-level design alternatives –** The system is partitioned into subsystems, and the subsystems are in turn partitioned into smaller assemblies. The process continues until system components – the elemental hardware and software configuration items – are identified. Figure 20 shows a partial decomposition of an electronic toll collection system that identifies all of the major subsystems and the components for the Video Enforcement subsystem.

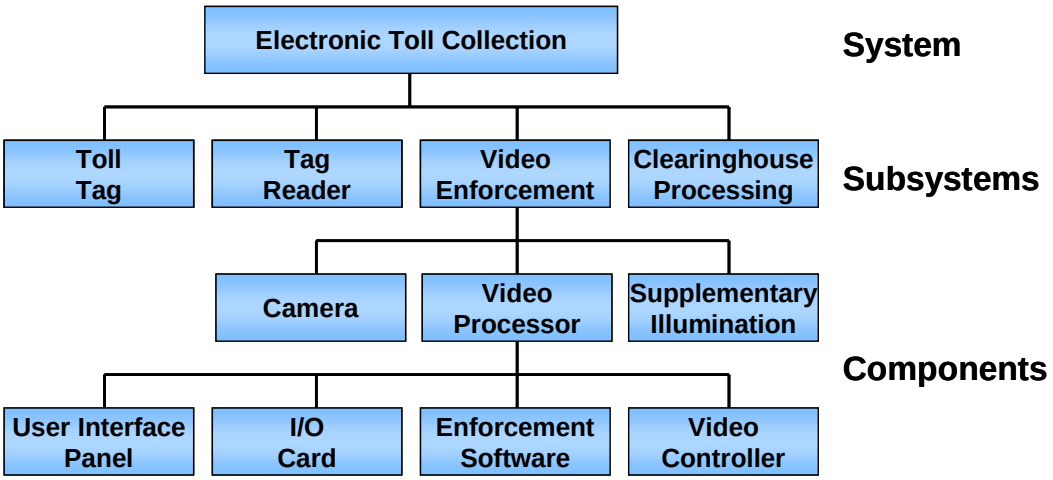


Figure 20: Electronic Toll Collection Subsystems and Components (Excerpt)

There are many different ways that a system can be partitioned into subsystems and components. In this Electronic Toll Collection example, we might consider whether the Clearinghouse Processing subsystem should be handled by a single centralized facility or distributed to several regional facilities. As another example, vehicle detectors could be included in the Video Enforcement subsystem or in the Tag Reader subsystem, or both.

Even a relatively simple traffic signal system has high-level design choices. For example, a traffic signal system high-level design can be two-level (central computer and local controllers), three-level (central computer, field masters, and local controllers), or a hybrid design that could support either two or three levels. High-level design alternatives like these can have a significant impact on the performance, reliability, and life-cycle costs of the system. Alternative high-level designs should be developed and compared with respect to defined selection criteria to identify the superior design.

The selection criteria that are used to compare the high-level design alternatives include consistency with existing physical and institutional boundaries; ease of development, integration, and upgrading; and management visibility and oversight requirements. One of the most important factors is to keep the interfaces as simple, standard, and foolproof as possible. The selection criteria should be documented along with the analysis that identifies the superior high-level design alternative that will be used. If there are several viable alternatives, they should be reviewed by the project sponsor and other stakeholders.

The Rule/Policy requires the systems engineering analysis for ITS projects to include an analysis of alternative system configurations.

- **Analyze and allocate requirements –** The requirements analysis described in Section 4.4.2 continues as the requirements are decomposed until there is enough granularity to allocate requirements to the system components identified in the high-level design.

  The detailed functional requirements and associated performance requirements are allocated to the system components. To support allocation, the relationships between the required system functions are analyzed in detail. Once you understand the relationships between functions, you can make sure that functions that have a lot of complex and/or time-constrained interactions are allocated to the same component as much as possible. Through this process, each component is made as independent of the other components as possible.

You would not want to develop a high-level design and requirements allocation for a complex ITS project without software tools. Fortunately, there are many good tools that support both requirements analysis and architectural design. The INCOSE tools database, available to nonmembers free of charge at www.incose.org, includes a broad range of systems engineering tools and a detailed survey of tools that support requirements management and system architecture.

- **Document the interfaces and identify standards –** Interfaces should be identified early, fully documented, and then managed throughout the project development. Interface specifications should be developed for external interfaces (i.e., interfaces between the current project and external systems) and internal interfaces (i.e., interfaces between project components). Interfaces between systems that are owned and operated by different agencies may require additional lead time to negotiate interface agreements.

  This is the place to identify ITS standards and any other industry standards that will be used in detail. There are a variety of standards that should be considered at this point. Take a look at all interfaces, both external and internal. Since your regional ITS architecture and/or project ITS architecture was based on the National ITS Architecture, many of the interfaces probably already have a set of ITS standards you should consider. You should also identify standards that are used in your region or state, and also in adjoining states if your project is a multistate deployment. A methodical assessment should be made for each interface to determine which standards are relevant, which standards should be deployed, and perhaps which standards should be phased in over time as part of a longer-range plan.

  Once you have taken a look at the relevant standards, beginning with your system's external interfaces, document the nature of the data, formats, ranges of values, and periodicity of the information exchanged on the interface. Then proceed to each of the internal interfaces and document the same information for those.

Agencies are encouraged to incorporate the ITS standards into new systems and upgrades of existing systems. The Rule/Policy requires the systems engineering analysis for ITS projects to include an identification of ITS standards. Consult the ITS Standards Program website at http://www.standards.its.dot.gov/ for more information and available resources supporting standards implementation.

- **Create Integration Plan, Subsystem Verification Plans, and Subsystem Acceptance Plans –** An Integration Plan, Subsystem Verification Plans, and Subsystem Acceptance Plans should be completed parallel with the high-level design. (See Section 4.7 for more information on integration and verification planning.)

Detailed Design

Hardware and software specialists create the detailed design for each component identified in the high-level design. Systems engineers play a supporting role, providing technical oversight on an ongoing basis. As you might expect, the detailed design activity will vary for off-the-shelf and custom components, as shown in Figure 21.

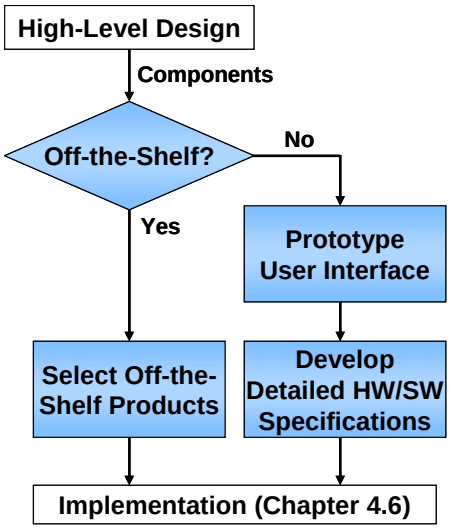


Figure 21: Detailed Design Activities

- **Prototype user interface –** If a user interface is to be developed, a simple user interface prototype is an efficient way to design it.

A prototype is a quick, easy-to-build approximation of a system or part of a system. A software prototype can be used to quickly implement almost any part of a system that you want to explore, but it is used most often to make a quick approximation of a user interface for a new system.

A user interface prototype should be employed to help the user and developer visualize the interface before significant resources are invested in software development. This is one area in particular where you can expect multiple iterations as the developers incrementally create and refine the user interface design based on user feedback. (You will find that it is often easier to get users to provide feedback on a prototype than on system requirements and design specifications, which can be tedious to review.)

While the user interface prototype is included here because it is an effective way to design the user interface, prototypes may actually be generated much earlier in the process, during system requirements development. The prototype can turn the requirements statements into something tangible that users can react to and comment on.

- **Develop detailed hardware and software component design specifications –** Detailed design specifications are created for each hardware and software component to be developed. In the high-level design, each component is defined in terms of its functionality and performance, with particular focus on its interfaces to external systems and other components. The level of detail in the detailed design specifications is greater than that in the high-level design in two important respects:
    - The detailed design will often include another layer of architectural design for complex components. Figure 22 shows another layer of decomposition that might be defined for the Enforcement Software component that was identified in the high-level design example for the Electronic Toll Collection system (in Figure 20). All hardware/software units and their interfaces are defined to provide a framework for development of the component.

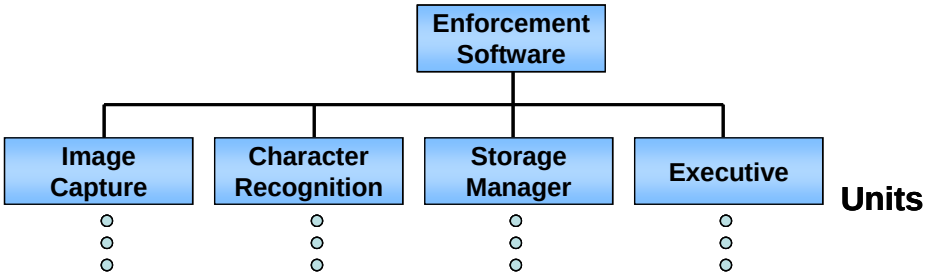


Figure 22: Architectural Design within a System Component

- The detailed design specifies exactly *how* the component will be implemented so that it meets the requirements. For hardware, schematic drawings and parts lists are defined. For software, this includes identification of algorithms, detailed data structures, and specification of third-party software packages that will be used. In the Electronic Toll Collection example, an off-the-shelf real-time executive would be selected and the image capture and character recognition algorithms would be defined.

The detailed design of each component should be reviewed to verify that it meets the allocated requirements and is fit for the intended purpose. Periodic or as-needed reviews can be held to monitor progress and resolve any design issues. For larger projects, coordination meetings should be held to ensure that concurrent design activities are coordinated to mitigate future integration risks. At the completion of the detailed design step, a broader stakeholder meeting is held to review and approve the detailed design before the implementation team begins to build the solution.

- **Select off-the-shelf (OTS) products –** One of the fundamental principles of systems engineering is to delay technology choices until you have a solid foundation for making the right choice. By waiting until this point in the process, the latest technologies and products can be selected, and these selections can be based on a thorough understanding of the requirements and the overall architecture of the system. The selections can also be made by specialists who are closest to the implementation and are therefore best equipped to make them.

  There are two fundamental ways that a product can be selected, depending on your procurement requirements and selected procurement strategy:

  - A trade study can be performed that compares the alternative products and selects the best product based on selection criteria that are in turn based on the specification.
  - A competitive procurement can be used that allows vendors to propose products that will best meet the specification.

  In either case, product selection should be driven by a good performance-based specification of the product.

  Specifications can be either *performance-based* or *prescriptive*. In a performance-based specification, you specify the functionality and the performance that are required rather than what equipment to use. In a prescriptive specification, you specify exactly the equipment that you want. A performance-based specification for a dynamic message sign would include statements like "The sign shall provide a display of 3 lines of 25 characters per line." A prescriptive specification would be "The Trantastic LED Model XYZ sign shall be used." Performance-based specifications tend to provide the best value because they allow the contractor or vendor maximum flexibility to propose the best solution that meets your needs.

  If a trade study is performed, then the functional and performance requirements that are allocated to the product should be used to define product selection criteria. An alternatives analysis document captures the alternatives that were considered and the selection criteria that were used to select the superior product. Existing trade studies, approved product lists, and other resources can be used to facilitate product selection.

  The evaluation of OTS products should be reviewed to verify that the evaluation criteria were properly defined and applied fairly and that an appropriate range of products was considered.

- **Create Unit/Device Test Plans –** Test plans should be created for each hardware and software component to test all requirements identified in the HW/SW design specifications.

4.5.3 Outputs

High-Level Design

There isn't a single "best way" to present the high-level design to stakeholders and developers since different users will have different needs and different viewpoints. Over the years, high-level designs have evolved to include several different interconnected "views" of the system. Each view focuses on a single aspect of the system, which makes the system easier to analyze and understand. The specific views that are presented will vary, but they will typically include a physical view that identifies the system components and their relationships; a functional view that describes the system's behavior; a technical view that identifies the interfaces in detail, including the standards to be used; and an informational view that describes the information that will be managed by the system. As shown in Figure 23, these views are just different ways of looking at the same system.

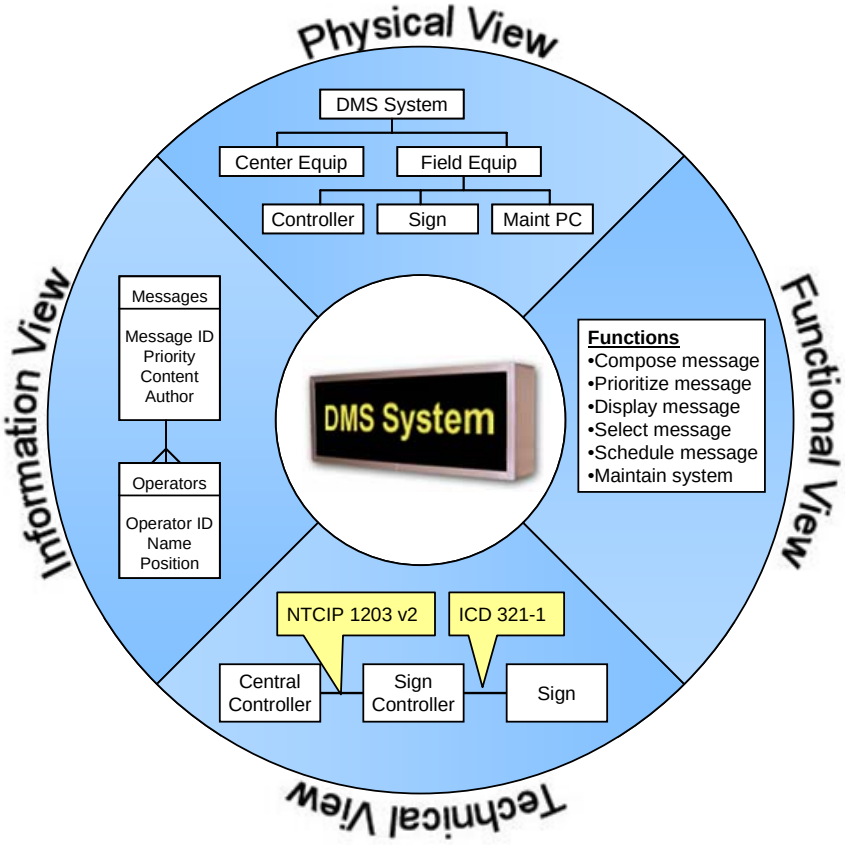


Figure 23: High-Level Design May Include Several Views

Other outputs of the high-level design process include Integration Plans, Subsystem Verification Plans, and Subsystem Acceptance Plans that will be used in the integration and verification of the system. (See Section 4.7 for further details.)

Detailed Design

This activity results in the design of hardware and software for all system components that will support hardware and software development and off-the-shelf product procurement. Other artifacts of the development process include unit/device verification plans. A record of the technical reviews that were conducted should also be included in the project documentation.

4.5.4 Examples

High-Level Design

The CHART II documentation includes a system architecture document that includes many different views of the CHART II system, such as entity relationship diagrams, Use Case diagrams, and network architecture diagrams. Table 11 is an excerpt from the document that shows the subsystems included in the CHART II software.

Table 11: CHART II Software Subsystems (Excerpt)

Software	
CI Name	**Subsystems**
CHART II	**Alert Management**
	Audio
	AVL
	Camera Control
	Communications Log Management
	Data Export Management
	Device Management
	Dictionary
	DMS Control
	HAR Control
	HAR Notification
	Message Library Management
	Notification Management
	Plan Management
	Resource Management
	Schedule Management
	SHAZAM Management
	Signals
	Simulation
	System Monitor
	Traffic Event Management
	Traffic Sensor System Management
	User Management
	Utility
	Video Monitor Management

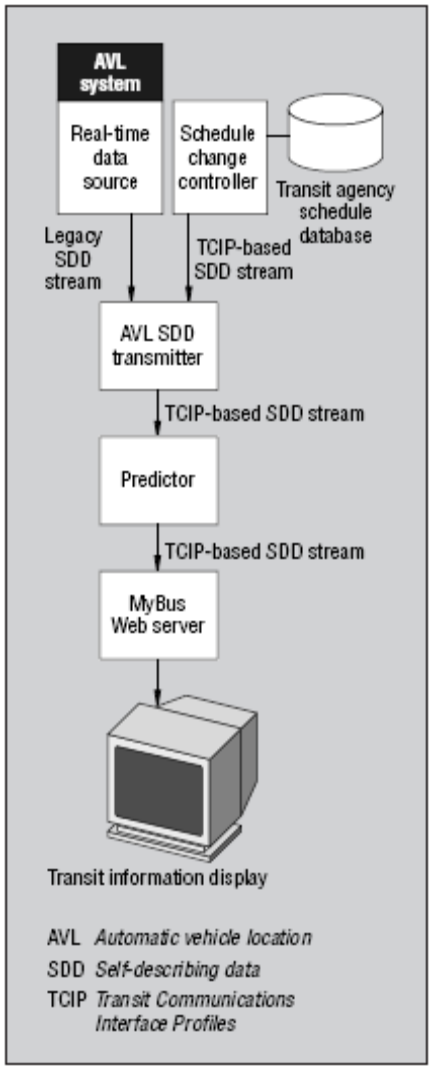


Figure 24: Metro Transit MyBus System Architecture

In contrast with the CHART II statewide system high-level design, many smaller ITS projects have relatively simple high-level designs, such as the system architecture for the MyBus system depicted in Figure 24. This figure identifies the subsystems and major interfaces in the MyBus system.

ITS projects that include significant user interface development should prototype the user interface to

help users visualize the software that will be developed before significant resources are committed. The objective is to develop a prototype that demonstrates the software look and feel with the least amount of work possible. The simplest prototypes are a series of static images in paper form. For example, when ODOT redesigned its TripCheck website, the implementation team developed a series of "wireframe" diagrams that showed the proposed interface design with enough detail to gather user feedback. One of the 40 wireframe diagrams that was included in the design package is shown in Figure 25.

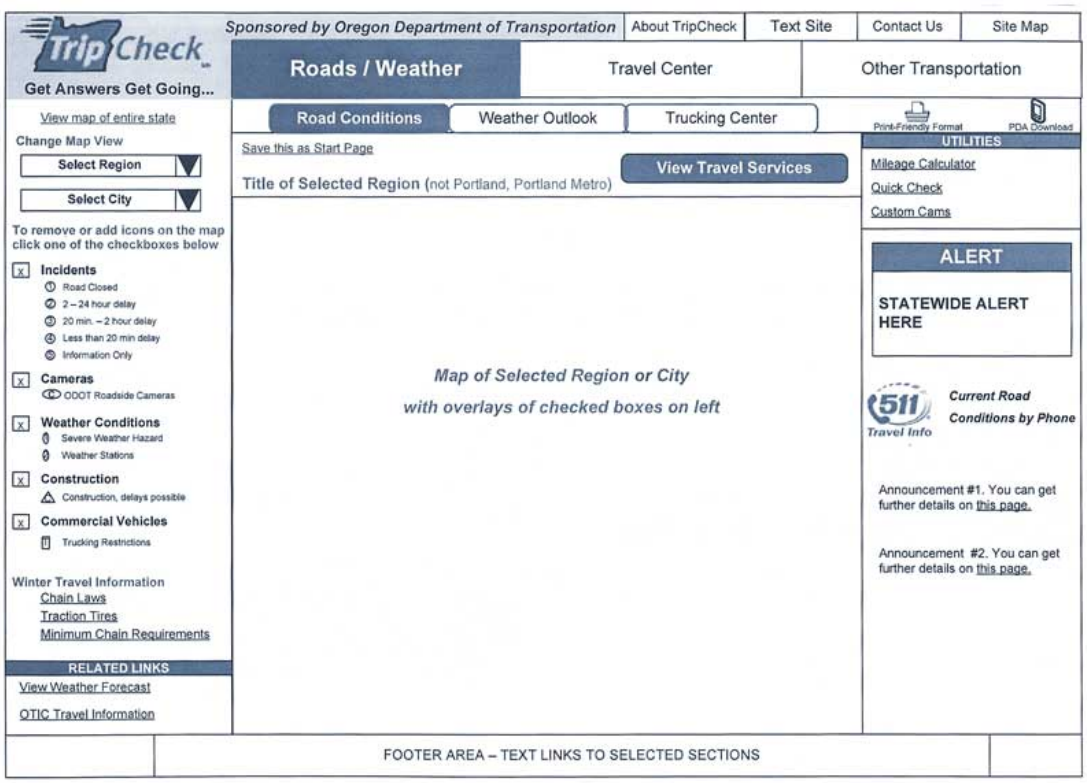


Figure 25: User Interface Prototype Example: ODOT TripCheck Wireframe Diagram

Detailed Design

There are many ways to document software detailed design. Most commonly, it is portrayed using object-oriented techniques and the Unified Modeling Language[17], but any technique that the implementation team selects is fine as long as it is detailed enough to support software construction and clear enough to support peer reviews and walkthroughs.

Table 12 is an example of a detailed design for part of the Shadow software that works behind the scenes to keep the traffic information on the ODOT TripCheck website up to date. Note that the interface is defined and that loosely structured program design language (PDL) is used to define the algorithm that is used to process transactions. If much of this appears to be gibberish to you, you are not alone. This is why many agencies use software specialists to provide an independent review of the detailed software development artifacts for higher risk software projects on their behalf.

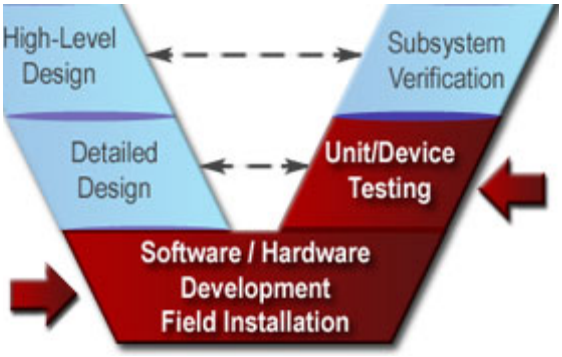
[17]There are many resources available if you want to learn more about UML; www.uml.org is a good place to start.

Table 12: Detailed Software Design Example: ODOT TripCheck Software Class Definition

Class Name:	clsShadow::ProcessTransactions
Type:	*Class Method*
Scope:	*Public*
Arguments:	*n/a*
Return Value:	*n/a*
Error Messages:	*Error reading shadow table*
Error Handling	*True*
Files Accessed:	*n/a*
Files Changed:	*n/a*
Methods Called:	*n/a*
Narrative:	*Processes any transactions waiting in the Shadow Table.*
Pseudocode(PDL):	*Open Connection to Shadow Table* *Retrieve all transactions* *Do While Not Rs.Eof* *Create a collection of clsTransaction objects* *Loop* *If we have events to process...* *Raise Event NewTransactions(clsTransactions)* *Clean Up*

4.6 Software/Hardware Development and Testing

In this step: Hardware and software solutions are created for the components identified in the system design. Part of the solution may require custom hardware and/or software development, and part may be implemented with off-the-shelf items, modified as needed to meet the design specifications. The components are tested and delivered ready for integration and installation.



OBJECTIVES	▪ Develop and/or purchase hardware and software components that meet the design specifications and requirements with minimum defects ▪ Identify any exceptions to the requirements or design specifications that are required
INPUT ***Sources of Information***	▪ System and subsystem requirements ▪ System design ▪ Off-the-shelf products ▪ Industry standards ▪ Unit/Device Test Plans
PROCESS ***Key Activities***	▪ Plan software/hardware development ▪ Establish development environment ▪ Procure off-the-shelf products ▪ Develop software and hardware ▪ Perform unit/device testing
OUTPUT ***Process Results***	▪ Software/hardware development plans ▪ Hardware and software components, tested and ready for integration ▪ Supporting documentation (e.g., training materials, user manuals, maintenance manuals, installation and test utilities)
REVIEW ***Proceed only if you have:***	▪ Conducted technical reviews of the hardware/software ▪ Performed configuration/quality checks on the hardware and software ▪ Received all supporting documentation ▪ Verified that unit/device testing has been successfully completed

4.6.1 Overview

Although hardware and software development may be the first task that comes to mind when thinking about an ITS project, the systems engineering approach focuses on the preceding requirements and design steps and on the integration, verification, and validation steps to follow.

This is where the investment in a clear set of requirements and a good system design should begin to pay dividends. The systems engineering process now provides technical oversight as an implementation team of specialists fabricates the hardware and writes the software. This is a highly iterative process, particularly for software, where key features may be incrementally implemented, tested, and incorporated into the baseline over time. Progress is monitored through a planned series of walkthroughs, inspections, and reviews, as shown in Figure 26.

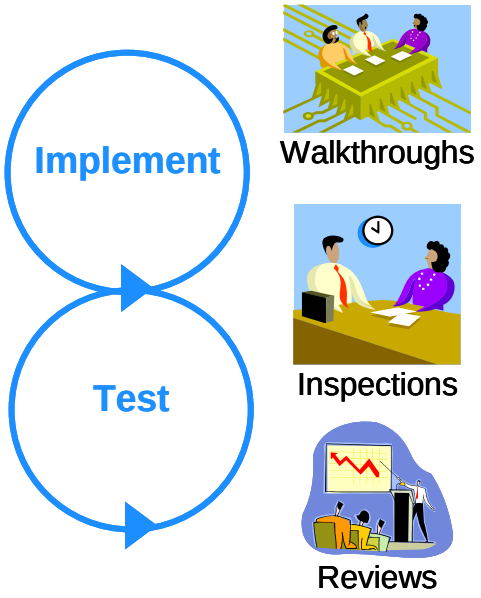


Figure 26: Monitoring Software/Hardware Development

Although the systems engineering approach does not specify the mechanics of hardware and software development (this is left to the implementation team), the development effort is obviously critical to project success. **This is the time to build quality into the hardware/software and to minimize defects.** A common refrain in the software industry is that you can't test quality into the software – you must build it in from the beginning. The systems engineering activities that are suggested in this chapter are intended to ensure that the implementation team builds quality into their products.

In practice, most of the hardware that is used for ITS projects is purchased off the shelf. Software development is more prevalent, but many ITS projects include little or no software development. ITS projects that do not include custom hardware or software development acquire the necessary off-the-shelf hardware and software components at this step. Detailed specifications created as part of the detailed design step described in Section 4.5 are used to support the acquisition. The system components are acquired, and bench testing is performed to verify that they meet their specifications. In such cases, the detailed hardware/software development and unit testing described in this chapter are not required.

Custom software development for ITS projects has proven to be a relatively risky endeavor. This is why software development receives more attention than hardware development in this chapter. It is beyond the scope of this document to discuss specific software development techniques, but there are several clear factors that contribute to software development success:

- No matter how clear and unambiguous the requirements appear, it is almost certain that the software customer and the software implementation team will interpret some of the requirements differently. Requirements walkthroughs that are described in Section 4.4.2 help to mitigate this risk, but ultimately the customer/stakeholders will have to monitor the software as it is being developed to ensure that the development is proceeding in the right direction. Expect and plan for course corrections and requirements changes along the

way, at least until we discover the way to build the "perfect specification". Ensure that the contract is flexible enough to have a couple of reviews and that it allows some visits or informal reviews with the developers to see how they are doing. This might be one of the project risks to include in your risk management plan. (See Section 5.3 for more information on risk management.)

> Consider the following requirement:
>
> "The system shall turn off the alarm when the user presses the 'F6' key."
>
> Even this seemingly clear, very specific requirement was subject to two interpretations. The customer assumed that a currently sounding alarm is turned off when the button is pressed, and the implementer assumed that the capability to sound an alarm is turned off. Even "good" specifications are subject to potential pitfalls like this.

- All documentation should be reviewed and approved. One of the biggest problems faced by system implementers is the customer's failure to review documentation, which leads to a system at the end of the project that does not meet the customer's expectations. If the documentation that is generated is not reviewed, then much of the benefit of systems engineering will not be realized. Depending on your background, you may need to find experts who can review more technical documentation on your behalf.

- Perhaps the best way to reduce software development risk is to proceed in small steps and build incremental software releases in short, successive time periods. Unlike a typical roadway project that can be fully specified and implemented all at once, complex software should be implemented incrementally, with months or even weeks between releases. Incremental, iterative development with frequent coordination and feedback is the best way to keep software development on track, particularly for projects where the requirements are not completely understood at the outset.

4.6.2 Key Activities

The hardware and software specialists implement and test each system component. Systems engineers play a supporting role, providing technical oversight on an ongoing basis to identify minor issues early, before they grow into large problems. The process works best when there is a close working relationship among the customer, the systems engineers (e.g., a consultant or in-house systems engineering staff), and the implementation team (e.g., a contractor or an in-house team). Each of the activity descriptions is followed by a discussion of the technical review and monitoring of that activity.

- **Plan software/hardware development –** The implementation team documents its development process, best practices, and conventions that will be used. The Software/Hardware Development plan should address development methods, documentation requirements, delivery stages, configuration control procedures, technical tracking and control processes, and the review process. The plan(s) should be reviewed by the customer and the broader project team.

The Software/Hardware Development plan should be reviewed and approved before development begins. Well-qualified implementation teams will already have proven processes in place that can be tailored for the specific project, so this shouldn't be viewed as a burdensome activity. The intent is not to mandate a particular implementation process but to ensure that the implementation team has an established process that they will follow. An implementation team that doesn't have a documented process is a red flag.

- **Establish development environment –** The tools that are used to develop and test the software are selected, procured, and installed, including development tools, source control tools, third-party application libraries, and test simulators. Every tool that is used to support software development should be documented specifically enough so that the development environment can be replicated if necessary.

Although it is sometimes overlooked, the development environment is just as critical to future software maintenance as the actual detailed design documentation and source code. Every tool that is used to develop and test the software should be documented, including version information and complete documentation of any customization or extensions. If this is a custom development and you have paid for the tools, include the development environment as a project deliverable.

A peer review or inspection can be used to verify that the development environment is adequate and accurately documented. Once established, the development environment should be placed under configuration management (discussed in Section 5.4) so that changes to the environment are tracked. Seemingly minor changes like application library upgrades or operating system service pack upgrades can cause problems later if they are not controlled and tracked.

- **Procure off-the-shelf products –** Off-the-shelf products are procured based on the product specifications developed in the detailed design step (see Section 4.5).

Delay procurement until the products are actually required to support the implementation. Too much lead time can result in hardware or software that becomes outdated before it can be integrated into the project. Too little lead time could cause procurement delays that impact the project schedule.

- **Develop software and hardware –** The software is written and the hardware is built based on the detailed design. The current state of the practice is to develop the software incrementally and release it in stages. The initial releases implement a few core features, and subsequent releases add more features until all requirements are satisfied. For example, a TMC project might first implement a basic dynamic message sign capability and demonstrate its ability to post messages to the sign and to monitor sign status. Then, more advanced message scheduling and message library management functions could be implemented. This incremental approach enables early and ongoing feedback between the customer and the implementation team. If this approach is used, then a staged delivery plan, which defines the order in which the software will be developed and the staged release process, should be included in the Software Development Plan.

  Releases will be developed, tested, and made available to selected users for feedback. Providing feedback on interim releases is only part of the technical oversight that should be performed. Code inspections and code walkthroughs should also be used to check the software quality; these are the only ways to ensure that the software is well structured, well documented, and consistently follows the coding standards and conventions. Independent reviewers with software expertise should be used to help verify software quality on the customer's behalf if the customer agency does not have the right expertise.

Most project managers who have managed software development efforts are familiar with the "90% complete" syndrome, in which software developers quickly reach "90% complete" status but the development effort then languishes as the final 10% takes much more work than anticipated. Project tracking should be based on discrete, measurable milestones instead of arbitrary "% complete" estimates from the software developers. For example, instead of tracking the developer's estimated "% complete", set up a monitoring

system that gives credit for completed software only when the piece of code has been successfully tested and integrated into the next release.

- **Develop supporting products –** Enabling products, such as training materials, user and maintenance manuals, online help, and installation and conversion software, are developed. It is natural to focus on the hardware and software in the "end product", but you also must develop and account for all ancillary products that are needed in a working system.

  Like the end-product hardware and software components, the supporting products can also be developed in stages and released incrementally to encourage early customer feedback.

- **Perform unit/device testing –** The software and hardware components are thoroughly tested to identify as many defects as possible. The first line of defense is the software developer, who should step through and test every line of code, including all exception and error cases. Additionally, a series of test cases are devised that will exercise the hardware/software component; these test cases are documented in a unit verification plan. After the software is complete and thoroughly debugged by the developer, the test cases are used to test the hardware/software and the results are documented. Identified defects are analyzed and corrected, and testing is repeated until all known defects are either fixed or otherwise resolved. Defect correction may be relatively simple or may include redesign of sections of code that are determined to be error-prone.

  While the developers will conduct their own tests to identify and fix as many defects as possible, experience shows that the test cases and formal tests should be conducted by an independent party, either within the implementation team or from another organization. The reason for this independence is obvious if you look at the objectives of the software developer and the software tester. The primary objective for the tester is to break the software while the primary objective of the developer is the exact opposite – to make the software work. Few individuals can effectively wear both of these hats. The degree of independence between the developer and the tester (i.e., different people in the same department, different departments, or different companies) and the level of formality in unit testing should be commensurate with the criticality of the software and the size of the project.

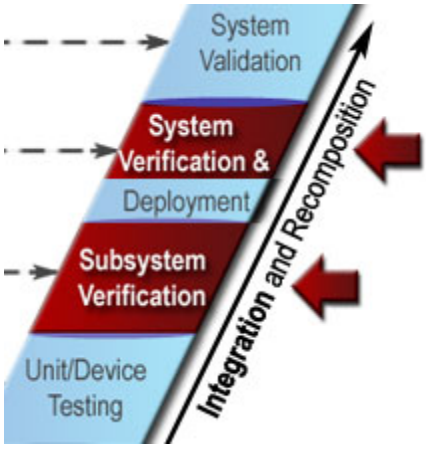
  The unit verification plan should be reviewed to confirm that it will thoroughly test the hardware/software unit. The traceability matrix should be updated to identify the components, test cases, and test status. The testing should be tracked as it progresses to verify that defects are being identified and addressed properly. A testing process that identifies few defects could indicate excellent software or an incomplete or faulty testing process. Use scheduled technical reviews to understand the real project status. You can monitor the rate at which defects are being discovered to estimate the number of remaining defects and make an educated decision about when the hardware/software will be ready for release.

4.6.3 Outputs

This step results in hardware and software components that are tested and ready for integration and verification. Artifacts of the development process are also delivered, including the Software/Hardware Development Plans, development environment documentation, unit test results, change control records, and supporting products and documentation. A record of the technical reviews that were conducted should also be included in the project documentation.

4.7 Integration and Verification

In this step: The software and hardware components are individually verified and then integrated to produce higher-level assemblies or subsystems. These assemblies are also individually verified before being integrated with others to produce yet larger assemblies, until the complete system has been integrated and verified.



.

OBJECTIVES	▪ Integrate and verify the system in accordance with the high-level design, requirements, and verification plans and procedures ▪ Confirm that all interfaces have been correctly implemented ▪ Confirm that all requirements and constraints have been satisfied
INPUT ***Sources of Information***	▪ System Requirements document ▪ High-level design specifications ▪ Detailed design specifications ▪ Hardware and software components ▪ Integration plan ▪ System and Subsystem Verification Plans ▪ Subsystem Acceptance Plans
PROCESS ***Key Activities***	▪ Add detail to integration and verification plans ▪ Establish integration and verification environment ▪ Perform integration ▪ Perform verification
OUTPUT ***Process Results***	▪ Integration plan (updated) ▪ Verification plan (updated) ▪ Integration test and analysis results ▪ Verification results, including corrective actions taken
REVIEW ***Proceed only if you have:***	▪ Documented evidence that the components, subsystems, and system meet the allocated requirements ▪ Documented evidence that the external and internal interfaces are working and consistent with the interface specifications

4.7.1 Overview

In this step, we assemble the system components into a working system and verify that it fulfills all of its requirements. Assembling a puzzle is a nice, simple analogy for this step, but the challenge in an ITS project "puzzle" is that you may find that not all of the pieces are available at the same time, some won't fit together particularly well at first, and there will be pressure to change some of the pieces after you have already assembled them. The systems engineering approach provides a systematic process for integration and verification that addresses the challenges and complexity of assembling an ITS system.

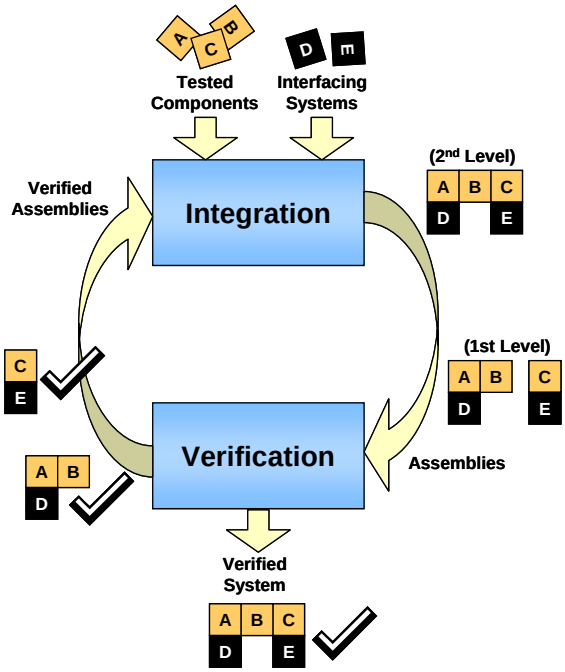


Figure 27: Iterative Integration and Verification

Integration and verification are iterative processes in which the software and hardware components that make up the system are progressively combined into subsystems and verified against the requirements, as shown in Figure 27. This process continues until the entire system is integrated and verified against all of its requirements. This is the opposite of the decomposition that was performed during the Requirements and Design steps, which is reflected in the symmetry between the left and right sides of the "V". Components that are identified and defined on the left side of the "V" are integrated and verified on the right.

In systems engineering, we draw a distinction between verification and validation. *Verification* confirms that a product meets its specified requirements. *Validation* confirms that the product fulfills its intended use. In other words, verification ensures that you "built the product right", whereas validation ensures that you "built the right product". This is an important distinction because there are lots of examples of well-engineered products that met all of their requirements but ultimately failed to serve their intended purpose. For example, a bus rapid transit system might implement a signal priority capability that satisfies all of its requirements. This system might not serve its intended purpose if the traffic network is chronically congested and the buses are never actually granted priority by the signal control system when they need it most. Verification is discussed in this section; system validation is described in Section 4.9.

Integrating and verifying the system are key systems engineering activities. The software and hardware specialists who led the previous step are also involved and provide technical support as their components are integrated into the broader system. Stakeholders should also be materially involved in verification, particularly in the system verification activities. As the verification proceeds from detailed component verification to end-to-end system verification, the implementation team becomes less involved and the stakeholders become more involved. The systems engineering activity provides continuity to the process.

4.7.2 Key Activities

Integrating and verifying the system include basic planning, preparation, and execution steps, described as follows:

- **Add detail to the integration and verification plans –** Recall that integration and verification planning actually began on the left side of the "V". A technique for verifying every requirement was identified as the requirements were specified, and a plan for verifying each requirement was documented. As the system design was defined, the plan for integrating the system components was developed. Detail was added to the general plan when the system was implemented, and the order in which project components and other required resources would be available was defined. The connections between the requirements, system components, and verification techniques were documented in a traceability matrix that was updated as the project development progressed.

  The integration plan defines the order in which the project components are integrated with each other and with other systems. Each integration step includes tests that verify the functionality of the integrated assembly, with particular focus on the interfaces. For less complex projects, the integration plan can be informal. For complex projects, there will have to be careful planning so that the system is integrated in efficient, useful increments consistent with the master schedule.

  The verification plan is expanded into procedures that define the step-by-step process that will be used to verify each component, subsystem, and system against its requirements. For efficiency, test cases are identified that can be used to verify multiple requirements. Each test case includes a series of steps that will be performed, the expected outputs, and the requirements that will be verified by each step in the test case.

The systems engineering analysis requirements identified in FHWA Rule 940.11/FTA Policy Section VI include "identification of … testing procedures", which are the same as the verification procedures that are described here.

Every round of verification that is performed as the system is integrated should be thorough so that defects are identified as early and at as low a level as possible. It is much easier to isolate a defect during component-level verification than it is during system verification, when the entire system is assembled and many different components could be contributing to the problem. To put it in plain language, it is much easier to find the needle before you have assembled a haystack around it.

- **Establish the integration and verification environment –** The tools that will be used to support integration and verification are defined, procured, and/or developed. For complex systems, this could include simulators that are used to mimic operational interfaces, test equipment that is used to inject failures and monitor system responses, etc. The verification environment simulates the operational environment as faithfully as possible and allows portions of the system to be tested before all components are completed.

If test and simulation tools are used to support system verification, then these tools should first be verified with the same care as the system. Verifying a system using a simulator that has not been verified could result in invalid results or compensating errors in which a defect in the end product is masked by a defect in the verification tool.

- **Perform integration –** The system is progressively integrated based on the high-level design and the integration plan. The system components are integrated with each other and with other interfacing systems. Integration tests are used to verify that the components and higher-level assemblies work together properly and do not interfere with one another. Integration tests are used to exercise the interfaces and verify that all interfaces are implemented according to the documentation. Proposed changes to the baseline high-level design, including any required changes to the interface documentation, are identified.

- **Perform verification –** Every requirement is verified using the test cases defined in the verification procedures. System requirements and the related subsystem- and component-level requirements may be verified several times as verification progresses bottoms-up from component to subsystem to system-level verification. For example, a requirement that the system "shall blank a selected dynamic message sign on user command" might be verified at several different levels. The capability of the sign to blank itself would be verified at the dynamic message sign (DMS) component level. The capability of the user interface to accept and relay a "blank sign command" might be tested at the subsystem level, and finally an end-to-end system test would be used to verify that the sign actually blanks on user command. The fully integrated system should be verified at the integrator's facilities before it is installed at the customer's site.

  There are four basic techniques that are used to verify each requirement:

  - *Test*: Direct measurement of system operation. Defined inputs are provided and outputs are measured to verify that the requirements have been met. Typically, a test includes some level of instrumentation. Tests are more prevalent during early verification, when component-level capabilities are being exercised and verified.
  - *Demonstration*: Witnessing system operation in the expected or simulated environment without need for measurement data. For example, a requirement that an alarm is issued under certain conditions could be verified through demonstration. Demonstrations are more prevalent in system-level verification when the complete system is available to demonstrate end-to-end operational capabilities.
  - *Inspection*: Direct observation of requirements such as construction features, workmanship, dimensions and other physical characteristics, and software language.
  - *Analysis*: Verification using logical, mathematical, and/or graphical techniques. Analysis is frequently used when verification by test would not be feasible or would be prohibitively expensive. For example, a requirement that a website support up to 1,000 simultaneous users would normally be verified through analysis.

  As each test case is performed, all actions and system responses are recorded. Unexpected responses are documented and analyzed to determine the cause and to define a plan of action, which might involve repeating the test, revising the test case, fixing the system, or even changing the requirement. Any changes to the test cases, the requirements, or the system are managed through the configuration management process.

It is important to keep strict configuration control over the system components and documentation as you proceed through verification. The configuration of each component and the test-case version should be verified and duly noted as part of the verification results. It is human nature to want to find and fix a problem "on the spot", but it is very easy to lose configuration control when you jump in to make a quick fix. (See Section 5.4 for more information about configuration management.)

As verification proceeds, you normally will have to retest each portion of the system more than once. For example, a new software release that adds new capabilities or fixes previously identified defects may be produced. It is important not only to verify the new features or bug fixes when verifying the new release but also to do *regression testing* to verify that the portion of the software that used to work still does. Regression tests are important because experience shows that old defects may reappear in later releases or that a fix to one part of the software may break another part. For large projects, automated testing tools can be used to automatically run a suite of regression tests to fully test each new software release.

Resist the temptation to scale back verification activities due to budget or schedule constraints. This would be false economizing because defects that slip through will be even more expensive to fix later in the system life cycle. As previously noted, it is most efficient to identify defects early in the verification process. This approach also minimizes the number of issues that will be identified during system verification, which is the most formal and most scrutinized verification step. Issues that occur during a formal system verification that is witnessed by stakeholders can undermine confidence in the system. Be sure to run the system verification test cases beforehand to the extent possible to reduce the risk of unexpected issues during formal system verification.

4.7.3 Outputs

Integration and verification result in a documentation trail showing the activities that were performed and their results. The outputs include:

- **Integration plan (updated) –** This plan defines the sequence of steps that were performed to integrate the system. It also defines the integration tests that were performed to test the interfaces in detail and generally test the functionality of the assembly.
- **Verification plan (updated) and procedures –** This plan documents the approach that was used to verify each of the system and subsystem requirements. The plan identifies test cases that were used to verify each requirement and general processes that were used to conduct the test cases and deal with verification issues. Verification procedures elaborate each test case and specify the step-by-step actions and expected responses. The traceability matrix ties the requirements to the design components and the test cases.
- **Integration test and analysis results –** This is a record of the integration tests that were actually conducted, including analysis and disposition of any identified anomalies.
- **Verification results** – This is a summary of the verification results. It should provide evidence that the system/subsystem/component meets the requirements and identify any corrective actions that were recommended or taken as a result of the verification process.

4.7.4 Examples

Many verification plans that have been developed for ITS projects are available on the Internet. Although they have many different titles – integration test plans, functional test plans, verification plans – they have similar content. For example, Table 13 is an excerpt from a functional test plan that was used to test the Oregon DOT TripCheck website. The script in the table lists each action that the tester should take and the expected result from the system in a step-by-step procedure that tests links in a website navigation panel.

Table 13: Verification Procedure Example: ODOT TripCheck Functional Test Plan (Excerpt)

STEP	INPUT	SCRIPT	EXPECTED RESULT
1	None	Test winter travel links	
1.a		Select Chain Laws	Opens: Pages/RCMap.asp?curRegion=ChainLaws
1.b		Select Traction Tires	Opens: Pages/RCMap.asp?curRegion=TractionTires
1.c		Select Minimum Chain Requirements	Opens: Pages/RCMap.asp?curRegion=MinChainReqs
2	None	Test related links	Each link opens a browser window with an external URL

Table 14 is a verification procedure from a Maryland Chart II Integration Test Plan that includes a bit more background for each test case in a slightly different format.

Table 14: CHART II Integration Test Plan (Excerpt)

Test ID: General 1				
Purpose: To show that a valid username/password is accepted for logging in to CHART II within 15 seconds, and that an invalid combination is rejected. In addition, this test case also demonstrates that the system returns control to the user and the user is not prevented from performing activities in other windows on the desktop. CHART-27, CHART-10, CHART-21, CHART-275, CHART-276, CHART-29, CHART-26			Test Start Date:	
Test Pre-Conditions: This test assumes a valid username and password of a user in the CHART2 system is known.			Test End Date:	
Test Step No.	Test Steps	Expected Behavior	Results As Expected (Y/N)	Comments
1	Click on the Login button on the GUI toolbar.	An hourglass should display immediately, within 5 seconds, till the login window is displayed. Then, you should be prompted for a UserID and password.		
2	Attempt to login with an invalid username or password.	The system should popup an error message indicating that an invalid user ID or password was specified.		
3	Attempt to login with the valid UserID and password.	The system should indicate that the user is logged in by showing Operations Center:Username on the GUI toolbar window.		
4	Click on Navigator	Navigator window is opened.		
5	Click on DMS node	List of DMSs is displayed on the right hand side of the Navigator.		

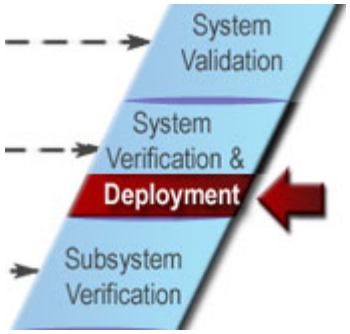
Reports are generated that document the actual results of the verification tests that were performed. Table 15 is a brief excerpt from a test result report for the desktop application that is used by ODOT to update data on the TripCheck website. Each row in the table summarizes the results for each test case. This excerpt was selected because it includes one of the few test cases in this report in which the actual results did not match the expected results. Note that in Test 2, an error occurred that exposed a software defect that had to be fixed. Identification of defects like this before the system is operational is one of the key benefits of a thorough verification process.

Table 15: ODOT TripCheck 2.0 System Test Results (Excerpt)

DESCRIPTION OF TEST	INPUT DATA	EXPECTED RESULTS	ACTUAL RESULTS
1 Enter an incident of type Herbicide Application.	An incident of type Herbicide Application.	Does not appear in TripCheck.	As expected, incident did not go into the transaction table.
2 Enter an incident that is then put on hold.	An incident that is on hold.	Does not appear in TripCheck.	When incident is put on hold a delete transaction is entered in the shadow table. An error occurred with this delete transaction and the incident remained on TripCheck.
3 Put the incident from step 2 back into active status in HTCRS.		Incident is on TripCheck.	As expected.

4.8 Initial Deployment

In this step: The system is installed in the operational environment and transferred from the project development team to the organization that will own and operate it. The transfer also includes support equipment, documentation, operator training, and other enabling products that support ongoing system operation and maintenance. Acceptance tests are conducted to confirm that the system performs as intended in the operational environment. A transition period and warranty ease the transition to full system operation.



OBJECTIVES	▪ Uneventful transition to the new system
INPUT ***Sources of Information***	▪ Integrated and verified system, ready for installation ▪ System Acceptance Plan
PROCESS ***Key Activities***	▪ Plan for system installation and transition ▪ Deliver the system ▪ Prepare the facility ▪ Install the system ▪ Perform acceptance tests ▪ Transition to operation
OUTPUT ***Process Results***	▪ Hardware and software inventory ▪ Final documentation and training materials ▪ Delivery and installation plan, including shipping notices ▪ Transition Plan with checklists ▪ Test issues and resolutions ▪ Operations and maintenance plan and procedures
REVIEW ***Proceed only if you have:***	▪ Formally accepted the system ▪ Documented acceptance test results, anomalies, and recommendations

4.8.1 Overview

Up to this point, the system has been tested primarily in a lab environment. The next step is to ship the system to the actual deployment site(s), install and check it out, and make sure the system and personnel are ready to transition to system operations and maintenance (O&M), as shown in Figure 28.

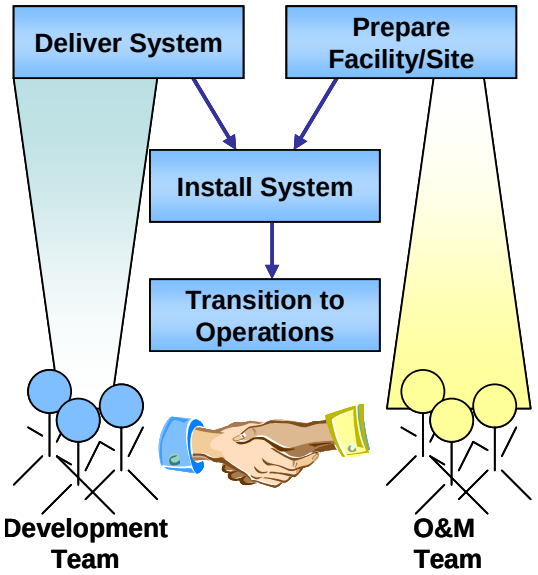


Figure 28: Transition from Development Team to Operations and Maintenance Team

Larger systems may be installed in stages. For example, a closed-circuit television (CCTV) camera network may be built out incrementally over the course of several years and several projects. This may be done to spread the costs across several fiscal years or to synchronize with other construction projects in the region. In other cases, phased deployment may be performed to mitigate risk by deploying the essential core of the system and then adding features over time. If it is necessary to deploy the system in stages, whether due to funding constraints, to mitigate risk, or to synchronize with other projects, it is important to understand the dependencies between successive deployments and to prioritize the projects accordingly.

4.8.2 Key Activities

The following tasks are cooperatively performed to deliver, install, and transition the system to full operational status:

- **Plan for system installation and transition –** This step represents the handoff of the tested system from the project team to the O&M team in the field. The deployment sites must be prepared, the system must be delivered and installed at each site and tested, and O&M staff must be trained. All of this is documented in a System Delivery and Installation Plan. If the new system is replacing an existing system, a smooth transition will be planned and documented in a Transition Plan, including a backup strategy to revert to the existing system just in case the new system does not operate as intended. Each of these plans is further detailed below.

The deployment strategy should take into consideration the complexity of the system, whether it will be deployed at multiple sites, and, if so, the order of the deployments. It might be a good idea to bring up a minimal configuration or a single installation at first and to add further functionality and other sites once the initial installation is operational.

- **Deliver the system –** The system must be physically moved from the development and test labs to the actual deployment site(s). In preparation for this, a complete set of documentation will be developed by the engineering team and coordinated with the site O&M team. This documentation will include all the logistical details for transporting the hardware and software, any facility modifications that may be necessary, personnel assignments for installation, and installation instructions. You should even include shipping details such as the mode of transportation, shipping schedule, and shipping notices. Also include instructions regarding how the system should be handled after

delivery to the operational site. Perhaps it will be moved to a storage area if the site is not yet available, to a staging area, or to the final installation location(s). Key to the systems engineering process is advance planning, and this is especially true for delivery and installation since the system may actually change hands from the engineering team to the system owner.

Until delivery, the system's components – the hardware and software – have been inventoried and under version control by the engineering team. Once delivered, however, ownership may change hands to the agency that will operate and maintain the system. The engineering and operating agencies should come to agreement ahead of time regarding who will maintain the inventory, the version of the software and hardware, any vendor maintenance agreements, and maintenance records to facilitate system delivery.

When the system is delivered, the O&M team should perform an initial inspection and preliminarily accept the system. This might be a formal review of the hardware/software inventory, a check of the documentation, or perhaps a start-up test. More extensive formal acceptance tests will be conducted once the system is fully installed.

- **Prepare the facility –** There are many war stories about the delivery of a system that didn't quite fit the installation site – for example, server racks that wouldn't fit through the equipment room door or CCTV cameras installed on 30-foot poles when DOT bucket trucks could reach no higher than 24 feet. For this reason, part of the planning process is to perform a site survey (including physical, electrical, communications, and lighting components) and prepare a site survey report and site installation plan. There might be some modifications required to the site or facility in order to accommodate the system, or something as simple as additional seating for personnel to operate the system. You should document any necessary site modifications in a site plan, execute the plan, and make sure the facility is ready to receive the system.
- **Install the system –** Following delivery of the system to a site that has been properly prepared and modified as necessary, the system will be installed. Sometimes, problems occur during system installation; make sure you've included a procedure for backing out all or part of the installed system in your installation plan. Following installation, verify that the system was installed correctly using documented verification procedures, also included in the installation plan. You could consider including the system operators in the installation tests since they'll be objective and this will give them a chance to learn more about the system.
- **Perform acceptance tests –** Formal acceptance tests as identified in the System Acceptance Plan are performed by the customer agency following installation. Even if the development is done in-house, there should be a formal decision that the system is ready (i.e., accepted) to go operational.
- **Transition to operation –** After the system has been installed successfully at the final deployment site, the next step is to transition to full operation. For a new, standalone system, this can be a relatively uncomplicated effort. However, if the system must interoperate with other systems, as is the case when installing new AVL software on an existing computer-aided dispatch system, additional integration and testing may be necessary. Or perhaps the new system is replacing an existing system (e.g., an older signal control system). In this case, careful transition planning must take place to minimize the disruption to ongoing signal operations.

  The first step is to create the Transition Plan, which clearly defines how the system will be transitioned to operational status. This plan should include the validation criteria; that is,

how are you going to know that the system is performing correctly once it is operational? It is a good idea to include a series of checklists in the Transition Plan that identify all key pieces that must be in place and working prior to switching over to full operation. If there are still open issues found during system testing (and there likely will be), evaluate each of them to determine whether or not they should be fixed or a work-around created prior to placing the system into full operation. A formal review of the Transition Plan should be held with the implementation team, the operations team, and other key personnel.

When transitioning to operation, especially when replacing an existing system, a contingency back out plan should be included as part of the Transition Plan so that, in the event that the new system does not operate correctly, you can revert to the older system until the issues have been sorted out.

All operations and maintenance staff should be in place and properly trained. The maintenance plans for the system should be reviewed by the O&M team; check to make sure that all maintenance procedures and hardware/software maintenance records are in place and adequate to properly maintain the system.

The operational procedures and any special equipment needed to operate or monitor the system should be ready, tested, and operating correctly. It's a good idea to take some performance measurements on the system at this stage so that you can estimate performance following transition to full operational status. Establish user accounts, initialize databases or files as identified in the Transition Plan, and make sure that all test data has been removed or erased. The system should be set to begin operations.

Some transitions to full operation can be complex, especially when an existing system that many people use is being replaced. Just as we get annoyed when we can't access the Internet for a few hours, users may also become irritated if the system is down for any period of time. You might want to consider planning the transition on a weekend or in the evening if possible to cause the least disruption to system users. Also consider holding a "dry run" so that everyone knows their role during the transition period and performs their assigned task to make the transition as smooth as possible.

Finally, a transition readiness review meeting should be held with the O&M team, the support personnel who are on hand to address last-minute issues, representatives from other interfacing systems, the project sponsor, and other key personnel. Use the checklist in the transition plan to assess system readiness. Only after all checklist items have been declared as ready should the go-ahead be given for the system to transition to full operational status.

Following transition, the team will quickly ramp down to include only the O&M personnel. It might be advisable to keep a few system support personnel around through the validation period so that any issues that arise in the early stages are resolved quickly.

4.8.3 Outputs

The primary output of this step is a fully installed system (in a facility or site modified to meet the system requirements) that has been transitioned to operational status. To support this effort, the following outputs should be generated:

- A hardware and software inventory, under configuration control, that includes versioning information, maintenance records and plans, and other property management information
- Final documentation and training materials
- Delivery and installation plan, including shipping notices

- Updated test plan and procedures
- Transition Plan with checklists
- Test issues and resolutions, and
- Operations and Maintenance Plan and procedures.

4.8.4 Examples

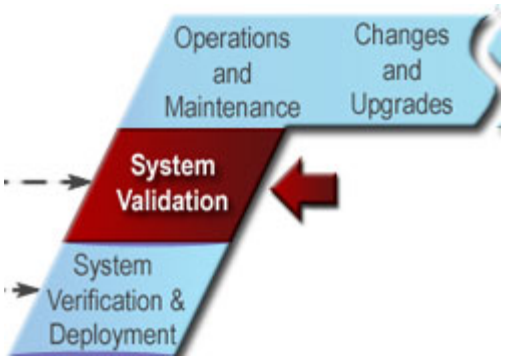
Deployment plans and installation plans can be complex documents for ITS projects that involve significant center and/or field equipment installation. Planning for deployment and installation must begin early in the project for such systems. For example, the Sunol Smart Carpool Lane Joint Powers Agency (JPA) developed a deployment plan as part of its Systems Engineering Management Plan during initial planning for the I-680 Smart Lane Project. This plan defines deployment activities (see Figure 29), roles and responsibilities, deployment personnel by position, installation equipment and tools, system documentation, and installation considerations such as safety, code and industry standards, planning requirements, weather accommodations, and shop drawing submittals. More detailed installation plans will be prepared by the system integrator based on this deployment plan.

- **Pre-installation Activities as follows:**
  - Verify civil and conduit work.
  - Work with the JPA to finalize the Installation Plan, Installation Schedule, and other deployment documents.
  - Ensure that all safety procedures are in place.
  - Secure Caltrans Encroachment Permit.
- **Roadside Equipment Installation as follows:**
  - FasTrak Antennas and Readers.
  - Tolling Zone Lane Controllers.
  - Enforcement Beacons.
  - Vehicle Detection System (VDS) Equipment.
  - Closed Circuit TV (CCTV) Equipment.
  - Communications Network Equipment.
  - Other equipment as identified in the RFP.
- **TDC Equipment Installation as follows:**
  - Trip Processor Hardware and Software.
  - Customer Service Representative (CSR) Workstations.
  - JPA Smart Lane website.
  - Interface to the Bay Area Toll Authority (BATA) Regional Customer Service Center (RCSC).
  - Interface to the Caltrans TMC.
  - Interface to the CHP Enforcement Equipment.
  - Other equipment as identified in the RFP.
- **Post-installation Activities as follows:**
  - Verify that all of the equipment and software is installed properly
  - Verify that each internal subsystem communicates properly to each other.
  - Verify that all installed equipment and software operates properly.

Figure 29: I-680 Smart Lane Project Deployment Activities Overview

4.9 System Validation

In this step: After the ITS system has passed system verification and is installed in the operational environment, the system owner/operator, whether the state DOT, a regional agency, or another entity, runs its own set of tests to make sure that the deployed system meets the original needs identified in the Concept of Operations.



OBJECTIVES	▪ Confirm that the installed system meets the user's needs and is effective in meeting its intended purpose
INPUT ***Sources of Information***	▪ Concept of Operations ▪ Verified, installed, and operational system ▪ System Validation Plan
PROCESS ***Key Activities***	▪ Update Validation Plan as necessary and develop procedures ▪ Validate system ▪ Document validation results, including any recommendations or corrective actions
OUTPUT ***Process Results***	▪ System Validation Plan (update) and procedures ▪ Validation results
REVIEW ***Proceed only if you have:***	▪ Validated that the system is effectively meeting its intended purpose ▪ Documented issues/shortcomings ▪ Established ongoing mechanisms for monitoring performance and collecting recommendations for improvement ▪ Made modifications to the Concept of Operations to reflect how the system is actually being used

4.9.1 Overview

A few readers may be surprised to see that there is another step in the "V" between initial deployment and operations and maintenance. After all, in the last few chapters we have already verified that the system meets all of its requirements, installed the system and trained the users, and the customer has successfully conducted acceptance tests and formally accepted the system. Aren't we done?

The answer is: yes and no. Yes, the system has been put into operation and is beginning to be used for its intended purpose. No, we aren't done. Now that the system is beginning to be used in the operational environment, we have our first good opportunity to measure just how effective the system is in that environment (i.e., system validation).

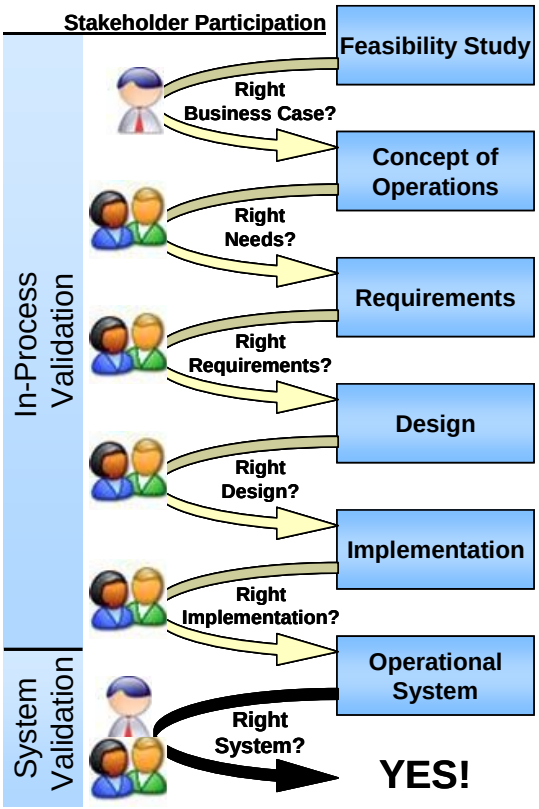


Figure 30: Validation Occurs Throughout the Systems Engineering Process

In systems engineering, we draw a distinction between *verification* and *validation*. Verification confirms that a product meets its specified requirements. Validation confirms that the product fulfills its intended use. The majority of system verification can be performed before the system is deployed. Validation really can't be completed until the system is in its operational environment and is being used by the real users. For example, validation of a new signal control system can't really be completed until the new system is in place and we can see how effectively it controls traffic.

Of course, the last thing we want to find is that we've built the wrong system just as it is becoming operational. This is why the systems engineering approach seeks to validate the products that lead up to the final operational system to maximize the chances of a successful system validation at the end of the project. This approach is called *in-process validation* and is shown in Figure 30. As depicted in the figure, validation was performed on an ongoing basis throughout the process:

- The business case for the project was documented and validated by senior decision makers during the initial feasibility study.
- User needs were documented and validated by the stakeholders (i.e., "Are these the right needs?") during the Concept of Operations development.
- Stakeholder and system requirements were developed and validated by the stakeholders (i.e., "Do these requirements accurately reflect your needs?").
- As the system was designed and the software was created, key aspects of the implementation were validated by the users. Particular emphasis was placed on validating the user interface design since it has a strong influence on user satisfaction.

Since validation was performed along the way, there should be fewer surprises during the final system validation that is discussed in this step. The system will have already been designed to meet the user's expectations, and the user's expectations will have been set to match the delivered system.

4.9.2 Key Activities

The system validation is the responsibility of the system owner and will typically be performed by the system users.

- **Update the Validation Plan and develop procedures –** An initial Validation Plan was created at the same time as the Concept of Operations earlier in the life cycle (see Section 4.3). The performance measures identified in the Concept of Operations forced early consideration and agreement on how system performance and project success would be measured. A Validation Plan was prepared that defined the consensus validation approach and the outcomes that should be measured.

It is important to think about the desired outcomes and how they will be measured early in the process because some measures may require data collection before the system is operational to support "before and after" studies. For example, if the desired outcome of the project is an improvement in incident response times, then data must be collected before the system is installed to measure existing response times. This "before" data is then compared with data collected after the system is operational to estimate the impact of the new system. Even with "before" data, determining how much of the difference between "before" and "after" data is actually attributable to the new system is a significant challenge because there are many other factors involved. Without "before" data, validation of these types of performance improvements is impossible.

In addition to objective performance measures, the system validation may also measure how satisfied the users are with the system. This can be assessed directly using surveys, interviews, in-process reviews, and direct observation. Other metrics that are related to system performance and user satisfaction can also be monitored, including defect rates, requests for help, and system reliability. Don't forget the maintenance aspects of the system during validation – it may be helpful to validate that the maintenance group's needs are being met as they maintain the system.

Detailed validation procedures may also be developed that provide step-by-step instructions on how support for specific user needs will be validated. At the other end of the spectrum, the system validation could be a set time period when data collection is performed during normal operations. This is really the system owner's decision – the system validation can be as formal and as structured as desired. The benefit of detailed validation procedures is that the validation will be repeatable and well documented. The drawback is that a carefully scripted sequence may not accurately reflect "intended use" of the system.

- **Validate the system –** The system is validated according to the Validation Plan. The system owner and system users actually conduct the system validation. The validation activities are documented and the resulting data, including system performance measures, are collected. If validation procedures are used, then the as-run procedures should also be documented.

The measurement of system performance should not stop after the validation period. Continuing performance measurement will enable you to determine when the system becomes less effective. The desired performance measures should be reflected in the system requirements so that these measures are collected as a part of normal system

operation as much as possible. Similarly, the mechanisms that are used to gauge user satisfaction with the system (e.g., surveys) should be used periodically to monitor user satisfaction as familiarity with the system increases and expectations change.

Frequently, the way in which the system is used will evolve during initial system operation. Significant departures from anticipated procedures should also be noted and documented in the Concept of Operations. For example, consider an HOV reversible lane facility that uses system detectors to verify that all vehicles have exited the facility. During system operation, the agency may find that the reliability of system detectors is not as high as anticipated. To compensate, the agency adjusts its operating procedures to perform a physical tour of the facility prior to opening it up in the opposite direction. The agency should amend its ConOps to reflect this new way of operating the HOV facility.

- **Document validation results –** The data resulting from the system validation is analyzed, and a validation report is prepared that indicates where needs were met and where deficiencies were identified. Deficiencies can result in recommended enhancements or changes to the existing system that can be implemented in a future upgrade or maintenance release. If an evolutionary development approach is used, the validation results can be a key driver for the next release of the product. (See Section 6.2.2 for more information on development strategies.)

Deficiencies of the project development process should also be reviewed to determine where the process may have fallen down, so that an improved process can be used on the next project. Without worrying about attribution to individuals, determine how a significant deficiency slipped through the process. Were the needs not properly specified? Were requirements incorrectly specified based on the needs? If so, were there opportunities for the stakeholders to walk through the requirements and identify the problem? A "lessons learned" review of the project development process at the conclusion of the system validation can be very valuable.

4.9.3 Outputs

System validation should result in a document trail that includes an up-to-date Validation Plan; validation procedures (if written); and validation results, including disposition of identified deficiencies. There are several industry and government standard outlines for validation plans, including IEEE Standard 1012[18], which is intended for software verification and validation but is also applicable to broader system verification and validation. Note that this standard covers both verification and validation plans with a single outline.

4.9.4 Examples

There are few good examples of system validations that have been performed for ITS projects. Some of the best examples are evaluations that have been performed for field operational tests (FOTs), and other evaluations that have looked in detail at the benefits of ITS. For example, the evaluation of the ORANGES Electronic Payment Systems FOT initially identified system goals and then related them to quantitative and qualitative performance measures, as shown in Table 16. Each of the performance measures was then evaluated, in many cases using before-and-after study techniques, to determine whether the system goals were achieved. Figure 31 shows results supporting the transponder market penetration goal (Goal 2). This evaluation report is a good example of many validation

[18]IEEE 1012-2004: IEEE Standard for Software Verification and Validation.

techniques, including the collection of baseline data, before-and-after studies, statistical analysis, evaluation of other causal factors, and interview and survey activities.

Table 16: ORANGES Evaluation Goals and Performance Measures

FOT Evaluation Goal	Measure
1. Increase parking revenue	• Revenue received
2. Increase transponder market penetration	• Number of smart card users that newly acquire a transponder
3. Reduce transaction times	• Average transaction times
4. Increase prepaid revenue share	• % revenue prepaid
5. Reduce monthly pass distribution costs	• Procurement, inventory, delivery, commissions for any conventional passes made available on smart cards
6. Increase automated payment equipment uptime	• % equipment availability
7. Cardholders use the joint account	• Card use profiles • Average prepaid balance • Modal use profile
8. Understand customer perceptions • General benefits • Ease of use • Convenience of revaluing	• Customer feedback
9. Understand operations/maintenance staff perceptions, including: • General benefits • Reduced payment disputes • Reduced transfer abuse • Ease of customer use • Maintenance	• Operations/maintenance staff feedback
10. Understand planning/management staff perceptions, including: • General benefits • More comprehensive data collection	• Planning/management staff feedback
11. Understand interagency perceptions, including: • General institutional issues • Interagency collaboration	• Partnership feedback

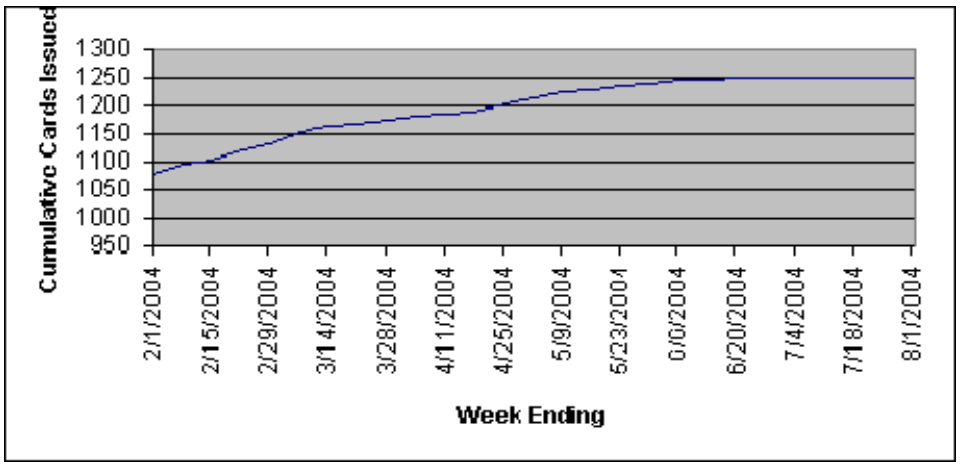


Figure 31: ORANGES Evaluation – Cumulative Transponders Issued

4.10 Operations and Maintenance

In this step: Once the customer has accepted the ITS system, the system operates in its typical steady state. System maintenance is routinely performed and performance measures are monitored. As issues, suggested improvements, and technology upgrades are identified, they are documented, considered for addition to the system baseline, and incorporated as funds become available. An abbreviated version of the systems engineering process is used to evaluate and implement each change. This occurs for each change or upgrade until the ITS system reaches the end of its operational life.

OBJECTIVES	▪ Use and maintain the system over the course of its operational life
INPUT ***Sources of Information***	▪ System requirements (operations/maintenance requirements) ▪ Operations and Maintenance Plan and procedures ▪ Training materials ▪ Performance data ▪ Evolving stakeholder needs
PROCESS ***Key Activities***	▪ Conduct Operations and Maintenance Plan reviews ▪ Establish and maintain all operations and maintenance procedures ▪ Provide user support ▪ Collect system operational data ▪ Change or upgrade the system ▪ Maintain configuration control of the system ▪ Provide maintenance activity support
OUTPUT ***Process Results***	▪ System performance reports ▪ Operations logs ▪ Maintenance records ▪ Updated operations and maintenance procedures ▪ Identified defects and recommended enhancements ▪ Record of changes and upgrades ▪ Budget projections and requests
REVIEW ***Proceed only if you have:***	▪ Demonstrated that the system has reached the end of its useful life

4.10.1 Overview

Now that the ITS system is up and running, it enters a "steady state" period that lasts until the system is retired or replaced. During this period, operators, maintainers, and users of the system may identify issues, suggest enhancements, or identify potential efficiencies. New releases of hardware and software will be installed and routine maintenance will be performed. Approved changes and upgrades are incorporated into the system baseline using the systems engineering process, as shown in Figure 32. O&M personnel might also identify process changes that may streamline O&M activities. All changes to the processes should be documented.

Successful operations and maintenance of the system will lead to customer and user satisfaction; for example, the CCTVs will be online and fully functional at all times; rush-hour drivers will be able to obtain accurate, up-to-the-minute speed, accident, and construction reports before they head out the door; and transit vehicles will arrive on time. This is when the system benefits are realized.

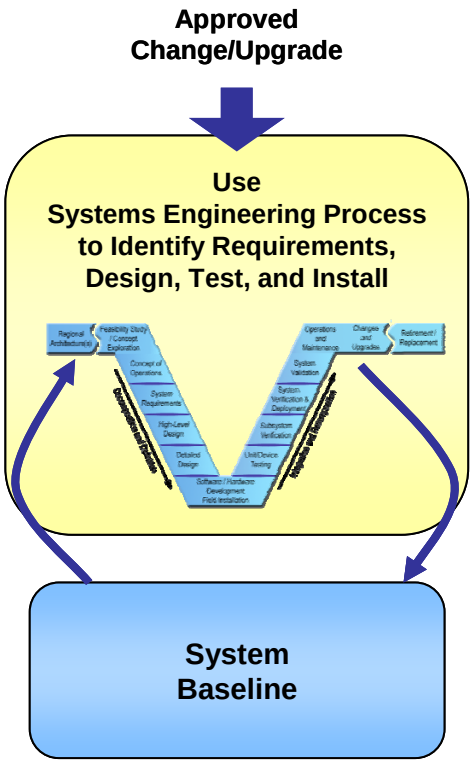


Figure 32: Changes/Upgrades Performed Using Systems Engineering

4.10.2 Key Activities

In most systems, operations and maintenance is where the lion's share of life-cycle costs are incurred. The key activities are performed periodically unless a change is considered severe and affects system performance dramatically.

- **Conduct Operations and Maintenance Plan reviews** – Operations and maintenance roles and required resources are defined in the Concept of Operations (see Section 4.3) and are refined as the system is developed. At this point, operations and maintenance personnel and the system sponsor should all be in agreement on the level of support to be provided with regard to staffing, frequency of technology refreshes (e.g., how often the software or hardware is upgraded to a new release), performance monitoring and reporting, processes for handling identified issues, and level of support provided to the end user.
- **Establish and maintain operations and maintenance procedures** – Although the processes to be used for identifying, tracking, resolving, and recording all system issues will have been established during the initial deployment step, specific detailed procedures will be further developed and maintained as efficiencies are identified. All personnel will be trained in the procedures and are responsible for their use.
- **Provide user support** – End users of your system, whether they are traffic management center operators or a person whose farecard is not working in the new farecard reader, need to be able to contact someone for user support. This support could be handled by a formal call center or perhaps only a person who performs the task during spare time via e-mail, depending on the type and complexity of the system to be supported. Either way, the user support personnel should be properly trained, should document all calls from

initiation through final resolution, and should have access to system experts if needed. These user support personnel should also provide periodic updates on user inquiries and resolutions.

A database that holds information about all user support inquiries can help you to review the types of calls that were received and to notice trends. If there seems to be a recurring problem or confusion about some aspect of the system, it could mean that a system modification should be considered.

- **Collect system operational data –** During earlier phases in the system life cycle, you will have determined how to collect system performance metrics and will have used the performance data to validate the system (see Section 4.9.2). During operations and maintenance, you should collect sufficient performance data to help you determine how well the system is operating over time. For example, in a transit management center, the on-time arrival performance data might be collected from the AVL software. If you are providing a website that displays incident and speed information, a positive response from a user who is asked whether the information was "helpful and accurate" could be collected. Feedback from operators and travelers will provide a measure of customer satisfaction. In-process reviews can be held periodically to review collected metrics, assess system performance, and identify potential system improvements.
- **Change or upgrade the system –** Like any computer system, planning for change and upgrade of your ITS system may start the day that the system is turned on. The system will evolve over its lifetime as stakeholder priorities change and technology advances. Changes can also result from user-reported issues and recommendations and from system improvements identified from the review of operational data. If you decided to deploy only part of the system during the initial deployment step, this is when you'll incrementally add the rest of the system – whether it's additional functionality or equipment at additional sites (e.g., additional CCTV deployment). (See Section 6.2.2 for more information on incremental development.)

  All proposed changes should be prioritized and will require careful cost estimates, schedules, planning, testing, and coordination with operations and maintenance prior to installation. Each approved change will require a new system release level and should be coordinated between the O&M and development teams.

Each potential change to the system should be assessed by the affected stakeholders and the project sponsor to determine whether or not it should be incorporated. Before approving the change, you should clearly understand and document the effect that it will have on other parts of the system, on the operation of the system as a whole, and on the maintenance of the system. If you make this assessment early on by following the systems engineering process, you won't discover a problem months later in the lab when the impact on the schedule and budget will be significantly higher.

Changes are approved and managed using the configuration management process defined in Section 5.4. You should use the systems engineering process, from Concept of Operations through design, verification, and installation, to add any approved change to the system. Basically, each change requires another, possibly abbreviated, pass through the "V". Approved changes are typically aggregated into builds or releases, although you may want to introduce particularly complex changes individually.

Each build or release should be subjected to thorough verification testing prior to installation. There are many stories of "changes that affected only a few lines of code" that ultimately resulted in operational failure. It is important to run regression tests that

verify that a seemingly minor change in one part of the system didn't have an unexpected effect on another part of the system. Statements like "I didn't change that area so there is no need to test it" should be a red flag.

In many cases, the development and test lab that was available during the initial system development may not be available once the system has been deployed. (It might even be the system that was deployed!) Therefore, it's common to establish a test environment to test software product upgrades or minor fixes without interfering with the current operational system.

- **Maintain configuration control of the system –** The deployed system is under configuration control, so every time the system changes, even if only a minor software patch was added, the system baseline must be updated. This means that all documentation, databases, and any other operational data must also be updated. A project library should be established that includes the latest baseline versions of all project documentation. (Section 5.4 includes more information on configuration management.)

This is one area where state of the practice lags a bit in ITS. It is common for agencies to require good configuration management practices during system development but to lose configuration control after the system is delivered. For example, if you want to know the configuration of a field controller at a particular location, you will have to take a trip to the field and have a look inside the cabinet at many agencies.

- **Provide maintenance activity support –** A fully functional system should be available for use at all times except for minimal prescheduled maintenance periods during off-hours. Maintenance records on all equipment should be documented. Sufficient equipment, materials, supplies, and spares should be in place, inventoried, and working properly. The suggested quantities for each of these items should be included in the maintenance plan prior to transitioning to full operational status.

Consider using a database tool or a similar property management application to help you keep track of all equipment, together with maintenance records, maintenance schedules, and so forth. Check it weekly and schedule the maintenance required.

4.10.3 Outputs

The current system configuration, including hardware, software, and operational information, must be documented and maintained. A complete record of all system changes should also be documented and readily available. This is especially helpful when trying to duplicate an anomaly identified by a user or operator.

System performance reports should be generated, both from any installed automated performance monitors and from user-support calls received. Trend analysis reports can be generated and reviewed to identify system deficiencies.

- Document Maintenance and Operations Activities
- Develop and Maintain a Cost Database for Maintenance and Operations
- Analyze Maintenance and Operations Requirements
- Analyze Staffing Requirements for Maintenance and Operations
- Develop a Training Program for Maintenance and Operations Personnel
- Prioritize Maintenance Needs
- Develop and Maintain a Spare Parts Inventory
- Develop a Maintenance Plan
- Develop an Operations Manual

Figure 33: Kentucky ITS M&O Plan Best Practices

4.10.4 Examples

Operations and Maintenance Plans

The Kentucky Transportation Center developed a *Maintenance and Operations Plan for ITS in Kentucky* that provides recommendations for supporting and coordinating ITS maintenance and operations activities throughout the Kentucky Transportation Cabinet. It inventories ITS equipment and systems, identifies national best practices for operations and maintenance (see Figure 33), assesses current maintenance and operations practices in Kentucky, and makes recommendations. Many of the recommendations and best practices identified in the report will be relevant to other agencies. This broad agency-wide plan complements the detailed procedures that are used to operate and maintain individual systems.

Operations and Maintenance Procedures

Operations and maintenance procedures are detailed and don't make particularly good reading unless you actually operate and maintain one of these systems, in which case they are indispensable. These manuals will be subject to relatively frequent changes as personnel will find errors and new and better ways to operate and maintain the system. A short excerpt from the CHART II O&M Procedures is shown in Figure 34.

A.1.5 Dialogic board installation/config

A.1.5.1 Setting the board identification number

- If more then one board is to be used, each board must have a unique identification number. Turn the rotary switch (SW100) to select one of the 16 board ID settings.

A.1.5.2 Attaching the PEB Terminator (XTERM) (PEB MODE ONLY)

- Attaching the PEB terminator to the XTERM socket. To terminate the voice resource board, use the resource module position. Insert the terminator in the resource module position, make sure that Pin 1 as indicated by black dot is positioned in the upper right corner in the XTERM socket.

A.1.5.3 Installing Dialogic Adapter

- Insert the Dialogic adapter in system ISA slot

Figure 34: CHART II O&M Procedures

Change and Upgrade Plans

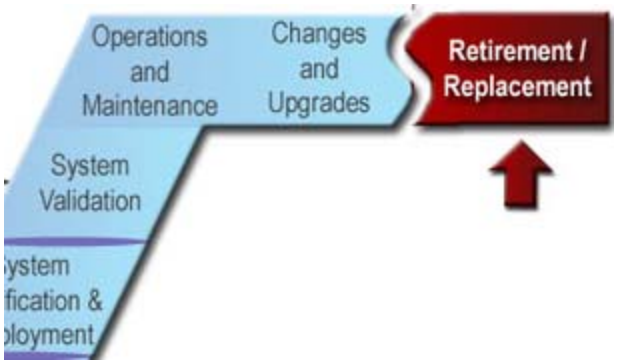
Metro Transit in Seattle, Washington, upgraded its existing Transit AVL system to support transit traveler information systems as part of the Smart Trek program. To support this upgrade, detailed cost estimates were made based on systems engineering analysis of the AVL enhancements that would be required to support the traveler information objectives of the Smart Trek project. The estimate is shown in Table 17.

Table 17: Metro Transit AVL System Upgrade

Equipment Description	Non-Recurring Costs	Recurring Costs
ARI CPU Boards (Quantity:1360)	$ 442,570	
Modem Boards (Quantity: 1360)	$ 247,410	
Termination Boards (Quantity:1360)	$ 26,206	
Labor to Retrofit MDU Units (Quantity: 1360)	$ 34,284	
Delivery and Pickup of MDU and Cases (Quantity: 1360)	$ 3,834	
Sales Tax on Above	$ 70,000	
AVL Software	$ 250,000	
MDU Test Tool (Software and Hardware)	$ 55,200	
Contract Management Labor	$ 40,660	
Basic System Maintenance (Hardware)		$ 10,000
Basic System Maintenance (Labor - 1 FTE)		$ 125,000
ISP Costs (3 lines)		$ 2,700
Total	**$ 1,170,164**	**$ 137,700**

4.11 Retirement/Replacement

In this step: Operation of the ITS system is periodically assessed to determine its efficiency. If the cost to operate and maintain the system exceeds the cost to develop a new ITS system, the existing system becomes a candidate for replacement. A system retirement plan will be generated to retire the existing system gracefully.



OBJECTIVES	▪ Remove the system from operation, gracefully terminating or transitioning its service ▪ Dispose of the retired system properly
INPUT ***Sources of Information***	▪ System requirements (retirement/disposal requirements) ▪ Service life of the system and components ▪ System performance measures and maintenance records
PROCESS ***Key Activities***	▪ Plan system retirement ▪ Deactivate system ▪ Remove system ▪ Dispose of system
OUTPUT ***Process Results***	▪ System retirement plan ▪ Archival documentation
REVIEW ***Proceed only if you have:***	▪ Planned the system retirement ▪ Documented lessons learned ▪ Disposed of the retired system properly

4.11.1 Overview

Systems are retired from service for a variety of reasons. Perhaps the system is being replaced by a newer system, or maybe the Concept of Operations has changed such that stakeholder needs are going to be met in an alternative manner that will no longer require use of the system. For example, the emergency call boxes that currently dot many of the nation's highways are beginning to be retired because their usage has decreased dramatically due to widespread use of cell phones. Many of the first-generation ITS systems are twenty years old and approaching the end of their useful life. Regardless of the reason for the retirement of the system, you should make sure that everything is wrapped up (e.g., hardware and software inventory identified for disposal is audited, final software images are captured, and documentation is archived), the contract is closed properly, and the disposal of the system is planned and executed.

4.11.2 Key Activities

This step represents the end of the system life cycle – the retirement and disposal of the ITS system. An important characteristic of the systems engineering process is the planning of all events; the retirement of the system should be planned as well.

The retirement plan should include a complete inventory of all software and hardware, final system and documentation configurations, and other information that captures the final operational status of the system. This should include identification of ownership so that owners can be given the option to keep their equipment and use it elsewhere. It should also include how the system and documentation will be disposed of, including an assessment and plan if special security measures should be in place or if there are environmental concerns that might dictate the site of disposal. You should also plan to erase the content of all storage devices to protect any personal data that might pose privacy concerns. The retirement plan should be reviewed and approved by all parties, including the agency or contractor providing O&M, the owner of the system (if different), and other key personnel.

If the system to be retired is not documented as well as it should be, steps are taken to capture all necessary data and reverse engineer interfaces and any system configuration information that is needed to support a replacement system. Existing databases may need to be exported and translated into a format suitable for the replacement system.

The next activity is to execute the retirement plan and record the results. It's also a good idea to hold a "lessons learned" meeting that includes suggested system improvements. All recommendations should be archived for reference in future system disposals. The O&M contract should be officially closed out if one exists.

4.11.3 Outputs

A system retirement plan will be generated that describes the strategy for removing the system from operation and disposing of it. Its execution will result in the retirement of the ITS system. The final system configuration, including hardware, software, and operational information, will be documented and archived, together with a list of "lessons learned".

5 ITS PROJECT MANAGEMENT PROCESSES

In addition to the process steps identified in the "V", there are several project management and control activities that are essential in order for a project to be successful. The project planning, project monitoring and control, risk management, and configuration management processes shown in Figure 35 all support systems engineering. These activities are discussed briefly in this chapter; a list of additional resources may be found in Chapter 7.

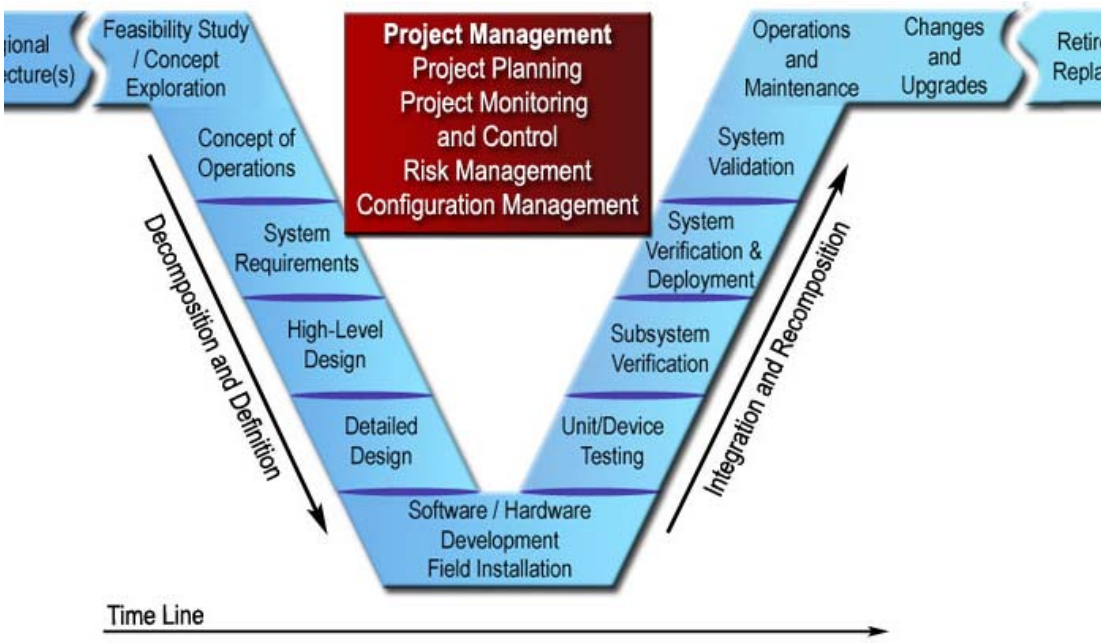


Figure 35: Project Management Activities Cut Across All Steps of the "V"

5.1 Project Planning

Project planning lays out the activities, resources, budget, and timeline for the project. This effort, which begins early in the project life cycle, results in the creation of two major plans, the Project Plan (PP) and the Systems Engineering Management Plan (SEMP). Both of these documents should be written in such a way that a newcomer to the project team can understand the type and scope of the project, the responsibilities of the major players, the staffing, the schedule and budget, and the processes that will govern the project. There are typically several additional plans (e.g., Risk Management Plan and Configuration Management Plan) that will either become appendices to the PP or SEMP or will stand on their own, depending on the size and complexity of the project.

5.1.1 Project Plan

The Project Plan (PP) documents how the project will be managed and controlled from a programmatic standpoint. It identifies the detailed work plans for both administrative and technical tasks. For each project task, the PP documents what is to be done, by whom, with what funds, when, how (processes to be used), and dependencies.

When contracting is involved, the PP will either be created by the agency prior to contracting for the project or will become the first output of the effort, created by the contractor immediately after beginning the project.

The Project Plan should include the purpose and overview of the project, task descriptions, resources and budget allocated to each task, deliverables, and project schedule. It should also include a budget plan that estimates annual/monthly costs and identifies where funds will come from, a project organization, as well as roles and responsibilities relative to project execution. If there are any Integrated Product Teams (IPTs) that will be utilized on the project, their mission and membership should be described. The PP should include a list of all documents to be generated, key milestones, and formal meetings. Process descriptions or flowcharts that will govern how the project is controlled should be appended. It's helpful to create a table that identifies the owner or lead for each process and document as well as those organizations that provide a supporting role. All tools to be used to manage the project should be listed, along with key project performance measures that will be tracked to monitor schedule and budget performance. In general, the PP is the "how to" manual for managing the project's implementation, and as such it should be formally reviewed and approved by all major stakeholders prior to the start of the project.

5.1.2 Systems Engineering Management Plan

The Systems Engineering Management Plan (SEMP) is the top-level plan for managing the systems engineering effort to produce a final operational system from initial requirements. Just as the PP defines how the overall project will be executed, the SEMP defines how the engineering portion of the project will be executed and controlled. It describes how the efforts of system designers, test engineers, and other engineering and technical disciplines will be integrated, monitored, and controlled during the complete life cycle. For a small project, the SEMP might be included as part of the PP document, but for any project of greater size or complexity a separate document is recommended.

The information contained within a SEMP can be organized in different ways, but in general it should include an introductory section (including system description, top-level schedule, and relevant documents), technical planning and control, systems engineering processes tailored specifically for the project, and plans for coordinating the efforts of multiple engineering disciplines to accomplish the project tasks. Make sure that the SEMP and the PP are consistent – it's fine to reference the PP in the SEMP and vice versa.

Technical Planning and Control – This section describes how the project will be controlled from a systems engineering point of view. It includes the engineering organization and responsibilities, identification of technical and performance monitoring reviews to be held during the project life cycle, the system test strategy, technical performance measurements to be monitored, the configuration and data management strategy (see Section 5.4 for further details), the risk management strategy (see Section 5.3 for further details), and identification of any critical items that may require special risk management.

In addition, there are a host of other plans that should be created near the beginning of the project life cycle. Depending on the size and complexity of the project, plans may be small and included as part of the SEMP or they may be referenced by the SEMP as standalone documents. Minimally, the SEMP should identify all of the relevant project documents. Plans to be described or referenced in the SEMP include:

- Interface Control Plan, describing the nature of external interfaces and responsibilities of organizations on each side of the interface.

- System Integration Plan, describing the strategy for how the software and hardware that comprise each subsystem will be integrated with other subsystems to form the overall system and including the dependencies and operational capabilities at each stage in the integration.

- System and Subsystem Verification Plan, describing how requirements will be verified for the system and each subsystem. This plan may also include the test lab environment(s), tools required, and dependencies.
- System Validation Plan, describing the approach that will be used to validate the project delivery.
- Software and Hardware Development Plans, describing the facilities, tools, and processes to be used to produce the project's software and hardware including development of custom software/hardware and procurement of commercial software/hardware products.
- Installation Plan, describing logistics for system deployment and installation procedures.
- Operations and Maintenance Plan, describing the organization, staffing, and processes for operating the deployed system, including maintenance, technical refresh plans, enhancement process, and procedures.
- Other plans, such as a Training Plan, a Safety Plan, or a Security Plan, may also be needed to address special issues of the project.

Systems Engineering Processes – This section of the SEMP describes the processes to be used for execution of the various systems engineering steps covered in Chapter 4, as tailored for your project. This is a good place to include a discussion of the project's approach to meeting the requirements of FHWA Rule 940.11/FTA Policy Section VI.

This section should include a definition of all high-risk areas, including critical technologies that might pose some challenge for your system. The SEMP will include a list of the tools that will be employed during the development (e.g., a requirements traceability tool).

Coordination of Engineering Disciplines – This section describes how the various inputs into the systems engineering effort will be integrated and how appropriate disciplines will be coordinated with that effort. In a complex project, there will be multiple engineering disciplines contributing to the success of the project. For example, for projects that have a user interface, operability/human engineering will provide input during the development cycle to ensure that the design is user-friendly and intuitive. If system reliability is a major issue, specialists should assess the design to make sure that it will meet performance requirements. In the SEMP, the dependencies between these various engineering disciplines and the project life cycle will be documented. This will help the systems engineer to make sure that input is solicited from each engineering discipline at the appropriate time and that the right people are at the various technical reviews.

5.2 Project Monitoring and Control

The plans discussed in the PP and in the SEMP include the steps that will be taken to monitor and control the project from a systems engineering standpoint. Two aspects of this monitoring and control, project tracking and project technical reviews, are discussed next.

5.2.1 Project Tracking

The PP and the SEMP define the tasks and schedule for the project and the processes that will be followed to produce the deliverables. Once the project is underway, how can you track progress against the plan? When should you start to worry that the project is veering off track? Is the project on track as long as cost and schedule are meeting the plan?

It's tough to answer any of these questions without creating some means of measuring progress. Metrics can help you to

If you can't measure it, then you can't manage it.

Peter Drucker

track progress and more importantly track problems on the project. Performance measures, which are formed by combining metrics, may be used to indicate progress or achievement and may be programmatic or technical. A few examples of performance measures are shown in Table 18.

Table 18: Example Performance Measures

Performance Measures		
Technical		Programmatic
Development Progress	System Performance	
# of requirements defined / # of requirements predicted	Transit vehicle maintenance database usage / # disk space allocated	# of tasks performed / # of planned tasks scheduled
# of SW modules completed / total # of planned installations	# of times traffic signal equipment reports faults / Total # traffic signal equipment faults	Cost as of today / Budget as of today
# of field sites installed / Total # of planned installations	Failure rate of DMS equipment / Predicted failure rate	
# of acceptance tests passed / Total # of acceptance tests		

Programmatic performance measures are primarily concerned with spending and schedule progress and are documented in the PP. *Earned value measures* are one of the best ways to accurately measure cost and schedule performance. Earned value provides a measurement of work accomplishment compared with resource expenditure (cost or effort). A detailed discussion of this technique can be found in many project management resources, including the *Project Management Book of Knowledge* (PMBOK) and the NHI Project Management Course. (Consult Chapter 7 for resources.)

Technical performance measures are documented in the SEMP and fall into two general categories:

- Outputs of the systems engineering process steps (e.g., the number of requirements changes after the requirements have been baselined), or
- Performance or effectiveness of the system being developed (e.g., incident response time for an incident management system)

Metrics and measures should be defined early in the project and tracked either periodically (e.g., monthly) or at defined milestones. Monitoring the performance measures on a regular basis will help you to notice trends and to predict and head off potential issues before they grow into large problems. By using programmatic measures, you will be better able to adjust the task schedule and move resources as needed to minimize the impact to other tasks. By monitoring technical measures, you will notice trends that will help point to inefficiencies in the system design. Both programmatic and technical measures are also useful in identifying points in your processes that could be improved.

5.2.2 Project Reviews

Project reviews provide a structured and organized approach to reviewing project products to determine if they are fit for their intended use. These reviews are a primary method of communicating progress, monitoring risk, and transferring products and knowledge between project team members. The reviews often occur at the completion of a "V" process step and

represent *decision points* that must be passed successfully before moving to the next step in the process.

Note that there are programmatic project reviews that are focused on budget, schedule, resourcing, and other program control topics. Technical reviews center more on the project development aspects of the program, and these are the focus of the following discussion.

Project reviews should be done in partnership with the system developer and should not be treated as an audit to discover contract noncompliance. The reviews should be collaborative with no fingerpointing. Discuss ahead of time what should be done if an impasse is reached during a project review. "What if" planning in a risk management plan or a review-specific plan before an impasse is reached can help to keep the project moving forward productively.

Sometimes it seems that the focus at major project reviews is on impressive presentations that put the best face on the project. Remember that the review is not convened to assess the quality of the presentations. The real intent is to evaluate objectively the output of the current step and to assess honestly whether the team is ready to progress to the next step.

Make sure that the right people are in the room when you hold the project review. For example, if requirements are being reviewed, the systems engineers, design engineers, test engineers, O&M team, and other relevant parties should all carefully review the material and provide constructive feedback. A list containing the names of key reviewers should be made prior to the review. Unless these reviewers provide written comments in advance or are present at the project review meeting, the item under review should not be signed off as complete.

You should go into a major project review with the perspective that delaying the transition to the next step can be the best choice depending on project status. Too often, ITS projects are allowed to progress to the next step even if the previous step has not been completed. The common sentiment is that the project should maintain its schedule and catch up technically in the next process step, but this rarely works. For example, a project that progresses to high-level design before requirements are baselined is at risk of misunderstandings concerning what is to be designed, developed, tested, operated, and maintained.

Every major product (e.g., specifications and test results) created in the life cycle should be reviewed. In Chapter 4, we identified the various decision-point reviews that occur to assess readiness for progressing to the next step in the "V". Each of these reviews could be as simple as two or three team members walking through the product, or it could involve the whole team and, in many cases, the contracting agency.

Some of the formal technical reviews that may be planned for an ITS project include:

- Project Planning Review (e.g., SEMP, PP, Risk Management Plan).
- Concept of Operations review to ensure that the operation of the system has been defined to meet the needs of the stakeholders.
- Requirements review to make sure that requirements are appropriate and traceable to user needs and verification tests and that all parties have a common understanding of the meaning of each requirement.
- High-Level Design review to present the project architecture, and the system analysis performed to arrive at the high-level design, sometimes referred to as a Preliminary Design Review (PDR).
- Detailed Design review to ensure that the detailed design is ready for implementation, sometimes referred to as a Critical Design Review (CDR).

- Implementation reviews, including reviews of prototypes and hardware/software products as necessary.
- Test Readiness review to identify whether the components, subsystems, or system are ready for the verification steps.
- Operational Readiness review to obtain agreement from all parties that the system is ready for full operation and maintenance.

In addition to the formal decision-point reviews that are shown in the "V", many less formal reviews may be conducted; these include walkthroughs, interim progress reviews, and other ad hoc reviews that may be scheduled to address a particular risk or development challenge.

5.3 Risk Management

Risk management is the identification and control of risks during all phases of the project life cycle. Murphy's Law is alive and well during most projects, so it's essential that you anticipate the risks and put plans in place for addressing them. The goal of risk management is to identify potential problems before they occur, plan for their occurrence, and monitor the system development so that early action can be taken if the risk occurs.

Risk management is composed of the following general steps:

- Risk identification
- Risk analysis and prioritization
- Risk mitigation
- Risk monitoring

If you don't actively attack the risks, they will actively attack you.

Barry Boehm

5.3.1 Risk Identification

The objective of the risk identification step is to identify the key risks to project success *at the beginning of the project.* This will require that project managers, stakeholders, and possibly outside experts brainstorm about where the risks may lie. They should take a look at all potential risks, from initial development all the way out through operations and maintenance and eventual retirement of the system.

There are many areas of risk that might affect your project:

- Technical (e.g., Is the project using any technologies that have not been widely deployed or that the project team is unfamiliar with? Are the requirements well defined? Are the development or test facilities inadequate? Is all technical documentation receiving adequate review?)
- Institutional (e.g., Does the project require agreements related to agency data sharing that haven't yet been created? Are there regulations or agency hurdles that must be overcome for the project to succeed?)
- Schedule (e.g., Is the schedule too aggressive? Are there particular tasks for which small schedule slips will have a major impact on the final deliverable? Is the schedule dependent on timely, government-promised equipment, information, or review?)
- Funding (e.g., Is the funding for the project secure, or is only part of it in place? Are there pending agency budget cuts that might impact development or operations?)

- Personnel (e.g., What will happen if there is a loss of key agency or contractor personnel? What will be the impact if key personnel do not have adequate experience?)
- Environmental (e.g., Does the deployment schedule call for installations at a typically rainy time of year? Are there environmental restrictions that might impact system deployment?)
- Commercial (e.g., Is sufficient schedule time allocated to allow for dependencies on commercial off-the-shelf [COTS] product deliveries?)

Risks should be expressed as "If <situation>, then <consequence>" statements. For example: "*If* an agreement cannot be made with Agency ABC to obtain incident information, *then* the dispatching software will not be able to take incident information into account when calculating the best route."

During this first step, it is important to obtain a broad sample of potential risks. However, try to keep reality in check. You're not trying to capture highly improbable events but rather those that may well occur and are likely to impact the project the most from a schedule, cost, or technical standpoint.

It is also important to recognize the opportunities that go along with the project risks. There is a risk that you will fall behind an aggressive schedule, but the shorter schedule may provide some additional benefit or opportunity. It is common to brainstorm project opportunities along with project risks, since risks are almost always incurred when attempting to capitalize on some opportunity. As you identify the risks, consider the aspects of the program that, if performed a certain way, might lead to potential cost savings, schedule improvements, or improved quality in the technical approach. The following steps in the risk management process are equally relevant to opportunity capture.

5.3.2 Risk Analysis and Prioritization

Once risks have been identified, the next step in the process is to analyze and prioritize them. In order to do this, we need to determine the impact should the risk occur, and the probability of its occurrence.

How severe will the impact be if the risk occurs?

For each risk identified, you should make an assessment of what will be impacted if the risk occurs. Risks typically fall into one or more of the following areas:

- Technical (risks affecting the quality of the resulting system)
- Schedule (risks that cause schedule slippage)
- Cost (risks that cause cost to exceed budget)

Depending on the nature of the project and the nature of the risks, this assessment of severity can be qualitative (e.g., high, medium, or low) or quantitative (e.g., weeks of schedule slippage or amount of cost overrun).

When determining risk severity, start with the qualitative approach (i.e., high, medium, low), but create some quantifiable criteria for each level. For example, a high-severity risk might cause a schedule slippage of more than six months. Such criteria will give you a way to measure the impact you expect from any particular risk.

What is the probability that the risk will occur?

For each risk identified, assign a probability that the risk will occur (e.g., high, medium, or low). It's a good idea to define each of these probabilities quantitatively to make sure that everyone is analyzing the risks the same way (e.g., high = greater than 30% probability that the risk will occur).

Once each risk has been analyzed, the next step is to prioritize the risks according to the severity of the impact and the probability of its occurrence, as shown in Table 19. Any risk that falls into the upper left-hand corner group lettered A, B, and D is important. These risks have a higher probability of occurrence and a greater impact. Mitigating these likely risks will give you the best cost, schedule, and/or technical savings. Depending on available resources, try also to consider the risks that fall into cells E, C, and F, in that order.

Table 19: Risk Prioritization Matrix

		Probability of Occurrence		
		High	**Medium**	**Low**
Severity of Impact	**High**	A	B	C
	Medium	D	E	F
	Low	G	H	I

5.3.3 Risk Mitigation

The objective of risk mitigation is to identify and evaluate alternatives for handling the risks identified above. From a project management standpoint, there are several ways to address risks:

- Avoid the risk altogether (e.g., change in requirements or design)
- Take actions to reduce the likelihood or severity of the risk (e.g., prototype a new technology or develop and deliver software incrementally)
- Accept the risk and do nothing, which is often the approach for risks with low impact and low probability of occurrence

A risk mitigation strategy describing the steps to be taken to lessen the severity of the risk and the probability of its occurrence should be created for each risk. The strategy should identify the risk, an assigned owner, and the schedule and budget for implementation of the mitigation steps. The contingency schedule and budget should be allocated up front in order to have the time and funds to address each risk adequately. It's not enough just to create mitigation plans – you should begin to execute them according to the schedule and documented approach. The goal is to reach an acceptable risk mitigation solution between the customer and the contractor so that the project remains technically viable, on schedule, and on budget to the extent possible.

The risk mitigation strategy can take the form of a standalone risk management plan or it can be a section in the SEMP, depending on the size and complexity of the project.

As we discussed earlier, opportunities for potential cost savings, schedule improvements, or improved quality should be considered along with risks. During this step, you should put a plan in place to exploit each opportunity you've identified and implement the plan.

5.3.4 Risk Monitoring

Risks should be monitored throughout the life cycle to determine whether the mitigation steps are actually lessening the severity or probability of each risk. It's also possible that the nature of the risk has changed (e.g., the due date for delivery of the software code was extended because another project was delayed).

One way to monitor risks is to develop metrics that will give an indication that a risk is occurring. These metrics (e.g., cost versus schedule to identify if cost overruns are beginning to occur) should be easy to track and should provide some signal of a potential problem. Risk status should be reviewed periodically during progress meetings and other reviews.

5.4 Configuration Management

Configuration management (CM) can be defined as "*A management process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design and operational information throughout its life.*" (From ANSI/EIA 649-1998)

Establishing the system baseline, or configuration, and managing change to that baseline, are key processes for ensuring that system integrity is maintained throughout the life of the system. Consider it this way – if you had to recreate the system at a certain state in its life cycle or duplicate the deployed system in the test lab to check out a fault, would you have all of the configuration data and documentation version information you would need to do so? Following a strict configuration management process will ensure that you do.

The configuration management process consists of five major activities:

- Configuration management planning – Planning for what needs to be controlled to completely define the configuration of the system, how you change a controlled configuration, how you keep track of those changes, and how you verify that the CM processes are working
- Configuration identification – Identifying the functional and physical characteristics of a configuration item
- Configuration change management – Controlling change to those characteristics
- Configuration status accounting – Keeping track of the status of changes to the configuration items (e.g., proposed, approved, or implemented)
- Configuration auditing – Verifying that CM procedures are being followed as well as the consistency of documentation against the configuration item

This discussion of configuration management provides just a basic overview of what is a major discipline within systems engineering. There are resources that provide more detail and specific tools and techniques for the implementation of configuration management. These resources may provide additional information for adapting the information presented here to address specific project needs. Such resources include:

- TMC Pooled-Fund Study Configuration Management for Transportation Management Systems – Final Report September 2003; available at http://tmcpfs.ops.fhwa.dot.gov/cfprojects/new_detail.cfm?id=24&new=2
- A Guide to Configuration Management for Intelligent Transportation Systems – April 2002; available at http://www.itsdocs.fhwa.dot.gov/JPODOCS/REPTS_TE/13622.html

5.4.1 Configuration Management Planning

The processes and procedures to be used to manage the configuration of the system and changes to that system are documented in a Configuration Management (CM) Plan. The CM Plan may be a separate document (if the project is of sufficient size or complexity) or it may be part of another project planning document (e.g., PP or SEMP). The CM Plan is created at the beginning of the project life cycle. Once approved, the CM processes and procedures described in the plan are used throughout the remainder of the life cycle through retirement or replacement of the ITS system.

During the CM planning phase, you should select your CM tool. This tool will capture the configuration definition and will be used to establish and maintain the system baselines (e.g., hardware or software). It could be as modest as a spreadsheet if your project is small or an industry-standard CM tool if the project is more complex.

It is also during this early planning phase that the configuration control board (CCB) is established. The purpose of the CCB is to control the baseline of the system and all changes that are proposed and implemented on it. Its membership and process should be documented in the SEMP or the CM Plan. Membership includes representation from both engineering and program management and from both the contractor and the contracting agency. For small projects, this may be only two or three people. The CCB reviews every change, including its technical and programmatic justification, implementation schedule, potential effects on other parts of the system and project cost/schedule, and the risks involved with deploying the change.

5.4.2 Configuration Identification

Configuration identification is the selection of the software, hardware, and documentation that will be tracked. These configuration items collectively represent the system baseline and typically include:

- Hardware items
- Software modules
- Communications interfaces
- Project documentation (e.g., Concept of Operations, requirements, and design documents)
- Support tools

Each configuration item must have a unique identifier, clearly marked on it in some fashion, so that the item can be identified and tracked. For continuity, many projects use identifiers that are consistent with the nomenclature used to identify items in the project contract.

5.4.3 Configuration Change Management

Once the configuration items have been identified, any changes to them must be handled in a controlled fashion. All changes must be clearly described and presented to the CCB to assess the technical, cost, and schedule impacts. Only after the CCB has approved the change should it be implemented and the baseline changed. Once the change has been approved and implemented, it is formally documented, the baseline is updated, and the control number is updated. You should define the format of the control number – a revision number or version number – in the CM Plan.

Your Traceability Matrix can be a useful tool to help assess the impact of a proposed change since user needs are traced to requirements, which in turn are traced to system components and verification tests.

5.4.4 Configuration Status Accounting

At any time during the system life cycle, you should always know the configuration of every item. From the time that a change is proposed and all the way through its approval cycle and final implementation in the deployed system, the change should be tracked. When it has been superseded by another change, the initial change should be archived. A complete history of all changes to all configuration items should be maintained throughout the life of the system and eventually archived.

It's a good idea to follow the same process for any test equipment used to debug the system. For example, if there are multiple dynamic message signs that will be tested, it's important to make sure that all tests were run using test equipment configured the same way.

5.4.5 Configuration Auditing

How do you know that your project team members are following the documented CM processes to establish the baseline and control changes to it? You should periodically audit the processes and procedures that they're using against those in your CM Plan and also assess whether or not the CM processes are working effectively.

An auditing exercise should also verify that the state of each configuration item and its associated specifications, interface control documents, and other data are indeed as recorded in the CM tool. Check that a rabbit trail that tracks the history of changes to each item is complete.

6 APPLYING SYSTEMS ENGINEERING

6.1 The Traditional Project Life Cycle and Systems Engineering

The systems engineering approach discussed in previous chapters may be viewed as an extension to the traditional project development process that is already established in transportation agencies. As transportation organizations gain experience with ITS projects and the systems engineering approach, they typically find that they can weave the systems engineering processes and best practices into their overall project development process.

6.1.1 Traditional Transportation Project Development

Transportation projects are identified and funded through transportation planning and programming/budgeting phases. Funded projects are then implemented using a process similar to the traditional capital project development process shown in Figure 36, but the exact process used for ITS projects will vary with the type of project. For example, ITS projects that install only field equipment (e.g., variable message signs) would use a process that is very close to the traditional process shown in Figure 36. ITS projects that involve hardware and software development and integration would require additional systems engineering analyses that would be significant extensions to the traditional process.

Figure 36: The Traditional Transportation Project Development Process

While project development processes vary from state to state and from organization to organization in each state, the transportation project development process tends to have the same major steps.

- **Project initiation –** In this step, the project manager is identified, the project team is assembled, and the project development is planned. A high-level definition of the project is developed, costs are estimated, and the required forms and checklists are completed to garner approval for the project from the sponsoring and funding agency(ies). For FHWA and FTA, this is a critical point in the process where approval to proceed is given and federal funds are obligated.
- **Preliminary engineering –** In the traditional capital project development process, environmental, right-of-way, and other studies are performed depending on the type of project. These studies result in better understanding of the project requirements and constraints. ITS projects that include a construction component will require these same studies as well as additional engineering analyses to fully specify the project requirements for the ITS portion of the project. Note that from a federal aid perspective, preliminary engineering also includes plans, specifications, and estimates (PS&E). PS&E is split out separately here to differentiate between requirements-oriented and design-oriented steps in the traditional project development process.
- **Plans, specifications, and estimates (PS&E) –** The detailed design for the project, including detailed project specifications, estimates of material needs, and associated costs,

is documented. In a traditional construction project, this step provides companies with all the information they need to develop an accurate bid. Construction elements in an ITS project will also require traditional design documentation (e.g., layout sheets, plan and elevation views, and cross-section details). Design documentation is required for the hardware and software components in an ITS project, but it takes the form of high-level design, interface specification, and detailed hardware and software specifications.

- **Construction –** The project is built. In a traditional transportation project, this is construction of the actual physical improvement. In an ITS project, this includes the procurement and implementation of the hardware, software, and enabling products (e.g., manuals, operating procedures, and training). This step also includes both the inspection of the physical improvement and integration and the testing of the implemented system.
- **Project closeout –** After final inspection, the completed project is accepted, as-built plans are created, a project history file is completed, and final project documentation is submitted for audit prior to final payment.

6.1.2 Mapping Systems Engineering into the Project Life Cycle

As it has evolved through a century of building roads and public transit systems, the transportation project delivery process used by most agencies today already includes many important features of the systems engineering process. In both processes, the system is specified in increasing detail, beginning with needs, moving to requirements, and then into design. Multidisciplinary project teams and systematic stakeholder outreach and communications are hallmarks of a good transportation project development process and sound systems engineering practice. By taking advantage of this similarity in concepts and processes, the systems engineering process can be integrated as an extension to the agency's existing project development process. This alignment of the traditional transportation project development process with the systems engineering process is shown in Figure 37.

Making these types of linkages and mainstreaming ITS development into the agencies' project development process makes it easier to incorporate systems engineering into each agency's process.

Although there are similarities, there are also key differences between the traditional process and the systems engineering approach that should be considered when planning your next ITS project. For example, in the traditional transportation project development process, there is clear contractual separation between the consultant that prepares the PS&E and the contractor that builds the project. This is a risky approach for many ITS projects, in which it is important to have more continuity across the project development life cycle so that the contractor who ultimately implements the ITS system clearly understands the underlying user needs and requirements and has the latitude to implement the most cost-effective solution. For example, the contractor that implements custom software for an ITS project should also participate in the detailed software design. You would not want to impose a contractual barrier between the software designer and the software implementer. Differences like these have led to use of innovative procurement practices for ITS projects that are discussed in the next section.

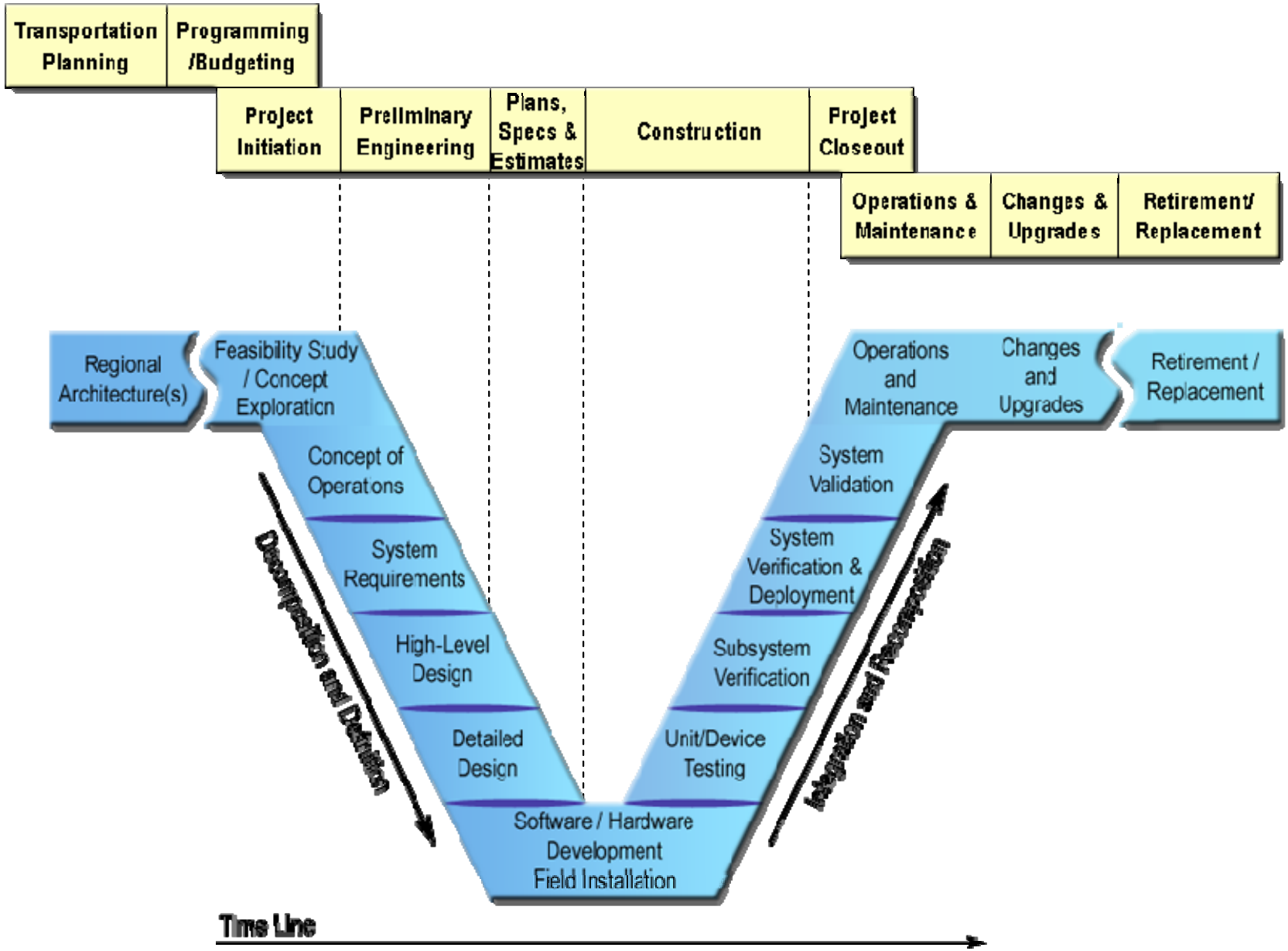


Figure 37: Systems Engineering as an Extension of the Traditional Project Life Cycle

6.2 Applying Systems Engineering in Your Project

6.2.1 Procurement and Systems Engineering

The project development process is strongly influenced by the selected procurement strategy. ITS projects have been procured through traditional low-id, systems manager, design-build, and other innovative procurement approaches. The procurement strategy will influence who (i.e. agency, consultant, or contractor) takes the lead for each process step, but the fundamental systems engineering process steps (Concept of Operations, Requirements, Design) should still be accomplished for all types of ITS system development projects. An important thing to remember is that the agency is never completely off the hook, regardless of the procurement approach. With any approach, there is always a need for the agency to be involved in the process.

The systems engineering analysis requirements identified in FHWA Rule 940.11/FTA Policy Section VI include a requirement for analysis of procurement options.

A poorly chosen procurement strategy can adversely impact a project just as much as the lack of a sound systems engineering approach. It is important to tailor the procurement strategy based on the type of ITS project and not to assume that it always has to be done the same way. The traditional approach of putting a specification on the street and awarding the implementation to the low-bid contractor may work well for projects with extremely well-understood requirements. However, the complications of inexperienced personnel, new requirements and technologies, changing environments, and shifting priorities lead to the need for newer approaches for many ITS system acquisitions. This is particularly true when a project includes custom software development.

Table 20 identifies some of the procurement approaches that have been used for ITS projects and highlights a few of the considerations for their use in your next project.

Table 20: Procurement Approaches for ITS Projects

Approach	Key Considerations	Comments
Commodity Supplier	Low-bid selection of prequalified packages; fixed-price contract	Applicable only for unmodified off-the-shelf software or hardware
Low-Bid Contractor with Consultant Design	Consultant performs 100% of design; may provide additional services during implementation Low-bid selection of contractor; fixed-price contract No collaboration on design between agency and contractor	Conventional way to procure construction projects Only applicable to extremely well-defined ITS projects Not applicable to ITS projects with significant software development
Systems Manager	Single contractor responsible for planning, design, implementation, and testing Negotiated (qualifications-based) award precludes contractor from providing construction/equipment Separate low-bid procurements	The selected systems manager has overall responsibility for delivering an operational system Additional contracting burden on agency Accommodates iterative

Approach	Key Considerations	Comments
	managed by the agency for construction and equipment	development and collaboration between agency and contractor Preferred for projects with significant software development
Design-Build Contractor with Design Consultant	Requirements/high-level design may be prepared by a consultant prior to contractor selection Best-value contractor selected based on price and qualifications Single contractor responsible for design, implementation, and testing Contractor can provide construction services/equipment	Some agencies do not allow design-build contracting Loss of continuity/flexibility if contractor does not participate in requirements/initial design Do not use a low-bid process to select a design-build contractor Preferred for major projects with significant construction
Consultant	Negotiated (qualifications-based) award limits consultant work to personal services Consultant services used for system requirements, design, and agency support during system implementation (define test plans, etc.)	Used to support initial consultant/system manager selection in previous approaches Can also be used to supplement in-house capabilities, support systems engineering process improvement, etc.
Outsourcing	Used to support acquisition of a capability/function rather than a specific system Best-value (qualifications-based) or low-bid (cost/rates-based) selection	Outsourced functions may include traveler information, toll collection; may involve a public-private partnership

A hybrid approach may be preferable for some ITS projects. For example, systems that include significant software development and significant construction may best be implemented with a hybrid approach that includes separate contracts for each of these activities.

Obviously, selecting the "best" procurement approach for your ITS project and your agency is not a trivial matter. Fortunately, there is help. *The Guide to Contracting ITS Projects* and its companion web-based tool, available at http://www.citeconsortium.org/Model/index.htm, have been developed through the National Cooperative Highway Research Program (NCHRP). The documentation provides insights into the procurement processes for ITS and advises on how to make the right choices based on the particular circumstances of a project and the procuring agency. The web-based tool walks you through an eight-step decision model, beginning with initial decisions about the type of project and finishing with recommendations on how to build the contractual terms and conditions.

The contracting model includes decisions about work allocation, method of award, and contract form and type. Each step in the model consists of a series of questions to elicit the

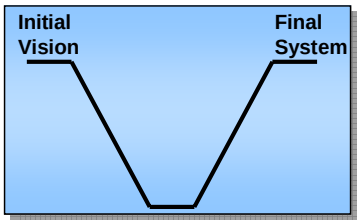
many possible dimensions, including project size and complexity, staff experience, and agency policies.

In a recent Talking Technology and Transportation (T3) webinar, *Successful ITS Procurement*, Mac Lister of the FHWA Resource Center and Phil Tarnoff of the University of Maryland concluded that "The right procurement approach may not guarantee success, but the wrong approach will guarantee failure." The key is to keep asking "What are we buying?" The answer to that question, along with any uncertainty you may have in answering it, should guide you in selecting the right approach for you and your project.

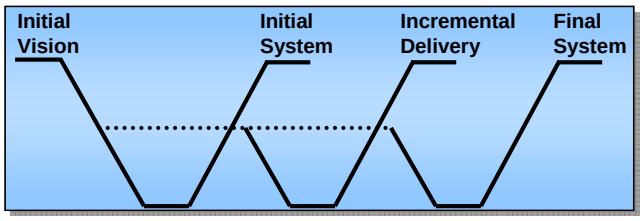
6.2.2 Selecting a Development Strategy

There are several different ways that a system can be developed and delivered using the "V" systems engineering model. The best development strategy depends on how much you know about the system that you want to implement, whether you have all the funds that you need to implement the system in one fell swoop, your agency and contractor capabilities, and your assessment of the project risks. Three basic development strategies can be used:

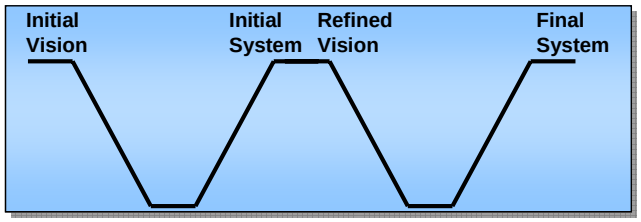
- **Once-through –** Plan, specify, and implement the complete system in one pass through the "V". This approach, also sometimes called the "waterfall" approach, works well if the vision is clear, the requirements are well understood and stable, and there is sufficient funding. The problem is that there isn't a lot of flexibility or opportunity for recovery if your vision or the requirements change substantially.



- **Incremental –** Plan and specify the system and then implement it in a series of well-defined increments that each deliver a portion of the system. In this case, you are making one pass through the first part of the "V" and then iterating through the latter part for each phased increment. This is a common strategy for field equipment deployment where system requirements and design can be incrementally implemented and deployed across a metropolitan area in several phases and several projects.



- **Evolutionary –** Plan, specify, and implement an initial system capability. Gain experience with the initial system and define the next iteration to fix problems and extend capabilities. Refine the Concept of Operations, add and change system requirements, and revise the design as necessary. Continue with successive iterative refinements until the system is complete. This strategy can be shown as a series of "Vs" that are placed end to end since system operation on the right side of the "V" influences the next iteration. This strategy provides the most flexibility but also requires project management expertise and vigilance to make sure the development stays on track. It also requires a measure of patience on the part of your stakeholders since they need to stay with you and continue to support the overall program even though their



favorite feature may not be ready as soon as they had hoped. For particularly complex projects, a *spiral model* may be used, which is an evolutionary approach that is driven by risk management and extensive planning in each iteration. In the spiral model, the initial iterations include prototyping, analyses, and studies that are intended to reduce risk prior to implementation of an operational capability. The products in each iteration are defined to reduce risk as the system's degree of definition and implementation is increased incrementally. In any evolutionary deployment strategy, each iteration should include stakeholder involvement and decision points to ensure consensus on direction and commitment to proceed as the system is incrementally implemented.

There are many variations and hybrid combinations of these three basic strategies. Table 21 summarizes the distinctions among them.

Table 21: Development Strategy Comparison[19]

Development Strategy	Define All Requirements First?	Multiple Development Cycles?	Deliver Interim Releases?
Once-Through	Yes	No	No
Incremental	Yes	Yes	Optional
Evolutionary	No	Yes	Yes

Table 22 identifies some of the reasons to pick one strategy over another. In general, the once-through strategy is well suited for low-risk projects and the evolutionary strategy is more adaptable and well suited for higher-risk projects where there are significant unknowns.

Table 22: Development Strategy Opportunities and Risks[20]

Development Strategy	Opportunities (Reasons to Use)	Risks (Reasons to Avoid)
Once-Through	▪ All capabilities needed/desired at first delivery ▪ Must phase out old system all at once ▪ Efficient – IF you know exactly what you want	▪ Requirements are not well understood ▪ Rapid changes to requirements possible ▪ Large system with many unknowns ▪ Limited staff or budget available now
Incremental	▪ Early capability is needed ▪ System breaks naturally into increments ▪ Funding/staffing will be incremental	▪ Requirements are not well understood ▪ All capabilities needed/desired at first delivery ▪ Must phase out old system all at once

[19]From "The Spiral Model as a Tool for Evolutionary Acquisition", Boehm, Barry and Hansen, Wilfred.

[20]Ibid.

Evolutionary	▪ User feedback is needed to understand full requirements ▪ Early capability is needed ▪ System breaks naturally into increments ▪ Funding/staffing will be incremental	▪ All capabilities needed/desired at first delivery ▪ Must phase out old system all at once

6.2.3 Tailoring Systems Engineering for Your Project

Chapters 4 and 5 described the systems engineering process steps in some detail. On some projects, a given step may be performed very informally (e.g., on the back of an envelope, or in an engineer's notebook); on other projects, the activity may be performed very formally, with interim products under formal configuration control. This document is not intended to advocate any level of formality as necessary or appropriate in all situations.

The systems engineering analysis requirements identified in FHWA Rule 940.11/FTA Policy Section VI allow the systems engineering analysis effort to be tailored so that it is on a scale commensurate with project scope.

INCOSE also stresses variation in the systems engineering process:

> Like all processes, the Systems Engineering process at any company should be documented, measurable, stable, of low variability, used the same way by all, adaptive, and tailorable! This may seem like a contradiction. And perhaps it is. But one size does not fit all.[21]

This message is particularly important for ITS projects because so many of our projects are smaller, less complex, less risky projects like signal system upgrades. Even for small projects, you still should have documented requirements, design, and verification procedures. Tailoring isn't an invitation to skip steps. Tailoring allows you to adjust the amount of formal documentation and reviews and to focus the process on those steps that are most critical to your project's success.

In order to decide on the process that is appropriate for your project, you should perform a risk assessment to understand the complexities involved and how many unknowns there are. Some ITS projects are much larger and more complex than others, which makes them a greater risk and thus candidates for more rigorous processes, as shown in Figure 38. Ultimately, you want to define a process that will address the project's risks, no more and no less, so a preliminary risk analysis is a good way to determine how much process is appropriate.

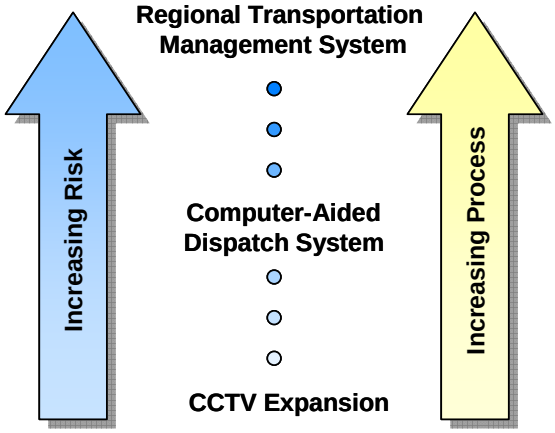


Figure 38: Tailoring Process Based on Risk

To get you started, Table 23 lists some of the project characteristics that affect project risk. The table shows, for example, that the number of stakeholders involved in a project is an indicator of project complexity and risk.

[21]From "A Consensus of the INCOSE Fellows" available at http://www.incose.org/practice/fellowsconsensus.aspx.

A project that involves only a single agency/department will tend to be lower risk and require less effort to develop a Concept of Operations and user requirements than a project that involves multiple agencies. There are a couple of disclaimers that should be made about the table. It includes rules of thumb that may not apply to your project. For example, user requirements can still be a challenge even for a single agency/department project if there is no consensus within the department. Also, your project will likely fall between the two "lower risk" and "higher risk" extremes in many areas.

As an example of how the systems engineering approach might be tailored for different types of projects, the *California Systems Engineering Guidebook for ITS*[22] uses three typical projects:

- **Project Example 1:** This project adds dynamic message signs (DMS) to an existing system. Fortunately, the systems engineering documentation for the current system exists and can be used as a basis for this incremental project's systems engineering effort.
- **Project Example 2:** This project is like Example 1, but it will add the capability to share control of the signs with another agency, which introduces risk. Both agencies need to come to terms on the Concept of Operations and requirements. The existing software must be upgraded to support the new interagency interface.
- **Project Example 3:** This project upgrades an existing signal system using off-the-shelf hardware and software. It illustrates the application of systems engineering to ITS projects that include no custom development.

Table 24 summarizes the tailoring information that is included in the *California Guidebook* for the three ITS projects. (Refer to the *California Guidebook* for a more comprehensive analysis of the three example projects.)

Of course, your project won't exactly match any of the three examples, and even if it did, you would want to take into account your own environment, process requirements, and staff experience when you tailor the process for your own project. The best approach is to think about the risks of your particular project and determine how best to mitigate them with a tailored systems engineering process. Think about the process ahead of time and write down what you are going to do so that everyone on your team and your stakeholders understand and agree on the right steps to follow and the level of detail that is needed. Whether you call it a project plan, a systems engineering management plan, or something else, it's critical to put your process and your plan in writing.

[22] *Systems Engineering Guidebook for ITS*, FHWA California Division and Caltrans.

Table 23: Tailoring the Process Based on Project Risk

Risk Factor	Lower Risk		Higher Risk	
	Project is/includes:	**Process Tailoring**	**Project is/includes:**	**Process Tailoring**
Project Type	Expansion/ enhancement	Review/use/modify existing SE documentation.	New system development	Create new SE documentation.
Project Size	Small (e.g., <$300K)	Use less formal project controls. You still need to plan the work, manage changes, and monitor and control risk, but the number of formal plans can be consolidated or significantly reduced to fit the budget.	Large (e.g., >$5M)	Use more formal project controls (planning, monitoring and control, risk management, configuration management) with documented formal plans for each aspect.
Project Complexity ▪ **No. of stakeholders**	One agency/department	Less formal reviews/walkthroughs, but buy-in is still important. Basic ConOps considers operations and maintenance, roles and responsibilities.	Multiple agencies	Emphasize reviews and walkthroughs. More focus on ConOps, including specific roles and responsibilities, operational scenarios, etc. More time specifying/ reconciling/prioritizing user requirements.
▪ **No. of interfaces**	Isolated system	Less formal high-level design – just make sure you understand the parts of your system and internal interfaces.	Regional integration	Focus on architectural design and interagency interface definitions and agreements.
▪ **Technology**	Proven technology	Adapt available, proven, performance-based design specifications.	State-of-the-art technology	Increase focus on feasibility studies, prototyping, trade studies/alternatives analysis. Perform studies prior to development contract.
▪ **Custom development**	Off-the-shelf procurement	Use higher-level performance-based design specifications.	Custom hardware/ software development	More focus on requirements analysis and modeling, architectural design, detailed design, and implementation process.
▪ **Criticality**	Downtime acceptable; neither safety nor security critical	Little/no specialty engineering required.	High availability required; safety and/or security critical	Add specialists to the team. Additional focus on process and quality assurance throughout development cycle.
Project Understanding	Good understanding of needs/requirements	Single iteration through the process to deliver complete system may be most efficient.	Needs/requirements unknown	Evolutionary, incremental approaches work best. Deploy a little, learn a little, and repeat.
Staff Experience	Staff experienced with similar projects	Use established processes. Use in-house systems engineering if appropriate.	Staff has no experience with similar projects	Use systems manager/consulting support. Additional oversight, staff development.

Table 24: Systems Engineering Application Examples[23]

Process Step	Example 1: Add DMS to Existing System		Example 2: Add Shared DMS Control		Example 3: Upgrade Existing Signal System	
	Effort	Notes	Effort	Notes	Effort	Notes
Regional Architecture	None	Existing architecture mapping applies	Low	Add new interface to existing mapping	Low	New mapping
Feasibility Study	None	Original feasibility study applies	Medium	Assess feasibility of adding new interface to existing system	Medium	Survey existing system and alternatives
Concept of Operations	None	Existing ConOps should apply	Medium	Focus on O&M responsibilities shared between agencies	Low	Define how system will operate and performance measures
System Requirements	None	Existing SRS applies – verify capacity	Medium	Define requirements for shared control, secure remote access	Low-medium	Short requirements document and verification plan
High-Level Design	None	OTS products; existing specs apply	Medium	Define architecture for shared control and define interfaces	Low	Identify legacy components, new components, and interfaces
Detailed Design	None	OTS products; existing specs apply	Medium	Specify SW design	Low-medium	Performance-based controller specs, central computer specs to support OTS procurement
Software/Hardware Development	None	Verify no update to SW necessary	Defined by SEMP/Plan	Custom SW; purchase COTS SW, servers, and comm HW	None	All components OTS
Unit/Device Testing	None	Verify factory tests performed	Defined by SEMP/Plan	Unit test custom SW; check out purchased HW and COTS SW	Defined by SEMP/Plan	Central computer tested; bench tests on controllers/firmware
Subsystem Verification	Low	Verify controller, signs, and comm working	Defined by SEMP/Plan	Host/integrate custom SW on HW	Defined by SEMP/Plan	Integrate/verify central computer to controller interface and verify functionality
System Verification	Medium	Reuse original procedures	Defined by SEMP/Plan	New procedures for new capabilities; all documentation ready	Defined by SEMP/Plan	Perform system-level acceptance tests; all documentation ready
Initial Deployment	Medium	Normal construction issues	Defined by SEMP/Plan	Staff trained; system installed and configured	Defined by SEMP/Plan	Deployment already occurred; support cutover/fallback
System Validation	None	System validation already performed	Defined by SEMP/Plan	Assess effectiveness; before-and-after study	Defined by SEMP/Plan	Validate performance, user, and maintainer satisfaction

[23]Summary of ITS project examples in the *Systems Engineering Guidebook for ITS*, FHWA California Division and Caltrans (URL).

6.3 Applying Systems Engineering in Your Organization

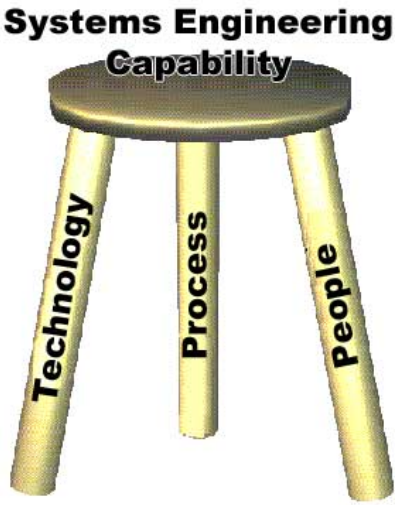


Figure 39: Quality Leverage Points

The three-legged stool shown in Figure 39 is a well-known metaphor for the three key leverage points that an organization has to improve its capability and efficiency in any area. The three legs that are shown are the major determinants of cost, schedule, and quality: the people in the organization, the technologies/tools that they use, and the processes and practices that are established. This metaphor is useful here as we consider the ways that an organization can improve its systems engineering capability.

The stool metaphor is a good one since there has to be balance across each of these leverage points in the same way that all three legs of the stool have to be the same length. For example, if you add technology and begin to use a sophisticated systems engineering tool but have not trained the people that will use the tool or established some best practices for its use, then it is likely that the organizational capability and efficiency will actually decrease as users struggle with the tool and spend less time performing productive tasks. The following three sections briefly discuss each of the three leverage points for improving the systems engineering capability in your organization; keep in mind that there must be a systematic plan for increasing the organization's systems engineering capability over time that balances all three of the leverage points.

6.3.1 People

> **There are no shortcuts around quality, and quality starts with people.**
>
> Steve Jobs

People represent a collection of skills, and a wide variety of skills are needed to successfully employ the systems engineering process in your organization. Every ITS project should have access to systems engineers who are familiar with the agency's processes and can tailor them to reflect the project size and level of complexity. The systems engineer will work with stakeholders to define the needs of the system, Concept of Operations, and system acceptance criteria. They will establish the requirements and specify and test the system. Whether this set of skills is in-house or contracted out, you should have a reasonable understanding of the systems engineering process and what can and should be expected during the project life cycle.

One good way to build competency into systems engineering is to identify one or more systems engineering specialists within your organization. These specialists would be a shared resource that could begin to make the application of systems engineering more consistent across projects. Depending on the size of your organization and the number of ITS projects, this could be a part-time job, a full-time job, or the responsibility of a small group. Many agencies have used their Information Technology group as a source for systems engineering and information technology skills.

In addition to identifying a few specialists, the organization should make a more general commitment to improve skills in project management and systems engineering for ITS

projects. Many organizations have encountered the pitfall of using a project manager who is skilled at traditional transportation projects but is ill prepared to manage a high-technology ITS project. Even if much of the systems engineering work is contracted out, all members of the project team should be trained so they can be effective participants in the systems engineering process. One of the recurring themes in this document is that broad stakeholder participation in the systems engineering process is essential.

Short courses in systems engineering are available through the National Highway Institute (NHI) and the Consortium for ITS Training and Education (CITE). (See Section 7.4 for more information.) A few larger agencies, including Caltrans and Florida DOT, have begun to implement systems engineering training that is more tailored to their specific processes. One benefit of implementing a common systems engineering process within your organization, as described in Section 6.3.2, is that it allows you to use a common training program that will be applicable to all agency ITS projects.

Note that the latest transportation legislation allows 100% use of federal funds for training and education. Specifically, SAFETEA-LU Section 5204(e), "Surface Transportation Workforce Development, Training, and Education", allows states to use 100% federal funding from the five core program areas for professional development activities. For example, this would allow you to use part of the funds from an ITS project to send agency project managers to systems engineering training or to host a commercially available systems engineering course at your facilities. For more detail on this opportunity, contact your FHWA Division Office Specialist.

6.3.2 Process

The quality of a product is largely determined by the quality of the process that is used to develop and maintain it.

Shewart, Juran, Deming, and Humphrey

One key to systems engineering is the use of a repeatable process or set of processes. As your organization employs systems engineering practices more and more, you should sit down with your colleagues at the conclusion of each project and determine what worked well and what should be done differently the next time. Start to document the processes you used during the project life cycle, and modify them based on your lessons learned. You'll want to pay extra attention to any project reviews you held to measure their quality. After you've developed a good set of processes that have been used on several projects, obtain agreement from your organization to establish processes for your organization. This will require buy-in from senior management, who should establish a policy that will support the processes and guidelines for their use.

Be sure to look at existing processes that are already defined in your organization. You may find that the Information Technology group has already implemented a good set of processes, many of which can be applied to ITS projects. The processes that are used for traditional capital development projects may also be helpful. For example, many agencies already have good, established project management processes such as risk management, establishment of integrated project teams, and value engineering processes, any of which may be equally applicable to ITS projects. When processes apply to many types of projects, it makes sense to define a single organizational process that can be applied to ITS projects and other types of projects.

There are many excellent resources that can help you to build systems engineering processes into your organization's standard business practices. The best-known resource is the

Software Engineering Institute's (SEI's) Capability Maturity Model Integrated (CMMI). CMMI is a good place to start since it reflects process improvement experience and lessons learned from broader industry and includes a wealth of processes and best practices for systems engineering, software engineering, and acquisition, all in a single unified framework. The staged representation of CMMI, shown in Figure 40, is instructive because it shows a proven approach to process improvement. You start with the right people trying to do the right things at Level 1, move to documenting the process on a project-by-project basis at Level 2, and then implement these tested processes for the organization at Level 3. This is a natural progression that results in good processes that support the organization's business needs and are supported by upper management and organizational policy. Each CMMI maturity level provides a foundation for the level above it.

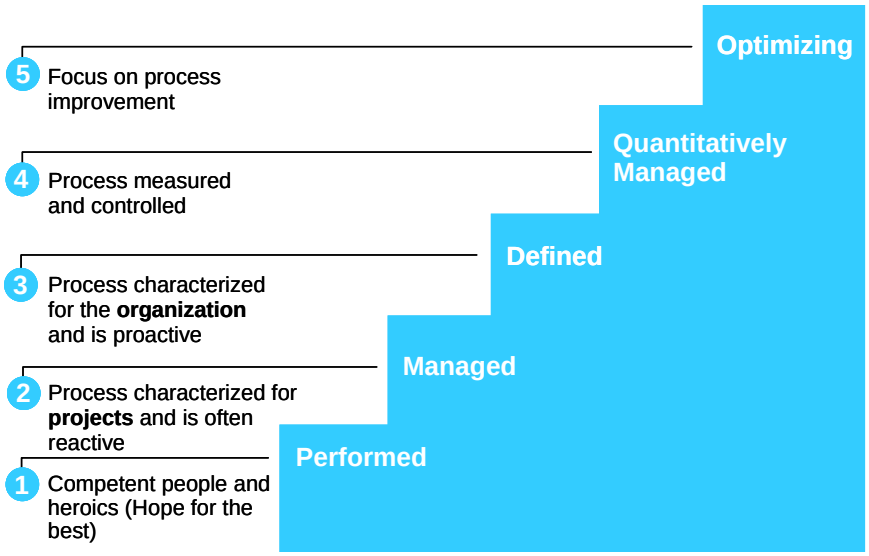


Figure 40: CMMI Maturity Levels

Once you have defined processes for the organization, the process that is used on a particular project will be based on the organizational process and tailored in a systems engineering management plan (SEMP) or similar document. Once all projects use tailored versions of the common process, the organization can begin to compare results, share lessons learned, and transfer people more easily between projects. Consistent historical data can be collected and used to improve estimates and consistency. A commitment on the part of the whole organization to use these processes, eliminate perceived shortcuts, and continuously improve the process will result in better-quality systems.

Documentation standards go hand in hand with organizational process standards. Document templates and formatting standards will save time and result in more complete and consistent documentation that can be shared more easily between projects. Documentation templates define the scope, content, and structure of each deliverable document. Like organizational process standards, standardized document templates can be tailored for each project. If your organization does not already have a library of such standards, they can be created based on industry standards and existing guidance. For example, the California Systems Engineering Guide and the Florida Systems Engineering Management Plan both include useful systems engineering documentation templates. (See Chapter 7 for potential resources.)

6.3.3 Technology

> **Technology applied to an efficient operation will magnify the efficiency.**
> **Technology applied to an inefficient operation will magnify the inefficiency.**
>
> Bill Gates

Like the systems engineering process itself, the use of systems engineering tools should be scaled to meet the needs of the project. The right suite of tools for a Transportation Management Center project would be overkill for a website upgrade. Systems engineering for small ITS projects is routinely done with standard desktop applications – a word processor for documents, a small database or spreadsheet for requirements management, and a drawing tool such as Microsoft Visio to generate the graphics and diagrams. For larger projects, more specialized systems and software engineering tools will allow you to do a better job of managing the larger set of requirements and more complex design artifacts that may be worked on in parallel by several engineers. Similarly, project planning, monitoring, and control can be handled with a tool like Microsoft Project for small-to-medium-sized projects. More sophisticated project tracking and control tools, document management, and configuration management tools will help you to better manage larger ITS projects.

Defining a standard suite of tools for your organization can facilitate sharing of information and people between projects. Of course, when the systems engineering effort is contracted out, the consultant/contractor may have their own preferred tool suite. Larger agencies have started to standardize on tools such as requirements management tools and to require that contractors use them so that the agency staff does not have to be fluent in the range of tools that contractors might otherwise select. If such a standard suite is defined for your agency, there should be access to training for the tools and guidance for applying them to an ITS project.

Although many good tools are readily available, you should weigh the introduction of a new tool and its resultant learning curve against its potential efficiencies. One agency recently experienced more than a year of delay as a sophisticated project management tool was procured, installed, and then found to require extensive customization to support the agency's business practices. Any major tool acquisition should itself be supported by a systems engineering analysis. Make sure you understand the needs, validate the requirements for the tool with prospective users, and then do a trade study of the alternatives. After acquisition, the new tool should be used on a pilot project or two before it is used more broadly within the agency.

7 RESOURCES

This guide borrows liberally from many excellent resources, including general systems engineering references and ITS-specific handbooks, guides, and white papers. The following resources were used in the construction of this introductory guide.

7.1 ITS-Specific Publications

FHWA Rule 940/FTA Policy, http://www.ops.fhwa.dot.gov/its_arch_imp/policy.htm

Guidance from Federal Transit Administration (FTA) on its policy, http://www.fta.dot.gov/16866_1044_ENG_HTML.htm

California Systems Engineering Guidebook for ITS, Federal Highway Administration, California Division and Caltrans

Systems Engineering Review Form (SERF), Caltrans Local Assistance Procedures Manual, http://www.dot.ca.gov/hq/LocalPrograms/lam/lapm.htm

Florida Systems Engineering Management Plan, Florida DOT, http://www.floridaits.com/SEMP/Index.htm

Systems Engineering and Architecture Compliance (Rule 940) Checklist for Use in Northern Virginia, http://www.dot.ca.gov/hq/LocalPrograms/lam/forms/msword/p07forms.doc

The Guide to Contracting ITS Projects, National Cooperative Highway Research Program (NCHRP), http://www.citeconsortium.org/Model/index.htm

Developing and Using a Concept of Operations in Transportation Management Systems, TMC Pooled-Fund Study, http://tmcpfs.ops.fhwa.dot.gov/cfprojects/new_detail.cfm?id=38&new=0

ITS Software: Effective Acquisition Practices, National Cooperative Highway Research Program (NCHRP). AASHTO, 2000

The Road to Successful ITS Software Acquisition, Federal Highway Administration

- Executive Summary: http://www.itsdocs.fhwa.dot.gov/jpodocs/repts_te/36s01!.pdf, EDL #4132
- Volume I: Overview and Themes (FHWA-JPO-98-035): http://www.itsdocs.fhwa.dot.gov/jpodocs/repts_te/36q01!.pdf, EDL #4130
- Volume II: Software Acquisition Process Reference Guide (FHWA-RD-98-036): http://www.itsdocs.fhwa.dot.gov/jpodocs/repts_te/36r01!.pdf, EDL #4131

7.2 General Systems Engineering References

INCOSE Systems Engineering Handbook, International Council on Systems Engineering, http://www.incose.org

NASA Systems Engineering Handbook, National Aeronautics and Space Administration, http://ldcm.nasa.gov/library/Systems_Engineering_Handbook.pdf

Department of Defense (DOD) Systems Engineering Plan (SEP) Preparation Guide: http://www.acq.osd.mil/ds/se/publications/pig/sep_prepguide_v1_2.pdf

Buede, D.M., *The Engineering Design of Systems: Models and Methods*, Wiley Inter-Science, 2000

Hooks, Ivy, *Writing Good Requirements*. Paper written for the Third INCOSE Symposium and published in the Proceedings of the Third International Symposium of INCOSE – Volume 2, 1993, http://www.complianceautomation.com/papers/writingreqs.htm

McConnell, Steve, *Software Project Survival Guide*, Microsoft Press, 1998

Young, R.R., *Effective Requirements Practices*, Addison-Wesley, March 2001, http://www.ralphyoung.net/

7.3 Selected Systems Engineering Standards

AIAA G-043-1992, *Guide for the Preparation of Operational Concepts Document*

ANSI/EIA-632, *Processes for Engineering a System*

CMMI SWE/SE/IPPD/SS, *Capability Maturity Model–Integration for Software Engineering, Systems Engineering, Integrated Product and Process Development and Supplier Sourcing*

EIA-649, *National Consensus Standard for Configuration Management*

IEEE 830, *Recommended Practice for Software Requirements Specifications*

IEEE 1012, *IEEE Standard for Software Verification and Validation Plans*

IEEE 1220, *Application and Management of the Systems Engineering Process*

IEEE 1233 – *Guide for Developing System Requirements Specifications*

IEEE 1362, *Guide for Information Technology – System Definition – Concept of Operations Document*

ISO/IEC 15288, *Systems Engineering – System Life Cycle Processes*

7.4 Systems Engineering Training

The SAFETEA-LU reauthorization allows states to use 100% federal funding for professional development, covering the systems engineering training courses listed here and any other public or privately offered courses in your region. Check with your FHWA Division Office for more information. A broad range of courses for ITS practitioners, including courses related to systems engineering and project management, are available through the National Highway Institute (NHI) and the Consortium for ITS Training and Education (CITE). Current course offerings are listed on these websites:

- ITS Professional Capability Building Program: http://www.pcb.its.dot.gov/
- Consortium for ITS Training and Education (CITE): http://www.citeconsortium.org

Technical Report Documentation Page

1. Report No. FHWA-HOP-07-069	2. Government Association No.	3 Recipient's Catalog No.
4. Title and Subtitle: System Engineering for Intelligent Transportation Systems		5. Report Date: January 2007
		6. Performing Organization Code
7. Author(s): National ITS Architecture Team		8. Performing Organization Report No.
9. Performing Organization Name and Address Iteris, Inc. 1515 S. Manchester Ave. Anaheim, CA 92802		10. Work Unit No. (TRAIS) 11. Contract or Grant No. DTFH61-01-C-00023
12. Sponsoring Agency Name and Address Department of Transportation, Office of Operations HOTM Room 3404 400 Seventh Street SW Washington, DC 20590		13. Type of Report and Period Covered 14. Sponsoring Agency Code
15. Supplementary Notes: Lee Simmons - Contracting Officer Technical Representative		

16. Abstract

The System Engineering for Intelligent Transportation System guide is intended to introduce you to systems engineering and provide a basic understanding of how it can be applied to planning, designing, and implementing intelligent transportation systems (ITS) projects. The guide leads you step by step through the project life cycle and describes the systems engineering approach at each step. It describes how to begin implementing the systems engineering approach on your next ITS project and incorporate it more broadly into your organization's business processes and practices.

17. Key Word Intelligent Transportation Systems, ITS, Architecture, Planning		18. Distribution Statement: No Restrictions. This document is available to the public.	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 115 (including front cover page not the back cover	22. Price $0.00

Department of Transportation, Office of Operations
HOTM Room 3404
400 Seventh Street SW
Washington, DC 20590
Toll-Free “Help Line” 866-367-7487
www.ops.FHWA.dot.gov
Publication No. FHWA-HOP-07-069
EDL-14340
January 2007