Systems Engineering
Jr. Handbook

The International Council on Systems Engineering
LA Chapter

http://www.incose-la.org

&

The Creativita Institute

www.creativitainstitute.org

TABLE OF CONTENTS

A Note to Teachers and Students

Welcome to Systems Engineering! In this student handbook, we would like to explore the subject of systems engineering with you to build a powerful set of concepts with processing tools for your future system design, development, and management projects.

Systems engineering is a way of thinking as well as a way of doing. Its roots began with the development of large-scale complex technical systems in the mid-20th century such as the Manhattan project. Since then many "super" projects for the engineering of large-scale and complex technical systems continue to evolve. Many of these projects, e.g., space station and Mars exploration mission, continue to inspire us.

According to a study from the Government Accountability Office (GAO) titled "Assessing the Relationship between Education and the Workforce" published on Jun 9, 2014, 'Both the number of science, technology, engineering, and mathematics (STEM) degrees awarded and the number of jobs in STEM fields increased in recent years. Many government programs, such as the Networking and Information Technology Research and Development (NITRD) were established to help adopt the education and practice of systems engineering (SE). They all recognized the critical need for the skills of engineering and systems.

To respond to the call, we are introducing SE in this student handbook to convey how the concept can apply to a variety of STEM projects. In the second part of the handbook, we will zero in on SE applications on Technology Competitions such as Robotics Contests.

Contributing Authors:

Hosame Abuamara, Pete Badzey, William Chang, Frederick Lawler, Jose Mancera, Phyllis Marbach, David Mason, Huey Nguyenhuu, Maryam Salimi, John Silvas, Shirley Tseng

System Engineering Principle

Most people, when faced with a problem to solve or a project to complete, do not jump in with a hammer and nails or a paintbrush before they know what they are supposed to solve or produce. This is part of human nature. Leaping into a solution without first finding out the goal of the activity inevitably wastes time, energy and resources. A disciplined approach to problem solving avoids a lot of this wasted effort if followed well.

Systems engineering is such an approach. In today's complex world, for instance, creating a complete smartphone business must include the smartphone designer, phone manufacture, phone carrier (e.g., AT&T, Verizon), content providers (App Store), and so on. Companies that specialize in App development must have an IT department to maintain their development environment, programmers who develop the Apps, security expert to help design and define their App to comply with cyber protection requirements, and marketing expert to help sale their Apps. To assure all parts are integrated seamlessly, the companies that deal with the users/customers are responsible for holding a rational and measured approach to guarantee the users/customers are receiving quality services from all the collected elements of network, software, and hardware that go into making the service work.

The discipline of systems engineering has many definitions and forms, but each has common essentials, namely:

1. Project planning
2. Analysis of project goals, objectives and requirements
3. Assessment of alternative paths to achieve #2, above
4. Repeatable design and implementation approaches for tasks in the project
5. Verifying that the finished product (or service) meets goals, objectives and requirements

By using a systems engineering approach, products and services can be developed efficiently and effectively, solving problems for human society while conserving resources and minimizing time for completion.

What is a System

The notion of a system is a 'whole' which consists of many interrelated parts (subsystems or system elements). Additionally, the integrated system's behavior and effect are greater than the sum of its individual parts. There is no restriction on whether the related parts are people or things, hardware or software.
Systems are everywhere in our daily discourse, from the cell phone to cable systems, from home monitoring to national cyber security systems. A prime example is a space shuttle (Figure 1). Each component on the shuttle is a full functioning system. The integration of these systems into a larger system (also called "system of systems") takes place through well-defined interfaces between systems. All these systems are managed and controlled by several computer systems that perform a common objective seamlessly to assure safe and efficient flight into space.

There are many definitions of a system, but one that will serve for the purposes of this handbook is as follows:

> *"A system is a collection of elements that, when working together, produce an effect that the individual elements operating on their own cannot produce." [1]*

Let's see the definition of a system from the INCOSE handbook [2]:

- A purposeful collection of inter-related components working together towards some common objective.
- A system may include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements and be operated by people.
- System components are dependent on other system components
- The properties and behavior of system components are inextricably intermixed

With the above definition, many products and services of modern life can be viewed as systems, from a microwave oven to a video game console to a banking computer network to a race car to a cruise liner to a satellite.

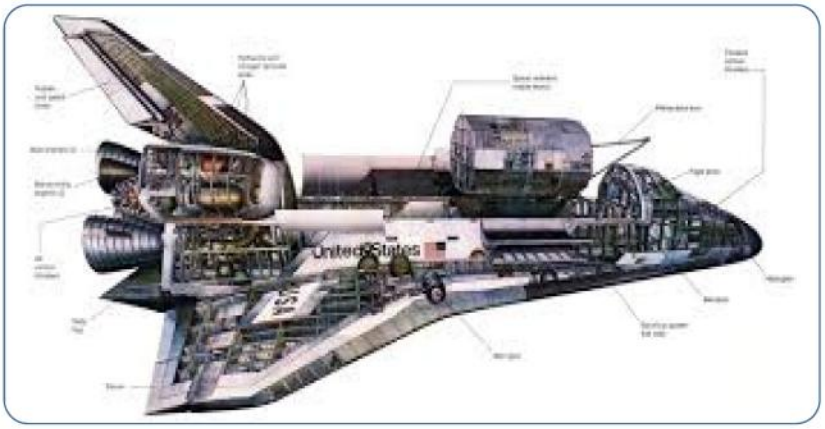


Figure 1. Space Shuttle System

Types of Systems

Engineers are responsible for defining, designing, and building

systems. There are three types of systems identified and defined in the INCOSE handbook:

- Products and Product Systems
    - Examples of these systems include any household appliances, such as TV, radio systems.
- Services and Service Systems
    - Any applications running on internet today are offered as services, for instance, online games. A service system can be a gaming host that support players online.
- Enterprises and Enterprise Systems
    - An example of enterprise systems is the software and hardware systems that are used in school as accounting tools to manage school budget/spending and teachers' salaries. Most of complex government projects are in this category; examples are Space Station, GPS systems, etc.

Complexity

Because components for a system may come from different vendors or manufactures, there is no guarantee these components will provide the same level of product details to the team that will put them together. Therefore, a good engineering practice should focus on the component's features and their interactions (interfaces) with other components rather than knowing how these components were built. An example is the development of the internet today. As the internet network continues to evolve, the shape of the concept and the reality of this "world wide web" continue to integrate with other networks: smart phone networks (a.k.a. cell phone networks), landline networks, stationary wireless computer connections, satellite links, and more. Other than following the same communications language (a.k.a. protocols), these networks can be built based on different technologies by different independent development companies. A fundamental system principle of the internet is an "open system" that allows individual networks' continuous adaptation and improvement via the "plug-and-play" capability.

There are many factors that can increase system complexity. Sample indications of system complexity and how they may influence systems engineering practice are listed below:

- Project Constraints (e.g., cost, schedule, vendor products, customers, etc.)
- Technical Performance Metrics (TPMs) (e.g., size, weight, space, speed, duration, flexibility, data throughput, operational latency, system reliability, etc.)
- Scope of "System" (i.e., what comprises the "System" being developed?) (e.g., hardware, software, module interface(s), etc.)

System Behavior

The context of system behavior in this handbook is to address the consequence of the interactions and relationships between system components (or subsystems) rather than the behavior of individual elements. In addition, a good systems engineer must be able to deal with the systems when they exhibit different kinds of expected or unexpected behavior.

System Science

Scientists tend to tackle their challenges with a structured approach, proceeding one step at a time, to bite off what they can chew, and avoid confusing sidetracks. For instance, instead of trying to tackle the whole universe at once, scientists take things apart and examine bits and pieces, acknowledging their own limitations. The application of this method in the SE practice is called functional decomposition and integration in systems design.

System Thinking

System Thinking focuses on the behavior of the whole rather than the individual parts. It is the ability to think of the interactions between elements of a system and their effect on the function or mission to be accomplished as opposed to a narrow focus on the elements. It also emphasizes the interfaces between/among the subsystems. Figure 2, below, provides a high level relationship between the System Thinking with other applicable philosophical components of Systems Engineering practices:

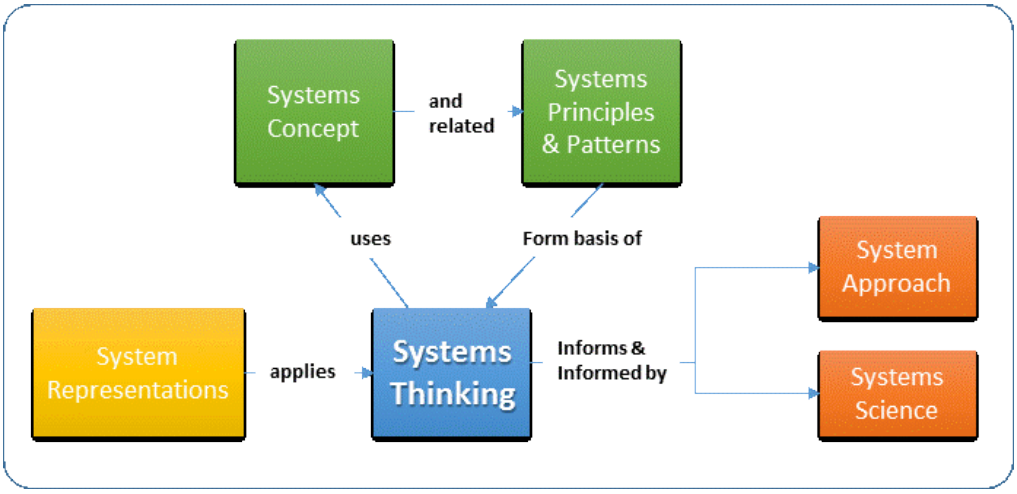


Figure 2. System Thinking [3]

System Modelling and Tools

The Systems approach shown in Figure 2 is a way of tackling real world problems using the tools of system science to enable systems to be engineered and used.

System modeling (SM) is needed during the design phase, and prior to building the actual system, to analyze the essential characteristics of a system and provide alternative solutions. SM standards include: SM Languages for Systems, Data Exchange Standards, Mathematical Modeling, Simulation Models, Model Transformations, and General Modeling Standards. Some example industry standards and tools are illustrated below:

- **System Modelling Concept and Standards**: A model is an abstraction of a system that offers insight about one or more of the system's aspects, such as its function, behavior, structure, properties, or cost. A system model represents aspects of a system and its environment. There are many different types of models. SM Standards include: Modeling Languages for Systems, Data Exchange Standards, Model Transformations, and General Modeling Standards. Using the SM enables for "what-if" studies to be performed, look at design alternatives, and analyze possible impacts before the system is built.
- **System Engineering Tools:** MBSE, DOORS, MATLAB, EXCEL, etc.

What is Systems Engineering?

Systems Engineering (SE) is a management methodology that controls the processes consisting of the definition, development, and deployment of a system to meet the customer objectives with the best outcome. Different from the traditional understanding of "engineering" that focuses more on a single and common area (e.g., discipline in electronic engineering), SE is responsible for the big picture. In particular, SE is an interdisciplinary (a team consisting of members having knowledge in different areas of engineering) approach and means to enable the realization of successful systems from concept to retirement.

SE is also the art of orchestrating a team with coordinated actions intended to complete an integrated system. The objectives of adopting SE can be illustrated as follow:

- **To determine fit**: SE emphasizes the system as a whole while it understands all the pieces of the system that best meet the requirements. This is accomplished by deploying a systematic and structured approach that applies quantitative and measurable terms, and values throughout the process.
- **To achieve balance**: SE practice aims for reasonable expectations by balancing cost/risk/schedule against function/performance.

- **To select the best compromise**: Recognizing that sufficient system design is not necessarily technically superior, SE practice is a way to help a project manager discover an effective means to optimize design complexity versus development simplicity.

The SE approach encompasses both the holistic (comprehensive) and modular views. It helps the team members appreciate key linkages across the whole system, allows the engineers to analyze it into parts with proper interfaces and synthesize knowledge about the parts to understand the whole. For instance, it's a bad idea to start designing a bathroom if one doesn't know what kind of house it's going into, or even to design the house without knowing a) how big it's supposed to be, b) how many people will live in it, c) what kind of environment it will be in, or d) the services it will require. These four (a-d) are examples of what we know as performance and functional requirements and interface specifications.

The SE practice offers useful guidance and suggests practical tools so the team can better understand system functions and details at different levels of the system to help reach an optimal solution. For instance during the design phase, the process includes guidance to decompose a system into sub-subsystems, and so on, to the lowest level. After the requirements are properly decomposed, each functional "block" (compartment) contains more specific and controllable insights which enable the team members to focus on the right level of detail without having to be overwhelmed by the complexity of the system. Conversely, SE also focuses on ensuring the subsystems work together as a whole to achieve the overall objectives of the system by modeling relevant characteristics of certain pieces while hiding some information from other pieces depending on what system level function is being performed. The SE process treats all parts (elements) as black boxes except for their external interfaces; therefore the system view for each engineer can be simplified. It is important to note that hiding information is not discarding it. A black box can be opened to explore details as necessary throughout the development life cycle of the system.

SE orchestrates an integrated team through a highly structured and interactive engineering process and helps to make the design and development of a complex system much more tractable and manageable. It allows for a part of the system to be studied or designed with minimal impact from other parts. As a result, subsystems can be designed concurrently to achieve design considerations while additional system analysis is being performed on other parts of the system.

From an implementation perspective, successful employment of the SE practice requires the leadership and communications skills across the SE teams that includes external stakeholders. Being organized and flexible at the same time are two key attributes of a strong SE leader. An effective SE team leader must provide clear guidance to the team on how communications will occur such as meetings attendance and reporting requirements to briefings that will be given

to the customer on a recurring basis and what each of the team member's role will be in preparing and delivering the subject briefings. For example, leading a team requires structure from ensuring there is a well-defined team organization, Work Breakdown Structure (WBS), integrated project schedule, regular occurring status meetings, as well as sufficient opportunities to meet with the customer for feedback. Although not everyone in the team can act as a "lead SE", leadership and communications skills can be clearly demonstrated by the SE's ability to collaborate with the teammates, recognize abrupt situations, and help the team adapt these dynamics accordingly. The effectiveness of these two skills can greatly minimize risks in dealing with system complexity and unexpected conditions throughout the lifecycle of the project, as will be discussed in the following sections.

Systems Life Cycle

The system lifecycle includes all phases of a system's creation and existence that can be broken down to different stages. The lifecycle permits division of a project into system conception, design and development, production and/or construction, distribution, operation, maintenance and support, and retirement. Throughout this book, the term **stage** refers to the different period of a system during its life cycle. Some stages may overlap in time, e.g., the utilization stage and the support stage.

System Life Cycle

To effectively manage a project, Program Management may employ the concepts of stage, milestones, and decision gates which are used to assess the evolution of a system through its various stages. Within each stage, it is essential that all activities performed achieve goals and serve to control and manage the sequence of stages, as well as the transitions between each stage must be clearly defined to avoid or minimize confusion.

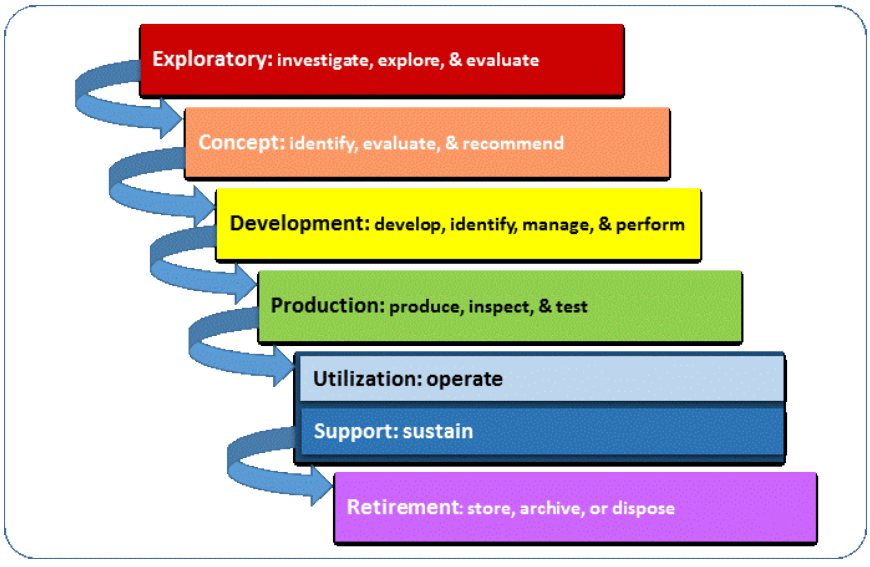


Figure 3. System Life Cycle [3]

A typical program is composed of the following stages:

- **Exploratory Research**: This stage generally includes the activities of mission definition (why the system is needed? who will be using it?) and the understanding of requirements (what is the system to look like? how will the system be used?). Determining objectives and applications of the project is critical to the development of successful systems.

- **Concept**: The concept stage helps the team discover alternate development options and determine the best approach to meet the requirements. Through the creation of operational models and prototypes, pros and cons of different functional or feature alternatives can be compared and clarified. This may lead to an incremental or evolutionary approach to system development.
- **Development**: The selected concept(s) are elaborated in detail at the lowest level with implementation guidance to produce the designed system or functionalities that meet the requirements. Because most complex systems take a long time to develop, program reviews with customer representatives normally take place frequently enough to assure the development effort is on track.
- **Production**: The production stage is about the creation of the system, whether it is fabrication for hardware-centric system or coding for software-centric system. It continues the activities of system realization and development through verification and validation. In a complex system production, modifications to the system may be required to address technical problems, reduce costs, or improve capabilities -- sometimes result in system requirement changes.
- **Utilization**: The utilization stage is when the system is delivered to the customer and deployed for use. If the delivered system (for instance, cell phone) is part of a big service system (a.k.a. system of systems), the integration and verification of this system's interoperability with other supporting systems (e.g., cell towers, call switching centers) are critical to the success of its deployment.
- **Support**: This stage includes the activities of system maintenance, logistics, and product and service life management, which may include activities such as service life extension or capability updates, upgrades, and modernization.
- **Retirement**: In the retirement phase, the system and its related services are removed from operation, and this includes the activities of disposal and retirement. However, if the system continues to have an application need, for instance a popular model of a family car, the retirement stage is replaced with a new life cycle for system extension or improvement.

Life Cycle Models and Processes

All the processes and models used within the system engineering Life Cycle are referred to as System Engineering Life Cycle Models. A Life Cycle Model is often visualized with a graphical 'Vee' Model representation of the processes and stages of a program development (See Figure 4). Although there are overlapping stages and the processes performed in each stage within the System Life Cycle illustrated in the previous section, the sequence of processes depicted in

the graphical 'Vee' model below in Figure 4 are designed to provide a structured framework for completing the design and verification of a system.

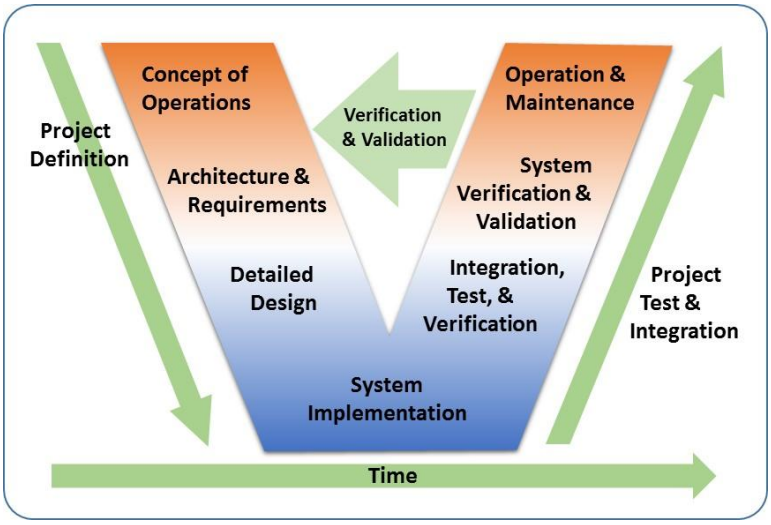


Figure 4. Graphical 'Vee' Model [2]

As shown these phases are worked from upper left and downward to the center, then upward on the upper right side. The initial phases of the life cycle capture the system-wide engineering Concepts of Operation. The Concept of Operations need not be fully completed first before the High Level Requirements are started (e.g., what the system is supposed to do and what performance is expected from the system.) As the High Level Requirements are maturing, the processes of the 'Vee' model proceed toward to the center of the 'Vee', and more details are created in the program until the Implementation of the design begins. With the functional decomposition process, the engineers are capable of specifying the whole system, its parts, and their interrelations clearly -- such top-down detailed design can enable thorough physical assembly. The subsystems or modules at intermediate levels are crucial for managing complex systems, because these activities allow the engineers to introduce complex details one step at a time.

When the system engineers complete the analysis of all subsystems and their interrelated components, they reach the bottom of the Vee. Turning the corner of the Vee, all required components/parts are manufactured and assembled to specifications in the phase of System Implementation. As the components begin to be tested and brought together into larger subsystems to prove that the product meets specifications, these stages are shown moving up the right side of the 'Vee' model, and completed at the Operation and Maintenance. If there is a need for the program or project to continue the next version development, the process will start the Vee again by the Verification and Validation of the new program objectives.

Systems Engineering Management

Different from the general perception of project management, Systems Engineering Management focuses upon the technical or engineering aspects of a project. It is about managing the resources and allocated assets to support and perform systems engineering, often including exploratory research and development (R&D) activities.

Project Processes

The development or evolution of a system is managed through the system life cycle discussed in the system life cycle discussed previously. For a complex system or a system consists of many elements such as software and hardware components, the contributions of systems engineering management of the project is critical to its success. The Venn diagram shown in Figure 5 portrays the interaction and dependency between Systems Engineering and Project Planning and Control.

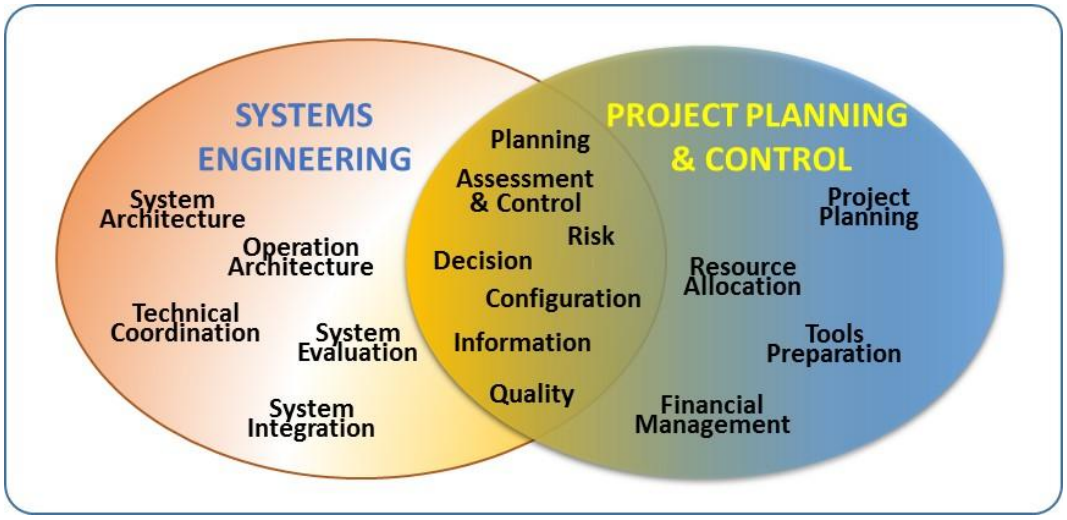


Figure 5. Interaction between Systems Engineering and Project Planning and Control [2]

These project processes are illustrated as follow:

- **Planning**: Following a disciplined process and adopting a variety of available tools to help set up the project plan, the project team has better chance to achieve the desired results within cost and schedule constraints. Some useful steps in the process can include:
    - Identify objectives and constraints
    - Identify key events in the project
    - Identify work packages
    - Schedule development

    - Cost and budget
    - Systems Engineering Management Plan
    - Risk Management
    - Configuration Management
- **Assessment and Control**: Through the collection of project data and the comparison of the results achieved against the plan, the team can gain more insight into the maturity of the project. A good practice must include:
    - Periodic assessments on all checkpoints and milestones of the schedule
    - Maintaining good communications within the project team and with the potential users
- **Risk Management:** This process attempts to anticipate and avert risks that may impact the schedule, functionality, cost, or other project objectives. It can occur anytime when there is a need to take appropriate actions to deal with uncertainty present throughout the system life-cycle.
- **Measurement**: Measurement provides the team an ability to quantify the project by collecting information relating to the system, and processes. The results can be used to objectively evaluate the system.
- **Decision Management**: Decision management is used to quantify the relationships among requirements, design choices, and risks.
- **Configuration Management**: This includes the establishment, maintenance, and control of requirements, documentation, and drawings produced. To help manage the impact of any change throughout the system's life cycle, Configuration Management ensures the change is necessary and any hidden risks that can adversely affect cost, schedule, and technical performance can be mitigated.
- **Information Management**: This process is to ensure information is properly organized, synchronized, stored, maintained, secured, and accessible only to those who need it. It is an important capability to facilitate the integrity and security of relevant system life cycle artifacts.
- **Quality Management**: Quality Management helps the team assure the processes were followed and the quality requirements are met. It involves outlining the policies and procedures for improving and controlling the various processes to achieve project efficiency and effectiveness in terms of time, cost, and quality.

Systems Engineering Practice on Robotic Competition

Robot competitions provide student teams a great opportunity to engage with engineering design challenges, including components of mechanical assembly, computer configuration, control software development, and system integration. Teams work together to design and build robotic systems that can perform predefined functions with or without human or control.

Ideally, all teams can just focus only on predetermined and controllable action requirements on the development of a system predefined for the competition. However, in an open competition, ultimate victory may be influenced by other factors such as the understanding/interpretation of the competing rules, design principles, development approaches, and execution strategy. Some of these factors may be subjective, although represented in the system's behaviors, functions, input, and output. As mentioned previously, most of these aspects can be classified into some representations of Systems Engineering practices, and, this chapter lists key process considerations to help facilitate the team's better appreciation of SE concept and applicability.

Planning

A team with a clear and objective-driven strategy is likely to perform better than others. The planning purview must be strategic, taking into account not only a system's life cycle from design to competition arena, but also the team members who are involved in various stages of the cycle. Although it is not necessary to overly complicate the design of a robot system, the team must be ready to manage some level of social impact and perception along with some requirements engineering, life-cycle analysis and multi-disciplinary teamwork for development and management.

Team Establishment

- What is the Team Name?
- What is the Team Logo?
- Who are the Team Members?
- What Roles are required for the competition (Leader, Note-taker, Testers, Assemblers, etc.)?
- When and where should the team meet? How often?
- What are roles of the mentors and teachers?
- Who else needs to be involved? (other stakeholders, such as external art designers)
- What communications mechanism (tool) can assist the team in working together more smoothly?

Project Planning

- Does the team understand the completion mission and objectives clearly?
- What is the goal (objective) of this team? What problems does the team intend to solve?
- What will the team build to differentiate the outcome from others? What impact will success have? How will it be measured? (What is success?)
- What is the system scope and system boundary? What are the risks and the payoffs?
- What are the project constraints and assumptions is the team operating under?
- How much will it cost? What is the budget? How can the team stick to the budget? (Program Management)
- How long will it take to build the needed team skills and develop, integrate, and test the system? How will progress be measured? What are the major check points (milestones) to assess the progress? Do they align with the competition schedule?
- What research does the team need to design and build the system?
- What are the applicable books, articles, and/or on-line resources?
- What tools, lab, materials, etc. are required for the entire competition (divided by phase)?

Note: Depending on the complexity of the target system, the team can be divided into several sub-teams (called Integrated Product Team or IPT). Each IPT can potentially consist of more than one student (for instance, multiple programmers) and be led by an IPT lead. Overall systems requirements and architecture are the responsibility of the systems engineering function (by a combined team with the program manager to balance the resources within technical constraints), which is typically performed by a chief engineer or architect to ensure all functional IPTs are communicating and sharing information for related design, integration, testing, etc. In a larger project, a dedicated systems engineering sub-team can be established to facilitate cross-IPT communications and data exchange. Figure 6, below, depicts an organizational chart of teams for smartphone development.

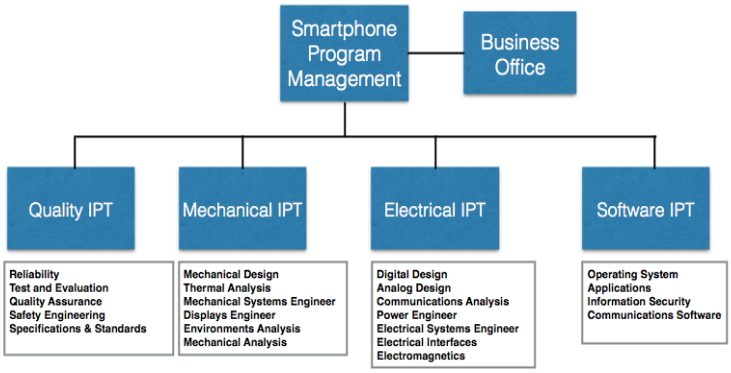


Figure 6. A Sample Organizational Chart (Smartphone)

A Work Breakdown Structures (WBS), shown in Figure 7, can be used as a sample framework to promote communication between systems engineering and project management and to provide a logical structure for allocating work assignments.

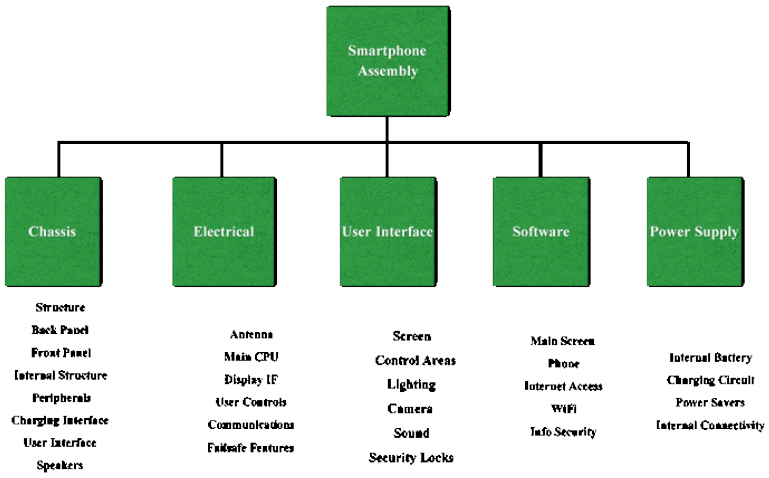


Figure 7. A Sample Work Breakdown Structures (Smartphone)

Requirements

Systems engineers consider both technical and contextual conditions. Requirement development depends crucially on the cooperation between the team (systems engineers) and the competition event organization (rule committees, judges, supports). Using the given requirements as the goal, the team can perform a systematic approach to construct a comprehensive set of requirements that are sensitive to cost, schedule, and operational effectiveness. Below is the list of questions to help facilitate the thinking process of requirement development:

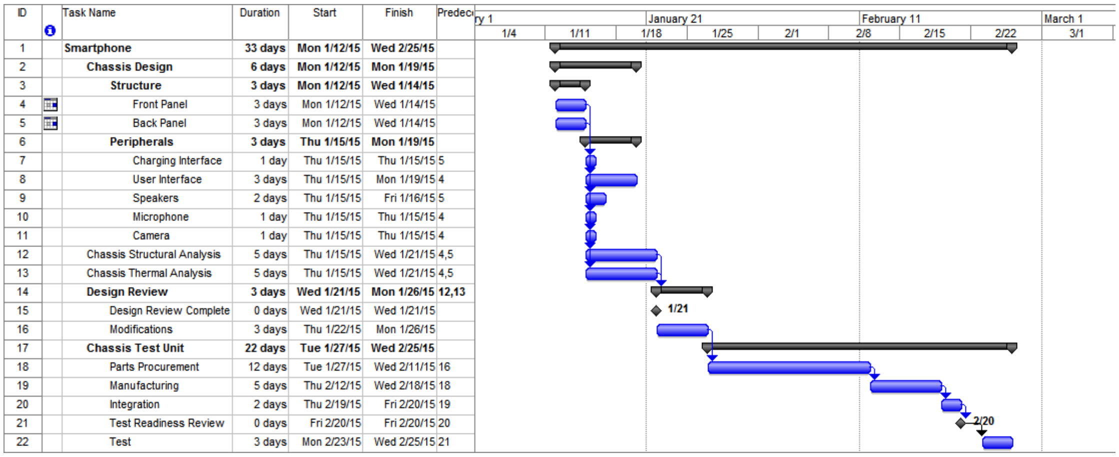
- What are the system requirements? What does the system need to do? Do the requirements cover both functional and operational (robot/operator interaction) areas? Are the requirements reflecting the winning strategy defined previously?
- Can the requirements be broken into groups of functions?
- What is the plan for how these groups of functions can be tested?
- If the system is going to be built on an existing system/products, what are the system/products currently designed to do? What are the limits of existing practice? What new capability can the team add? How does the new capability help the team to win?
- What solution options may be appropriate? What option is better/best? (analysis of alternatives, decision analysis & rationale)
- Can the defined requirements be traced back to the completion rules and policy (traceability)? Any redundant or unnecessary requirements?
- Sample System Description can include: [4]
    - Functional and Data Model
    - Task List
    - Architecture View
    - Boundary and interface view
    - System breakdown view
    - System elements
    - Grouping system elements

Schedule

Being able to plot out a schedule using a software package goes a long way towards helping plan a project. There are several tools available on the market for doing so, ranging from standard models to more sophisticated modeling used by industry. While a schedule can be constructed on a spreadsheet, a scheduling program has the capability to show links between tasks and even adjust related dates based on changes. Below are some sample contents for schedule planning:

- When is the contest? When are reports and presentations due?
- What are the main event/task milestones, including planning, calibration, testing, test setup, purchasing, assembly, presentations, etc.?
- Where will the contest be held?
- Identify holidays and school breaks

A portion of an example schedule is shown below in Figure 8:



ID	Task Name	Duration	Start	Finish	Predecessors
1	Smartphone	33 days	Mon 1/12/15	Wed 2/25/15	
2	Chassis Design	6 days	Mon 1/12/15	Mon 1/19/15	
3	Structure	3 days	Mon 1/12/15	Wed 1/14/15	
4	Front Panel	3 days	Mon 1/12/15	Wed 1/14/15	
5	Back Panel	3 days	Mon 1/12/15	Wed 1/14/15	
6	Peripherals	3 days	Thu 1/15/15	Mon 1/19/15	
7	Charging Interface	1 day	Thu 1/15/15	Thu 1/15/15	5
8	User Interface	3 days	Thu 1/15/15	Mon 1/19/15	4
9	Speakers	2 days	Thu 1/15/15	Fri 1/16/15	5
10	Microphone	1 day	Thu 1/15/15	Thu 1/15/15	4
11	Camera	1 day	Thu 1/15/15	Thu 1/15/15	4
12	Chassis Structural Analysis	5 days	Thu 1/15/15	Wed 1/21/15	4,5
13	Chassis Thermal Analysis	5 days	Thu 1/15/15	Wed 1/21/15	4,5
14	Design Review	3 days	Wed 1/21/15	Mon 1/26/15	12,13
15	Design Review Complete	0 days	Wed 1/21/15	Wed 1/21/15	
16	Modifications	3 days	Thu 1/22/15	Mon 1/26/15	
17	Chassis Test Unit	22 days	Tue 1/27/15	Wed 2/25/15	
18	Parts Procurement	12 days	Tue 1/27/15	Wed 2/11/15	16
19	Manufacturing	5 days	Thu 2/12/15	Wed 2/18/15	18
20	Integration	2 days	Thu 2/19/15	Fri 2/20/15	19
21	Test Readiness Review	0 days	Fri 2/20/15	Fri 2/20/15	20
22	Test	3 days	Mon 2/23/15	Wed 2/25/15	21

Figure 8. A Sample Microsoft Project Schedule

Budget Spreadsheet

Below is an example budget spreadsheet that includes both the income and expenses for a project:

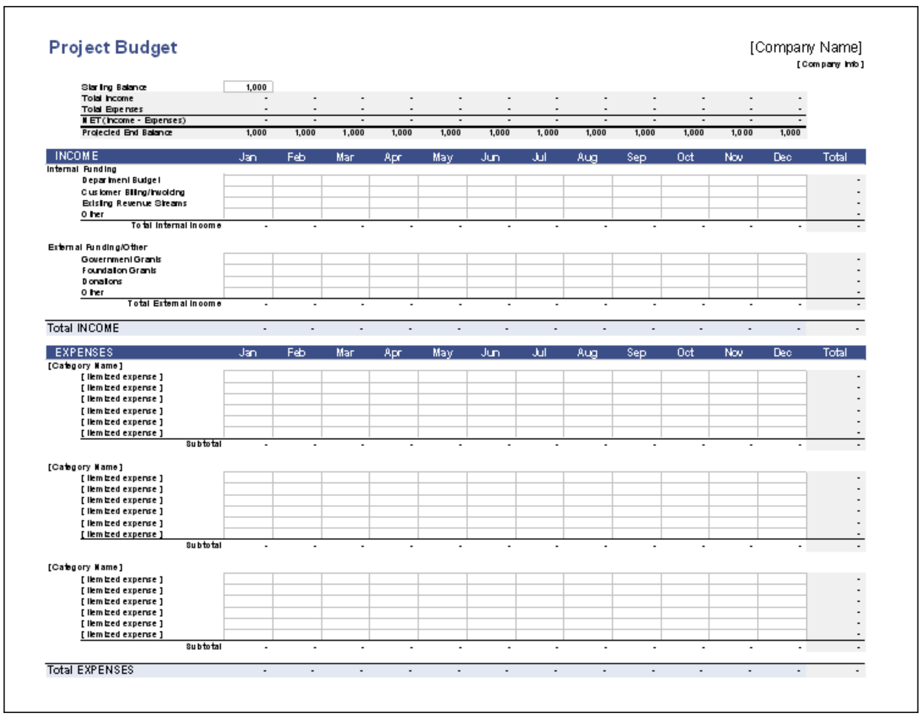


Figure 9. A Sample Budget Spreadsheet

Design and Implementation

Although most robotic competitions provide clear descriptions on what to build and how to gain extra points in competition, some rules or event properties may be relatively abstract and intangible. To treat them properly, good systems engineers must translate the corresponding

requirements clearly and with performance metrics to measure them. Transforming insights, data and requirements into a tangible system vital for differentiating your team from others. What would the perfect system be like?

- What constraints keep the team from building the perfect system? How to pick the best/most workable ideas?
- What are the things that **flow** in this system? What information impacts these flows?
- What are the things that are **stored** by this system? What influences the stock levels? What are the **outputs** of this system? What are the **inputs**? Which outputs are fed back into the system (become feedback?)
- What physical components will be needed?
- What kind of programming language and logic might be needed?
- How will progress be measured? How to verify and validate design? What are the measurable attributes?
- What trade studies are needed to establish the design?
- What models (physical, analog, schematic, mathematical, simulation) can the team use to simulate the end system behaviors?
- How many models are needed for the team to conclude the assumptions and designs?
- Sample steps for building and operating models:
    - Review criteria and constraints
    - Review alternative solutions to be modeled
    - Develop a modeling architecture and build needed sub-models
    - Document all modeling assumptions
    - Validate models either by formal analysis, or by testing
    - Ensure assumptions are consistent across all models
    - Define trade-off areas that will be explored for each alternative.
    - Operate the model and document results, problems, and questions.

Two useful references can be found at:

- http://clexchange.org/ftp/documents/system-dynamics/SD2009-02SomeBasicConcepts.pdf
- http://www.systemswiki.org/index.php?title=System_Dynamics_Modeling_Overview

Integration and Testing

This is included in the stages of system development:

- What are the plans for verification during integration and testing?
- What test equipment or environment is required? What is optional?

- What are the causes and effects related to this system? Are there causal loops? Can the design provide ways to bypass constraints and contradictions?
- Is there a need for successive functional modularization?
- What scenarios can be envisioned for the integration process? If everything goes perfectly, what would the results look like?
- What can go wrong during the assembly process? How can the team ensure that things go right?
- Are there any needs for modifications or additional details on the flow charts, diagrams, sketches of the prototype, ideas, etc.?
- What is highest risk change, new design practices/methods, modified algorithms, modified programs, or modified controls needed for this phase? What are the cross-functional information flow, inputs, and outputs?

Competition

There is no single formula for winning the competition. Hypothetically, a team with more resource or less constraints can choose more aggressive approaches in the arena. However, there are some simple principles applicable to most competitions:

- Building a simple and robust, yet effective robot system is key to doing well at competitions.
- The system must be easy to maintain and have a minimum reliance on special or customized parts.
- Sufficient practicing the operations should be applicable not only to the main operators in the arena, but also to the stand-by members.
- Everyone on the team must be prepared to be a captain. There are always surprises during the long competition season, especially an international event that requires multiple-tier competitions.
- When dealing with complex multiple-tier competitions, each of the required subtasks will be weighted by importance to the team (for instance 20% for starting, 20% for ending, 40% scoring, etc. and first match weighted 5%, second match weighted 10%, and so on). It is critical that the team understands the rules well enough to enable their operators to dynamically change course as necessary during the competition to either achieve higher gains or minimize impact to the current strategy.
- When participating in a highly competitive event, a "scouting system" sometimes can provide insight into the current situation and allow the team to make sensible configurations and operational decisions for the coming matches. Quality scouting can be a winning tool for developing the strategy for every match, not just eliminations but also choice of an alliance (rule dependent). A simple Excel spreadsheet that

(quantitatively and qualitatively) reflects the behaviors of other teams' performance helps the captain and the team to sort, filter, and manipulate their tactics to achieve higher scores.

- Finally, "Chance favors only the prepared mind!"

Closeout

Systems engineering combines technology development with strategic management, to plan ahead with long-term vision. Systems engineering knowledge and skills are about an interdisciplinary execution of a project attempting to articulate, rationalize, and develop best managerial practices, especially in high-technology areas. Observations of what worked, and using scientific methods, retrospective analysis, feedback from the judging team, identification of what didn't work, and tabulation of lessons learned are a few key actions the team can carry on for next competition or to pass on to future teams' development and roadmap planning. Some sample actions (after the conclusion of an event) are:

- What did the team learn in this event (or match)? This includes strategy, tactic, team dynamics, leadership, and so forth.
    - Elaborate problem-solving techniques learned
    - Identify the computer tools used for this project (CAD tools, Word? Excel? Power point)
    - Identify the mechanical and electrical tools that were critical during the competition
    - How was the team organized during the competition? Was it beneficial and effective? What improvements should be made?
- What did the team learn from this project? This includes science, math, engineering and also team dynamics, leadership, and so forth.
    - Identify the math skills learned and used for this project
    - Identify new vocabulary and concepts learned in studying for this project
    - Identify challenges, both team and design challenges, how they were met
- What went right? What went wrong? How can this team or future teams do better at next match/competition? Tabulate any process improvement suggestions.

Reference:

[1]. http://www.nrel.gov/docs/fy12osti/52616.pdf
[2]. Systems Engineering Handbook A Guide For System Life Cycle Processes and Activities, V3.2
[2]. Guide to the Systems Engineering Body of Knowledge (SEBoK) version1.0
[4]. DODAF - DOD Architecture Framework Version 2.02