Team structural elements include:

1. Communication.
2. Boundary maintenance.
   A. For material systems: Aside from possible internal ware, the boundary between the system and its environment often degrades quickest.
   B. For team systems: A team exists for a purpose and must maintain its boundary its purposeful boundary to ensure it remains efficient and effective toward its purpose.
3. Systemic linkages and internal dynamics.
4. Standardization and procedures.
5. Social coordination.
   A. Intra.
   B. Supra.

What are the 'resources' the InterSystem team has access to, and potential control over?

1. Intermediate economic services ("goods") may be resources.
2. The basic resources such as materials and energy are taken from nature.
3. Human effort, as contribution, could be a resource. Resources human physiological energy energetic component.

The InterSystem Team must have significant depth and breadth of technical expertise to review, evaluate, and operate a significant majority of design considerations. Areas of technical expertise necessary for proper Habitat InterSystem operation include, but are not limited to:

1. Human factors and human engineering (including crew workload and usability, human-in-the-loop evaluation, and human error analysis).
2. Human health and restorative measures.
3. Environmental health.
4. Safety.
5. Systems engineering.
6. Human functions and habitability functions (including nutrition, acoustics, water quality and quantity, etc.; i.e., the subsystems themselves - architecture, fabrication, computation, etc.).
7. Human interfaces and information systems.
8. Maintenance and housekeeping.
9. Materials cycling.
10. Exploration operations.
11. Mentoring and training.

A project's view of society as a habitat service system may include:

1. Project team size.
   A. The team is composed of x members.
2. Project duration.
   A. The total duration can range from [identify on schedule], if the information is known.
3. Project requirements.
   A. The total (or evolving) set of requirements to be completed by intersystem team human members and/or machines.
4. Early termination of project.
   A. The project can terminate when the lowest level of project success criteria is met.
5. Role of habitat service support operations in project.
   A. The habitat is a controllable, real-time sensitive operational potential. It is the role of intersystem teams to coordinate the real-time controlled operation and coordination of the global habitat service support system.
6. Human habitation.
   A. Human habitation capabilities include the multi-purpose, integrated habitat service module (i.e., A city) duplication and operation.
7. Sample return.
   A. All monitoring for demand or hazard must be performed transparently.
8. Project team timeline.
   A. The timeline and schedule for a project.

13.17.1 Socio-technical contribution

A.k.a., Contributor, technician, engineer.

An 'InterSystem Project Team' is a project team because they have the knowledge and skills, and have been contributively assigned accountability for some particular role.

There are two general types of socio-technical contribution:

1. Technician scientists (sometimes technicians, sometimes not).
2. Technician operations (technicians).

Any contributor on an InterSystem Team is a technician/engineer. However, technician scientists can also be open source contributors anywhere within the Community, and not part of the InterSystem Team engineering-technicians service.

In Community, there is an integrated socio-technical team to coordinate, develop and operate, the societal system. That team of consists of individuals (who may at the base level be considered 'designers', the human InterSystem Team) and computers (who may at the base level be considered computational InterSystem Team service support systems). The team optimizes their environment (mostly, cities) through intentional algorithmic thought (i.e., through the intentional design

of a socio-decisioning protocol).

13.17.2 Socio-technical team viewpoints

For any coordinated socio-technical, there are multiple viewpoints (information sources) through which work is coordinated.

The three team-oriented views in community are:

1. **The community user's view** - the view of any given individual user of services in the community.
2. **The InterSystem team's view (technicians, engineer's view)** - the view of any given contributor, who is part of the InterSystem team, and developing or operating a community service(s).
3. **The unified information systems view** - the view of the whole, unified information system (i.e., the view of all information sets, as much as possible).

There are four project deliverable (work output) viewpoints (information sources):

1. **Time view** - what is Δt for actions in planned sequence (i.e., has temporal possibility for experience?).
2. **Location view** - what is physical coordinates for time-bound actions with resources (has material possibility for experience).
3. **Resource view** - what is material composition for physical-bound objects (has touch, interfaceable?).
4. **Service view** - what is functional usage for object-bound relationships (i.e., real world entity-objects; has shape?).

There are three project deliverables (output information sets) for any socio-technical system:

1. **Project [information set] viewpoint** - the relationships between operational and capability requirements, and the various projects being implemented. The project information set visualizes dependencies among capability and operational requirements, system engineering processes, systems design, and services design within time.
2. **Services [information set] viewpoint** - the design for solutions articulating the expressed system (including: actors, controllers, performers, activities, services, goods), and their input-output resource transfers between systems, all of which provides for supporting operational and capability functions.
3. **Systems [information set] viewpoint** - the design for solutions articulating the [service] systems purposeful[ly expressed] existence, their composition, interconnectivity, and context providing for or supporting operational and capability functions.

The engineering viewpoints on a project are:

1. **Technical standards view** (TV, knowledge data added) - a set of deliverables (information sets, products) that define technical standards, implementation conventions, rules and other prototypical criteria for the design and/or operation of systems. Note that when a technical standard is applied to operations (to be executed at some time), then it is generally called a 'protocol' or 'procedure'. Protocols and procedures are perceived within this view. Known safe ways of designing and constructing systems.
2. **Operational view** (OV, time data added) - a set of deliverables (information sets, products) provide descriptions of the tasks and activities, operational elements, and information exchanges required to accomplish the intended direction [of change]. Standardized ways of co-operating [service] systems.
3. **Systems and services view** (SV, location data added) - a set of graphical and textual deliverables (information sets, products) that describe systems and services and interconnections providing for, or supporting, directional functions.

In a community-type society, the principal systems and services view is that of the local habitat service systems (cities), which form a globally network habitat service system (Read: city network). SV data focuses on explaining (reasoning/justification) how the purpose for specific actualized systems with specific physical and/or digital (hybrid) locations is met by objects and relationships (often through UML). The relationship between data elements across the SV to the OV can be exemplified as systems are developed and operated to support individuals and organizations (their operations).

The unified project-engineering viewpoint:

- **All view** - a view that provides a unified, integrated, whole, overarching description of the life-cycling system (i.e., the whole, socio-technical, information-material life-cycling system).

The common supplemental viewpoints that ensure an accurate alignment of understanding include:

1. **Capability viewpoint** - articulates the capability requirements, the delivery timing, and the deployed capability.
2. **Data and information viewpoint** - Articulates the data, data relationships, data alignment in a structural format that expresses content for

the capability and operational requirements of a system through system engineering processes, and systems and services tools and techniques.

13.17.3 [Societal] InterSystem team work **rotation**

A.k.a., Individual rotation, team rotation, etc.

When there is intention and consequences, then there is a team [of "stakeholders"]. Teams work with information and material that are integrated into a materially extant system in which 'life' exists (i.e., living organization occurs -- there are living systems). Components of the living organization come together to form a team organization [to most effectively and efficiently fulfill]. The team has the potential [capability] of recognizing the discoverable information-base of existence. The team then has the potential of optimizing the organizing its societal information construction system for highest fulfillment of each and every individual. Within any given organization, work is scheduled out to project teams.

Project teams deliver projects.

> **NOTE:** *Teams function optimally when they do the right then at the right time.*

13.17.4 [Societal] InterSystem team work **effectiveness**

A.k.a., Team effectiveness.

In order to be effective at scale, and hence at the societal InterSystem level, teams must have the following:

1. A shared understanding of the situation.
2. A shared direction.
3. A shared orientation.
4. A shared approach.
5. A shared informational environment.
6. A shared material environment.

Fundamentally, in order for teams to execute solutions effectively, the two team must work off of a single specification for the work.

13.17.5 [Societal] InterSystem team work **roles and responsibilities**

A proper functioning socio-technical system requires the co-ordination of the actions of all roles involved in operation.

The core InterSystem team organizational (structural) role is:

- Technician (a.k.a., socio-technician, engineer, operator) - a technically skilled contributor.

The primary roles involved in operations are:

1. **InterSystem contributing teams (technicians, engineers)** - assigned work for the coordination, development, and operations of the whole societal system through individual contribution upon an InterSystem Sub-Team. There are three conceptual dimensions of contribution to an InterSystem Team; three separately together functional roles. Coordination is an operation that sustains all development and operations. Coordinators coordinate the optimal allocation and timing of all resources and access. Developers test and develop the next iteration of the whole societal system. Operators execute upon the selected societal solution, to either implement a new solution or serve some humane function within the societal service system.
   A. **Coordinators**.
      1. Socio-parallel [project] decisioning - coordinate societal resource access decisions in alignment with a value orientation.
   B. **Developers.**
      1. Socio-technical [solution] decisioning - design societal service systems composed of resources in alignment with optimal socio-technical safety standards.
   C. **Operators.**
      1. Socio-technical [solution] executioning (Read: execution decisioning) - operate service system for user through a standardized optimal procedure.
      2. Recursive (all roles are sources of 'operator' information).
2. **Individual human accessors** - selectively access services (and service objects) as the outputs of InterSystem Contributing teams.
   A. **Users (the Community of)**
      1. Usage of end service, or service object [as designed and operated].

13.17.5.1 Professional team roles and responsibilities

The common "professional" roles and responsibilities of teams designing, developing, and operating integrated service systems include at a high-level, but are not limited to:

1. **Issuing entities** - The individual(s) with an issue that instantiates the requirement for a project.
2. **Developers** - The individual(s) whose responsibility is the development of the system for the project.
3. **Information analysts** - The individual(s) skilled in resolving [societal] information inquiries being used in the project.

4. **Definition analyst**: The individual(s) skilled in the development and definition of the computational controls of the project environment.
5. **Leads**: The individuals accountable for all aspects of the system design and construction.
6. **Subject matter expert (SME)**: The individual(s) and system(s) who have the knowledge and skills necessary to implement the project.
7. **Project coordinator**: The individual(s) responsible for all activities of a project. The project coordinator plans, controls, and coordinates a project.
   A. **Quality assurance analyst**: The individual system who audits and approves project deliverables from a QA perspective. Reviews plans and deliverables for compliance with applicable standards. Provides guidance and assistance on process matters and defining standards. Primary focus is on defect prevention.
   B. **Quality control analyst**: The individual system responsible for checking the product or service after it has been developed. Primary focus is to find defects.
8. **Training coordinator**: The individual system who is the key person and point of contact, interface, for all training required for the project.

13.17.5.2 Trainee team role

A.k.a., Apprenticeship, mentorship, internship, residency.

The trainee team role is that of someone who is training upon an InterSystem Team.

13.17.5.3 Hazard isolation roles

Because life-threatening failures may occur when working with existent systems, humans must design their systems so that hazards can be isolated and systems can be restored. In concern to materials, for example, remote placement of hazardous materials, redundant containment, and clean-up material are a few options for reducing risk. An emergency shower will, for example, isolate dangerous chemicals into a liquid contain from a contaminated human. Every InterSystem Team should be able to avoid or secure hazardous systems with which they work.

Additionally, the concept of hazard isolation applies to the encoding of the value of 'justice' in any society. When a human becomes a "hazard" (danger, risk) to others, they "isolate" that human. The concept 'isolate' carries two orientations -- an orientation that restores fulfillment relationships, and another orientation that does not -- isolation from presenting a danger to society, by means of:

1. Isolation from supportive and restorative relationships (as in, restorative justice), and hence, isolation from structural feedback.
2. Initial isolation for physical safety, with the application of supportive methods and restorative relationships so that the self-organizing entity can re-orient itself toward fulfillment, releasing its societal requirement for any core form of isolation.

In early 21st century society, police take the role of law enforcement and are the represent the service that physically isolates hazards to society. In a market, society is structurally composed of 'property' (an abstraction), and thus, a core part of the role of the police is protection of 'property' from hazard. In community, there are individuals trained and accountable for isolating both technical and human hazards in the environment. At the human level, however, the concept of 'police' does not precisely apply, and their role as isolators of individual-human hazards to society, would be accounted for by medically trained personal, who are more like EMTs (emergency medical trained) personal who also have training in detaining humans), versus the conception of 'police', which entails politics, jurisdictional laws, authorities, property, psychological combatant training, jail, prison, etc. (none of these exist in Community as they are commonly defined in market-State societal configurations).

13.17.6 [Societal] Intersystem team work tasks

A.k.a., Intersystem team work/actions.

Societal design is the accountability (responsibility) of the community, and therein, the InterSystem Team:

1. **Habitation-related tasks** - includes tasks associated with sustaining and evolving the services provided by the controlled habitat system. These tasks are divided into a priority matrix between sub-system service and operational process priority. Tasks that are directed at the long-term viability and ultimate fulfillment of humanity.
   A. **Automation and maintenance** - Automating routine habitability tasks, while still allowing for InterSystem intervention, will be a high-priority development capability for all of these systems to allow a reduced human workload. This would free up human time for higher-priority tasks, while yet retaining the ability to control systems as needed in the event of problems.
   B. **Redundancy** - Redundancy management (RM; monitoring) will be employed in the selection of backups to replace failed or degraded systems, or to manage the rotation of redundant systems to equalize hours of operation. Some systems will have one or more identical backup units, ensuring physical redundancy. Other systems,

for which there are no physically identical replacements, may have their functions assumed by non-identical systems, ensuring functional redundancy.

C. **Operating** - Doing any job in any habitat as part of a contributor to the InterSystem team.

2. **Scientific discovery endeavours/tasks** - includes field and laboratory tasks associated with answering the principal scientific questions.
3. **Skill areas** - These functions include command and coordination, routine, and contingency operations. Specifically, coordinating (piloting and navigation), system operations, system maintenance, repair of systems, and upgrade of system).

13.17.7 [Habitat] InterSystem team work **service structure**

The intersystem team habitat service structure:

1. Habitat service system design - open source and collaborative.
2. Habitat service system selection - preference criteria based on local population (cannot modify base functioning) customized layout, aesthetic, sub-services, timing, type and availability per demand and localization (location + control of location).
3. Habitat service system integrated and selected for design actualization. The solution inquiry process resolves.

Habitat services are, in significant part, formed from human needs; and hence, they are met continuously through a network of habitat service systems, which in and of themselves, have operational [InterSystem] deadlines (as in, the priority scheduling of tasks) in order to maintain themselves and adapt. In community, if "we" don't adhere to the deadlines "we" set, then our own services will likely fail.

13.17.8 [Inter-societal] InterSystem team work **roles and responsibilities**

The structural organization of human relevant roles and responsibilities may be relationally visualized through an organizational "breakdown" diagram on the part of an InterSystem project coordinator.

Due to the societal spanning nature of this project, its organizational structure necessarily interfaces separately with each type of society: the market-State and the community. Additionally, due to the presence of a larger global audience and the necessity for maintaining contractual agreements in the market-State, there is an Executive Steering Committee. In order to coordinate between these three divisions, a Main [Project] Coordinator (or, coordination system) exists.

The main societal coordinator is responsible for coordinating the flow and integration of information and materials between the three organizational divisions:

13.17.8.1 Division 1: Executive steering committee

The Executive Steering Committee is responsible for oversight, direction, and final project decisions related to market-State interaction . The Executive Steering Committee is responsible for communicating market requirements, State requirements, and human requirements to all stakeholders, while facilitating the resolution of any potential issues or changes that threaten the completion of the project.

13.17.8.2 Division 2: Market-State interface structure

The interface with the market-State society is an active societal construct engaged with on behalf of the Community (via this project) through project coordination. The market-State is interfaced with through electronic-jurisdictional contracts. The market-State must be complied with in order to access resources only available through the market-State.

13.17.8.3 Division 3: Intersystem team structure

Work upon the societal community system is organized through an InterSystem Team structure. The InterSystem Team structure is divided by three different primary system processes (coordination):

1. **Design (Informational specification and standardization)** - Responsible for the specification deliverable.
2. **Implementation (Material operation)** - Responsible for the operational deliverable.
3. **Social (Awareness and Sharing)** - Responsible for the social population deliverable.

14 [Project] Schedule

A.k.a., Time planning, plan timing, project timing, task timing, time logistics, the scheduling problem, time coordination, time management, time mapping, calendar mapping.

A schedule is a timeline of events. Scheduling is the process of deciding (and otherwise, coordinating) which of a given operation set gets performed, and when, on a given system set. To schedule is to setup a specific time when some event will occur. Scheduling is the process of coordinating (arranging, controlling, and optimizing) work and workloads in a production process. Scheduling is used to allocate resources, plan human contribution, plan production processes and acquire materials. Scheduling is an assignment over time of operations to systems, called a schedule. A schedule is the output of the scheduling process. More simply, scheduling is the process of coordinating work schedules (of humans and machines, a socio-technical system) to meet human requirements expressly input as deliverable activities.

Within the planning process, scheduling is the process of determining when tasks must be completed; when they can and when they must be started; and which tasks are critical to the timely execution of the project. A complete schedule is a function of total effort and resource allocation.

Scheduling (and the resulting schedule) are often considered a tool that defines what tasks are to be done, when, and by whom. Schedules define and track the progress and completion of a project.

A project timeline is most often called a Gantt chart. It is possible to add any schedule/time associated project variable to a project timeline; however, project timelines most often identify project milestones and tasks. Progress bars are included in timelines to identify the progress of a task(s) to a milestone(s).

The quality of any schedule is measured by its principal objective function:

1. The operational [project] completion function - Is there the state of 'completion' of the last operation (i.e., is the last operation complete)?
   A. There are # of tasks to be scheduled. Each task consists of one or more operations (processes). These operations must be scheduled on # of systems. The completion time of a task is the first point in time at which all of its operations are completed. The objective function (of scheduling) as an optimization function, involves minimizing either the completion time or the number of machines required to complete all the tasks by some specified deadline.

When working in a material environment, there is time-based information:

- **Scheduling [a time-planning solution]** is the process of calculating and assigning an arrival time for each deliverable (stop, output, etc.), with workers (transfer entities, contributors, etc.) being assigned time-bound roles (shifts) that adhere to working hours.

When scheduling in a material environment, there is also a location-based scheduling structure:

1. **Routing [a location-planning solution]** is the process of mapping out the unique paths (ways) that one or more transfer entities will take while they deliver or collect resources from each of their stop (deliverable) points. This involves considering the sequence of stops (deliverables), and the ways (approach, method) that will be taken by each transfer entity to successfully achieve this outcome.
   A. **Route optimization** follows logic steps, and is the process of analysing the projected routes and refining them to be more (or, most logically) efficient. This can be achieved by taking all physical and temporal relationships and locations into account and calculating an optimal path, given extant conditions.

NOTE: *Takt time describes pacing work to match the user's demand rate. Takt time planning then, is one method for work structuring around a set pace of work.*

14.1 Time

Time is the universal matrix of experience and activity. Time is an open matrix of possibilities for present action. In community, people decide what to do with their time together, to express their highest potentials and values. What is the experience of time, when labor is no longer alienated, but freely chosen? Contribution.

TRUISM: *Because people can decide what to do with their time when they experience time as free, it does not follow, for a socially self-conscious person (agent) aware of his/her mortality, that any use of time s/he decides is "good", is good (i.e., actually aligned with a common value set). People through imaginative reflection and projection, distinguish themselves from the social forces acting on them and decide as socially self-conscious agents what they will do. However, just because people can decide what to do with their time when they experience time as free, it does not follow for a socially self-conscious agent aware of his/her own mortality, that any use of time s/he decides is good, is good.*

14.1.1 Time duration

A time 'duration' is how long something (a task/activity) takes to complete. Duration can be visualized along a timeline (i.e., the chronological ordering of events).

14.1.2 Timing

Timing refers to performing an activity at the 'right' time, either according to a planned frequency or in response to an event.

14.1.3 Temporal-spatial coordination is scheduling

There is a requirement to identify time as well as location in environmental representation. Time is a concept perceived as the continued iteration ("progress") of existence, measured by an observer as events that are ordered relative, as "before" or "after", and which, at a given point in time, give rise to the notions of past, present and future. Time and location are often used together by an application to describe when a given condition exists or when an object was present at a given location (read: an objects epoch).

14.2 Deliverables-based project schedule

A deliverables-based project schedule facilitates the process of a project system:

1. Definitions.
2. Work-deliverable breakdown structure.
3. Schedule tasks.
   A. Enter all tasks.
   B. Determine predecessor tasks.
   C. Estimate the work.
   D. Estimate the duration.
4. Assign resources.
5. Add constraints.
6. Identify and operationalize contributing entities.

A spatial-temporal view of a set of operations for scheduling must include:

1. Set time, date, and location (1 operation).
2. Reschedule operation.
3. Postpone operation.
4. Change location operation.
5. Delete operation

14.3 Computing and scheduling

Computers and scheduling are closely related in two dimensions:

1. Assignment of operations on a machine is called a schedule.
2. Coordination in a multi-variate environment is more efficient and effective using computers do scheduling via computation more efficiently and effectively than humans do scheduling via cognition.

Additionally, there are three essential information characteristic sets associated with schedule computing:

1. Task characteristics (job characteristics).
2. Mechanism characteristics (machine characteristics)
3. Objective function characteristics (process characteristics).

The performance of a scheduling solution will likely fall into one of a number of possible categories, the most optimal of which is, generally:

1. An optimal solution in an amount of time proportional to a polynomial of the problem size.
   - $\leq Kn^k$
   - Wherein, n is the problem size, and there are constants K and k, which are independent of n (given, the problem is solvable in time).

There is a class of problems that can be solved in polynomial time (P, time-determined problems), and the superset of this class of problems non-deterministic problems (NP). NP is bound by a set of problems that can be solved by search or enumeration of a tree whose depth is itself bound by a polynomial in the problem size. (Lagerholm, 1998)

> **INSIGHT:** *Time could be viewed as that which is universally scarce.*

14.4 Schedule (timeline)

> *A.k.a., Gantt chart, project schedule, project timeline, calendar, project calendar, timeline, task-time network diagram, timetable, itinerary, time plan, planned time.*

A schedule visualizes activities in time; laying out the work and its phases on a calendar, mapping time-relevant items onto a calendar. All schedules are schedules of activity (as any action, task, work, deliverable, etc.) with all associated time information. A schedule is used to account for working, together, with real-world [resource-based] systems through time. A 'schedule' lists all project activities in time. Activities all start with verbs (what is to be done as an action). A [project] schedule is all project activities, dependencies and resources associated with time. The system records and tracts time, resources, and effort on the project. A schedule coordinates between time, activities, and a projects the resources (people, equipment, location) required to execute project tasks. A schedule is a "living" interface for coordinating and estimating work together.

A project schedule (a.k.a., gantt chart) visually combines project information essential to the coordinated

execution of the tasks in time and space, with people and resources (and in the market, money). A Gantt chart is one type of organizational chart which could be used to convey the Action, Time and Finance plans of and between workgroups. Once the work is broken down by tasks and sub-tasks (i.e., the WBS is delivered), the project coordinator will [process information to]:

- Arrange these tasks in temporal order.
- Schedule them out to InterSystem Teams and HSS service systems.
- Identify dependencies (inquiry: does the start of one task require the completion of another task).
- Highlight the completion points [on the diagram] of critical tasks (a.k.a., "milestones").

Scheduling involves the relating future events (activities, tasks) in the real world to some linearly sequenced coordinate system called 'time'.

Generally, a schedule lists the following work data:

1. Activities (tasks, work).
2. Deliverables (outputs, outcomes, products).
3. Phases (milestones, stages).
4. Time points and durations (start and finish dates).

A schedule may visualizes (at least) the following (i.e., the following are mapped onto a calendar):

1. What are the deliverables.
2. What are the tasks (work items to produce deliverables).
3. Where are the completion/integration sign-offs for the deliverables.
4. Who is responsible.
5. Who is accountable.
6. Who is consulted.
7. Who is informed.

In concern to work, a schedule visualizes:

1. The current activities and future activities on the timeline.
2. The current status of a project.
3. All other projects that any given project relates to.
4. All work packages in a project that have a time reference, such as phases, tasks, and milestones, as well as, relationships between them (i.e., all work packages, phases, milestones, tasks, and bugs/issues in a timeline view). Phase is a label for a set of linearly related tasks (e.g., development or planning).
   A. The work packages can have a start date and due date.
   B. Milestones may only have a due date.
5. All preceeds and proceeds between different work packages.

Any schedule is necessarily associated with data on users [of the schedule, contributors] and resources [accessible via the schedule]:

1. Contributed accountability coordination (worker identity)
2. Available resources coordination (resource identity)

A schedule is a type of chart that involves time:

1. A chart is a visualized display of data-base[d] information.
2. A schedule timeline ("gantt" chart) displays tasks as horizontal bars across a calendar (time-cycle), creating a visual representation of the project schedule, and other time-relevant.

A complete schedule may be calculated as a function of total work and available resource allocation. The schedule for a project is the timetable that specifies when each activity should start and finish.

An effective schedule is:

1. Understandable (visual).
2. Sufficiently detailed.
3. Highlights critical tasks.
4. Flexible.
5. Based on reliable estimates.
6. Conforms to available resources.

14.4.1 Elements of a schedule

Define the schedule's data structure as a list:

1. **Work breakdown structure** - a detailed list of [project] activities and [creation/development] tasks.
2. **Historical information** - from similar projects and other lessens learned.
3. **Personal calendars** - information from project contributors about their own time commitments.
4. **System calendars** - information on calendar events, significant common durations of time (e.g., holiday, vacation, work, cycle, maintenance).
5. **Resource planning and coordination** - the number of people available to the project.
   A. In community, there is the construction of a set of adaptive services that fulfill human need, want and preference. In the initial construction of the, hence forth, continuously operational habitat service system (part of the total societal system), there will need to be agreed upon dates for delivery of specific outputs. And, during operation, there will be maintenance

and replacement requirements, which will have static delivery dates [before urgency criticality is raised]. Individuals and systems agree on dates for the delivery of specific outputs, with a degree of flexibility relative to the task priority requirements themselves.
    1. In the market, there are milestones, or agreed on dates for the delivery of specific outputs.
6. **Visualize the schedule** - ready for inquiry *process.*
    A. Plan - "define" activity sequence and duration, develop the network integration or unique production diagram, and compose GANTT chart (i.e., the project implementation unique tasks timeline).
    B. Do - Communicate and update schedule core timeline with agreed upon tasked InterSystem Team positions (roles as part of an InterSystem Sub-Team) and tasks.
    C. Check - monitor schedule variances.
    D. Adapt - update the schedule.
7. **Monitor the schedule** - ready for *output.*
    A. Project schedule baseline - what is needed to sustain what degree of fulfillment (high-level categories include, but are not limited to: life support, some degree of technology support, and some degree of recreational-facility support.
    B. Schedule variance reports - when there is a variance from baseline in the scheduled fulfillment of need, and also when there is a variance from baseline in following (for automated and human systems) through with 'standard'[-ized] practices and procedures when contributing as part of an InterSystem Team.
8. **Update the schedule** - ready for *feedback*.
    A. Schedule updates become notifications.

Humans or automated systems, or some combination thereof, can perform [all] tasks. A unified information system allows for the reporting of habitat service's expected functionality. Is life support sustainable, and what are the plans for the systems evolution? The same goes for technical and exploratory service systems; are they meeting expectation and sustainable? Also, planning overlaps with criticality forming a criticality matrix applied to the determination of task priority [in a functional habitat service system].

14.4.2 Process for creating the schedule

The most common process for creating the schedule is:

1. Enter all the tasks (and sub-tasks) - as associated with the identified list of deliverables (from requirements document, WBS, etc.).
2. Determine predecessors (determine dependencies) - as the tasks that legitimately belong linked in an order (resource availability, decisions, outputs).
3. Estimate the work - as who will do the work, and when will the work be done by (accountability and completion date).
4. Estimate the duration of the work (timeline of activities).
5. Assign execution (team availability).
6. Assign resources (resource availability).

14.4.3 Project scheduling time-frame

A project may be a unique (one-time) endeavour, or it may have an ongoing and continuous objective. To some relative degree, of course, all processes (phases, stages, whole projects) have a specific time-frame, or finite life-span, to some situationally relative degree.

14.5 Schedule/-ing coordination

Aphorism: Plan the work, work the plan.

Schedule coordination includes the processes required to ensure timely completion of the project. A Schedule is created using a collaboration-driven estimation method; the reason for this is that a schedule itself is an estimate -- each date in the schedule is estimated, and if those dates do not have the people and their agreement, as those who are going to do the work, then the schedule will be inaccurate. Once the scheduling is in process (for it is continuous throughout), then project coordination involves monitoring the progress of the project and revising the schedule were required.

Schedule coordination consists of a series of tasks and steps designed to help manage the time constrains of the project, the steps are:

1. Defining the Schedule.
2. Publishing the Schedule.
3. Monitoring the Schedule.
4. Updating the Schedule.

Schedule inputs:

1. Work breakdown structure - contains a detailed list of all project activities and tasks.
2. Historical information - from similar projects and their lessons learned.
3. Calendar information - other commitments and calendar events.
4. Resource planning - planning for the collection, integration, and cycling of resources through a system.

Schedule progress conditions:

1. Plan - define activity sequence and duration, develop the network diagram and gantt chart.

2. Do - communicate and update schedule progress.
3. Check - monitor schedule variances.
4. Adapt - update the schedule.

Schedule outputs deliverables:

1. Project schedule baseline.
2. Schedule variance report.
3. Schedule updates.

14.5.1 Scheduling 'state' status

In some cases, the values of quantities included in scheduling have, or have not, been confirmed and are designated as:

1. **To Be Confirmed (TBC)** – details may have been determined, but are subject to change.
2. **To Be Determined (TBD) or To Be Supplied (TBS)** – the appropriateness, feasibility, location, etc. of a given event has not been decided (known, but not yet available).
3. **To Be Resolved (TBR)** - used when there is a disagreement on the requirement between technical teams.
4. **To Be Announced (TBA)** - details may have been determined, but are not yet ready to be announced. Note: This Does not apply in community, because the societal system is open source and transparently generated by the community.

14.5.2 Schedule delays

The whole project [completion timeline] will be delayed if task-deliverables and/or resources are delayed:

1. Schedule **critical path (tasks-deliverables)** - If anything (e.g., any task or deliverable) along this path (timeline) gets delayed, then the whole project will get delayed.
2. Critical **resource chain (resources)** - If those resources which are required are not available (i.e., not present when they need to be) and/or the quality of the available resources is not sufficient, then the whole project will be delayed

14.5.3 Principal schedule constraints

There are three schedule constraints that 'control' when *an activity starts or finishes*:

1. An activity must be completed by no earlier than a specific date - an activity may occur at any time after a specified date, but no earlier then the given date.
2. An activity must be completed no later than a given date.
3. An activity must be completed on a given date, no earlier or later.

14.5.4 Schedule modifications

A.k.a., Schedule timing, schedule alteration.

There are several common ways in which a project's schedule [timing] may be modified:

1. Add more resources - to shorten the time it takes to complete a scheduled activity or event (i.e., "crashing").
2. Do more actions - perform more activities simultaneously (i.e., "parallelization" and "FastTracking").

14.6 Scheduling system and user interface

A complete scheduling system and interface must meet the following criteria (i.e. the schedule coordination process must visualize a project schedule that meets the following criteria):

1. Complete - the schedule must be capable of representing all the work to be done. This is why the quality and completeness of the total information system, and its architecture, is so important.
2. Realistic - the schedule must be realistic with regard to time expectations and the availability of human and system contributors.
3. Accepted - the schedule must be acceptable to (have identifiable agreement from) the individual user.

14.7 Scheduling contribution time

To the InterSystem Team of a community-type societal system, at the highest-level, 'timing' refers to contribution as the selection and follow-through of [a] work [package]:

1. **When (time point)** the communication of an extant work package is distributed to the Community (for community and InterSystem Team contribution)?
2. **How long (i.e., duration)** the work package will take to complete (as whole and/or cycle)?
3. **When (time point; a.k.a., "milestone")** the work package is required to be complete (as whole and/or cycle)?

14.8 Schedule model

A schedule model involves all project information in association with a specifically applied scheduling method(s) and scheduling tool(s). In application, a societally coordinated schedule likely consist of a series of synchronous tasks (and sub-steps) designed

to coordinate [between] the time constraints of any societal-level project.

14.8.1 Scheduling method

I.e., How will control over a schedule occur? Plan the control through the selection of a method.

A scheduling method is a formal procedure that can be applied to any instance of a scheduling model in order to obtain a feasible schedule (i.e., schedule aligned with objectives). A scheduling method solves a scheduling problem. A schedule method is a procedure that takes an instance of a scheduling model as an input in order to produce (At least) one schedule as an output, given a real world situation.

There are a variety of possible scheduling algorithms for scheduling [a set of project] activities [visually] in time. The following are methods for planning schedule control:

1. **Program evaluation review technique (PERT; a.k.a., Critical path method, CPM)** - Using the data below, CPM calculates the longest path of planned activities to logical end points or to the end of the project, and the earliest and latest that each activity can start and finish without making the project longer. This process determines which activities are 'critical' (i.e., on the longest path), and which have 'total float' (i.e., can be delayed without making the project longer).
   A. List all activities required to complete the project (typically categorized within a work breakdown structure).
   B. Identify dependencies between the activities.
   C. Visualize the relationship between all activities in a precedence diagram.
   D. Identify logical end-points, such as, milestones or deliverable items.
   E. Assign time (duration) that each activity will take to complete.
   F. The PERT (PERT chart creation) procedure is:
      1. Tasks (activities) represented as arrows (a.k.a., activity-on-arrow diagram).
         i. For example, "Collect project data".
      2. Milestone (major completion stage, phase, min-max version save) are represented as nodes (Read: circles).
         i. For example, 'No project data' (start node, date) and 'submit all project data' (end node, date).
      3. Estimate of duration of time it takes to complete the activity.
         i. For example, The time duration between start and end nodes that is entirely encompassed by the arrow that represents the task (activity).
      4. Package PERT (i.e., PERT applied) for selection by contributing users and habitat service systems.
         i. For example, The 'instruction' to 'investigate' an 'issue' in a building within 10 minutes in order to prevent a building evacuation.
2. **Critical chain method (CCM)** - After the critical path(s) is determined (Read: calculated with software), resource information is added (also calculated) to produce a resource-optimized schedule, with a resource-constrained critical path.
   A. Determine resource availability - associate resource information, including a resource-precedence diagram, with the critical path.

   CLARIFICATION: *Though confusingly named, 'critical path' is the sequence of project network activities which add up to the longest overall time duration (i.e., the activities that create the longest distance between the start and the finish of a project). The critical path is the longest path through the schedule with either zero or negative total float.*

14.8.2 Schedule estimating

In a sense, every applied input could be viewed as a probability (or, "estimate") of potential input:

1. Contribution availability estimating.
   A. Probability of meeting contribution requirements, given that which is available and known (where, human contribution is the input).
2. Resource availability estimating.
   A. Probability of meeting resource requirements (where, real-world resources are the input).
3. Financial budget estimating.
   A. Probability of meeting financial requirements (where, money or trade is the input).

14.8.3 Scheduling tool

Project scheduling software can perform the scheduling method calculations (e.g., can perform CPM on a data set). A schedule tool is an information function that provides schedule component names, definitions, structural relationships and formats that support the application of a scheduling method (calculation).

15 [Project] Risk

A.k.a., Negatives, harms, threats, hazards, vulnerabilities, dangers, bad events, loss events, etc.

NOTE: *In general, a risk is considered something negative. This section refers to negative risks, versus positive risks. A positive risk is a potential event that might occur could be positive for a project. A positive risk would be an opportunity or advantageous outcome. However, in general, the term risk, when used by itself, implies something negative, implies a danger.*

In the dynamic landscape of project coordination, the successful execution of any project relies on the effective coordination and control of risks. Project risk coordination plays a pivotal role in safeguarding project objectives and enhancing resilience in the face of uncertainties. Risks are statements of what could potentially go wrong or the potential negative outcomes that may impact a project, operation, or objective. They represent uncertainties or events that have the potential to deviate from the planned course and adversely affect desired outcomes. Risk statements typically include information about the nature of the risk, its potential impact, the likelihood of occurrence, and any relevant context. Identifying and assessing risks is a fundamental step in risk management and mitigation. Risk statements are combined into a risk list (a.k.a., risk register), which is maintained as the project is executed. Whereupon risk mitigation strategies/activities may be applied before, during, and possibly even after execution of a project.

INSIGHT: *Risks ought to be reduced.*

When discussing risks, there are several important concepts that must be considered:

1. **Risk (a.k.a., concern, hazard, danger, negative risk, threat)** - exposure to danger. Risk is exposure to danger. Risks are what might go wrong, exposing someone or something to danger. Risks are those events or conditions that are likely to, or are, negatively influencing one or more project objectives, include scope, schedule, cost, quality, or other critical factors. A risk is any factor (or threat) that may adversely affect the successful completion of the project. Effectively, if something reduces the optimal completion of a project, then it is a risk. A risk is an identifiable danger to be overcome with a mitigation solutions before the risk becomes realized (as an incident), thereafter, requiring a more significant resource contribution.
   A. Risk assessment - is a prospective assessment of the likelihood of some incident/danger occurring (risk = rate * time for a bad/dangerous event to occur). A risk is the potential likelihood for something bad to happen.
2. **Residual risk** - risk remaining after all feasible mitigations have been applied. Residual risk is the risk that remains after efforts to identify and eliminate some or all types of risk have been made.
3. **Incident (a.k.a., occurrence, event, episode)** - is an unexpected or undesirable event or occurrence that has already transpired or is currently happening and could disrupt the project or lead to harm or damage. In other words, an incident is the occurrence of a negative event. When something bad has happened, the at-risk scenario became an actuality.
4. **Harm (a.k.a., damage, negative event, adverse effects, consequences)** - is an assessment of the damage that would be done if the risk was realized/actualized. Harm refers to the negative consequences or detrimental effects resulting from an incident or risk materializing. It can encompass damage to assets, financial losses, injuries, or any adverse impact on project outcomes.
   A. Harm assessment
5. **Mitigation (a.k.a., controls, risk reduction, preventative measures, countermeasures)** - a change(s) made, or to be made, to systems to reduce or eliminate a concern/risk. Mitigation involves the actions and strategies put in place to reduce or eliminate the likelihood and severity of risks or incidents. Mitigation aims to minimize harm and enhance the project's resilience.
6. **Issue (a.k.a., concern, problem, challenge)** - in the context of risk, an issue is a problem or challenge that has arisen during the execution of the project, and it often requires resolution or coordination to prevent it from negatively impacting project progress.

There are three common dimensions of risk:

1. **Hazards** and **exposure** (categories of danger):
   A. Human.
      1. Intentional.
      2. Unintentional.
   B. Natural.
2. **Vulnerabilities** (touch-point openings to potential harm):
   A. Social.
   B. Technical.
   C. Resource (economic as *object* or *money*).
3. Lack of [coping] **capacity** - inability to do something or know something:
   A. Infrastructural.

All concerns, risks and issues have a three characteristics:

1. **Likelihood** - of incident occurring - given existing and future expected situation.
2. **Consequence** - of incident occurring - priority response level (if incident occurs under expected conditions), result of damage.
3. **Uncertainty** - uncertainty of risk and risks occurrence. Uncertainty is uncontrollable; however, uncertainty can be reduced with better (more accurate) information.

A qualitative risk analysis is a formula for prioritizing risk-related issues:

- Risk = consequences • likelihood
- Wherein,
  - Risk = consequences (scale-of-harm) • likelihood (% occurrence as, likely to occur how often, every 24 hours)
  - consequences = scale-of-harm triage and prioritization
  - likelihood = % occurrence as, likely to occur how often, every 24 hours

After the risk value is acquired, there is always an [un-] certainty value (a.k.a., confidence level, measured error, etc.) given. The [un-]certainty value is typically a separate value, given to state a relative error-based confidence level -- how confident of not "being in" error is the stated risk value.

Risk likelihood refers to the probability of some risk becoming realized (i.e., suffering some harmful event, which was a risk), for example:

1. Likely.
2. Possible.
3. Unlikely.
4. Remote.

Note that some risks do not use this qualitative classification, instead they go further to sub-classify via counting, how often, and within what time interval the risk is likely to occur and become and real incident.

The consequence scale determines which of three/four bands each risk falls into: red (high), amber (medium), or green (low):

1. **Low (green) risks** – are currently tolerable (often as a result of existing action on them) and do not require specific extra action (although attention may be given to them to ensure they are not being over-managed thereby tying up resources that could be better employed).
2. **Medium (amber) risks** – are also not tolerable although management action is less time critical than red risks.
3. **High (red) risks** – are not tolerable, and will need immediate management action.

There are alternative ways of categorizing the consequence scale, for example:

1. **Negligible** consequences.
2. **Minor** consequences on overall needs, mission, etc.
3. **Major** consequences to overall needs, mission, etc.
4. **Critical** consequences to overall needs, mission, etc.

An alternative way of categorizing the consequence scale is:

1. **Minor** - relatively minor changes to the ecosystem or habitat; it is unlikely that there would be any measurable changes at whole trophic levels outside of natural variations.
2. **Moderate** - some measured change to the ecosystem or habitat, but without being a major change in eco-system or habitat functions. These changes are not disabling; they are not emergencies. These changes may be irritational, but are acceptable, until recovered.
3. **Major** - ecosystem or habitat components are functionally altered significantly. The level of change [due to the incident/risk] is not acceptable to enable one or more societal/environmental objectives to be achieved.
4. **Extreme** - Could lead to total collapse of habitat service, or collapse of an eco-system service (ecosystem processes).

NOTE: *Risks can change categories quickly, and hence, must be monitored.*

Common categories of risk remediation (mitigation) include:

1. **An absence** of what is required:
   A. Omissions (of information).
   B. Unclear (information).
   C. Illogical (information).
   D. In-coherencies (of information).
   E. Weaknesses (of otherwise useful structure).
   F. Inconsistencies (of applications).
2. **Barriers** to understanding and behavior change:
   A. Cultural barriers (social barriers).
   B. Motivational barriers.
   C. Profit and resource acquisition barriers.
   D. Physics (barriers of physical reality).
   E. State-regulatory (barriers of State authority).
3. **Actions** with the potential to de-rail understanding and behavior change:
   A. Actions taken on the part of market encodings, which consciousness requires decoupling from in order to operate community.
   B. Actions taken on the part of State encodings,

which consciousness requires decoupling form in order to operate community.

There are also different societal-level categories of risk, including:

1. People risks:
   A. Sufficiency of people to complete work.
   B. Accidents by people.
   C. Maliciousness of people.
2. Process risks:
   A. Failure to know of important processes.
   B. Failure to execute important processes.
3. Technology risks:
   A. Power loss.
   B. Technical failures.
   C. Replacements (if not developed internally).
4. Habitat risks:
   A. Climactic events.
   B. Market-State boom, busts.

Risk-type questions associated with ongoing fulfillment include, but are not limited to:

1. Does the community have uninterrupted access to their human needs/requirements?
2. At what quality/optimality are the needs being met?
3. Are those needs met in a regenerative manner?
4. Does the community have any unmet needs?
5. What concerns may cause the community's access to their [basic] needs to be interrupted?

A risk is a constraint or uncertain event (condition, state, or shape) that may present a potential problem for a project. A risk constraint is what is known that could go wrong and cause additional problems.

Note here that 'risks' and 'issues' are the same thing, problems. A 'risk' hasn't happened yet, and an 'issue' has happened (or is happening now). An 'issue' is a "risk" with a probability of happening 100% (not 99% as risk itself is categorized). Issues are experienced risks (i.e., "risks in reality"). Risks and issues are sometimes collectively known as "concern coordination" (or, "concern management"). Risks are mitigated and issues re-solved.

A concern will have 1 or more actions associated with it, and some actions will be associated with 1 or more concerns. Concerns and actions require actors (systems or people) and accountability.

> **INSIGHT:** *Trust is essential because it is how you make an accurate assessment of the risk.*

Risks are listed (registered) and analysis (quantitative and qualitative) are conducted on them. Qualitative risk analysis is the process of assessing the probability of identified risks occurring, their potential impact to project objectives and prioritizing risks based on the resultant risk exposure.

Importantly, risks lists (registers) should be maintained and the incidents (risk actualizations) should be monitored for.

The names of risks are typically written as short sentences or sentence fragments. Fragments complete the following types of sentences:

1. "The risk to this project/system is the event that ..." or a variant such as,
2. "The risk to this project is that there will be ..."

Generally, risks also include a short description of the consequences of the risk occurring.

Table 6. *Risk description format.*

Name of the Risk *(what bad could happen?)*			
Type name of risk here			
Description of Risk *(describe that bad thing happening and its consequences; risk scenario)*			
Describe effects of actualization of risk here			
Root Cause	**Could Happen**	**Possible Effects**	**Current / Planned Action**
The root cause analysis (what gives rise to the risk and/or risk exposure?).	Something bad that could happen given a root cause.	Some possible effects if the could happen were to happen.	The current or planned action designed to reduce the risk through termination, transfer, treatment, or toleration.

15.1 Risk categorization

The following are risks commonly associated to all projects. Project difficulty involves a number of variable conditions:

1. **Number of tasks** - more task would increase project/mission difficulty.
2. **Skill variety** - a project requires the integration of some essential abilities, like the ability of information searching and word processing, to complete the project. Skill variety would increase the project/mission difficulty.
3. **Time limit** - Time limit means that there exists a deadline. Tight time limit would increase the mission difficulty.
4. **Resource support** - Resources here mean that all the tools, equipment and solutions could help someone complete their mission. Limited resources support would increase the mission difficulty.
5. **Social dis-alignment and conflict** - conflict means that work is not getting done because there is not a resolution to an issue. The degree to which someone's salience toward an issue of concern will

likely lead to conflict, instead of resolution and/or continued project progress.

From a high-level perspective, risks may be categorized into the following sectors of society:

1. **Political** - as those relating to the political situation facing the authority whether that is global, national, regional, or local. It covers things like election cycles, policy direction, policy creation, policy administration, political re-organisations, political relationships and styles, activism, war and terrorism.
2. **Governing** - as those relating to the governance and decision-making arrangements of the authority. It covers things like the constitution, codes of conduct, leadership, checks and balances, and member-officer relations.
3. **Management and Professional** - as those relating to the need for the "authority" to be professionally fit and capable for a purpose. It covers things like recruitment and retention, succession planning, management style, management systems (e.g. project management, performance management), staffing, administration, morale, capacity, skills, professional judgement, absence management, grievance and disciplinary policies, and employee relations.
4. **Legislative and Regulatory** - as those relating to new and pipeline legislation and the authority's audit and regulatory environment. May also relate to the authority's own legal and regulatory powers.
5. **Competitive** - as those relating to the market situation and the authority's competitors. It covers things like exposure to the market, competitiveness/value for money of services, spotlight seeking (for pathfinders, awards, etc.) and competition with nearby or benchmark organisations.
6. **Reputation** - as those relating to the authority's reputation with government, partners, the media and the public
7. **Citizen** - as those relating to the authority's need to meet changing needs and expectations of its citizens and customers. It covers things like consultation, communication and involvement as well as access, demand and the customer complaints and litigation culture
8. **Economic** - as those relating to the global, national and local economy. It covers things like economic cycles, the economic base, employment and earnings patterns, migration and inflow patterns, house price affordability and availability, regeneration and new development
9. **Social** - as those relating to national and local demographics, residential and social trends. It covers things like age profile, ethnic profile, health trends, crime trends, residential patterns and profile of housing stock, leisure and cultural scene, family profiles, skills base and educational provision and attainment.
10. **Environmental** - as those relating to the physical environment. It covers things like land use, infrastructure, transport, waste, drainage and flooding, erosion, subsidence, landslip, disease, pollution, contamination, seismic activity, air quality, water quality, energy use and efficiency, noise
11. **Partner/Contractual/Supplier** - as those relating to the authority's partnerships, contracts and supplies. It covers local strategic partnerships as well as more straightforward contracts with the private sector and concerns procurement, contract and relationship management, governance, funding, skills, quality and effectiveness.
12. **Technological** - as those relating to the organisation's technological situation and environment. It covers things like strategy, innovation, obsolescence, the nature of systems, support, maintenance, access, security, data protection and reporting.
13. **Financial** - as those relating to the organisation's financial situation and systems. It covers things like adequacy of funding, gearing, financial planning, financial delegations, budgetary control, monitoring and reporting, commitments, cash and treasury management, taxation, pension funds, insurance.
14. **Legal** - as those relating to current legislation and the authority's ability to deal with it. It covers systems of legislation, and the availability of legal advice and support.
15. **Physical** - as those relating to the organisation's physical and people assets.

Societal-level risks can be categorized in the following ways:

1. **Financial cost risks** (market only) - are those associated with the ability of the program to achieve its cost objectives.
   A. Cost risk is a focus area required to maintain integration of the risk assessment process to ensure consistency of the final product.
2. **Authority permission risks** (State and market) - are those associated with the ability of the program to acquire the permissions necessary to achieve its objectives.
   A. Authority risk is a focus area required to maintain the permission of those in authority in order to ensure consistency of the final product.

3. **Schedule risks** - are those associated with the adequacy of the time estimated and allocated for the development, production, and operation of the system/product.
   A. Schedule risk is needed to maintain integration of the risk assessment process to ensure consistency in the final product.
4. **Technical risks** - are those that threaten the evolution of the design, the production, and/or the level of performance necessary to meet the operational requirements.
   A. This type of categorization should be based on the source or root cause.
5. **Program risks** - are those that threatens to terminate, weaken, or change the strategic objectives, direction, or vision.
   A. Driving factors of Program Risk are cost, schedule, and/or performance.
6. **Process risks** - are those that threatens the proper execution of the community processes defined, explained, and envisioned in community standards.
   A. Process risks are assessed in terms of variance from accepted standards and potential consequences of the variance.
7. **Product risks** - are those that threaten the production of any aspect of community, so that it may not be produced on time, within budget, and/or according to specifications (standards).
   A. Product risks are assessed in terms of technical performance measures and observed variances from established specifications (standards).
8. **Functional risks** - are those that affect the community's ability to support specific users or user functions. Typically, functional risks occur where specific community requirements and processes have been overlooked in the implementation.
   A. Function risks are assessed in terms of variance of expected function.

15.2 Risk analysis and action planning

A complete risk analysis involves the following elements:

1. Risk identification:
   A. Is the risk understood.
   B. Is it known what the root cause of the risk is (e.g., external source or threat; or internal source, problem, or initiative).
   C. Is it known where the uncertainty lies in terms of the bad event that could happen.
   D. Is it known what the possible effects could be if the bad event does happen.
   E. Is it known what the likelihood of the risk occurring is.
   F. Is it known what the impact will be if the risk occurs.
2. Acquire a baseline:
   A. Are the critical success factors for development and performance (Read: progress) identified.
   B. Is there understanding (visual clarity) why a risk is at the position it is on the risk map.
   C. Should a risk be re-positioned on the risk map.
3. Existing actions:
   A. What is already in place to mitigate and/or control the risk, and how adequate are these actions.
      1. What are the newest actions that have just been taken to control and/or mitigate the risk, and were they appropriate.
   B. Where existing actions adequate, are the results as expected.
   C. What evidence is there that progress is being made to control/mitigate the risk.
4. New actions:
   A. What future actions may be taken to control and/or mitigate the risk.

15.3 Risk criticality mapping for risk decision prioritization

Risks can be mapped into a matrix that relates a set of risk's impact to the risk's likelihood, thus providing qualitative data to support prioritization of decisioning. The map shows what risks ought to be prioritized. Along one dimension of the matrix lies the impact categories, and along the other is the likelihood categories. To form the mapping:

1. Take each risk in turn and determine how likely the risk is to happen from very low (1) to very high (5). For ease of ranking, determine how likely each risk is to happen.
2. Once the likelihood is determined from 1 to 5, then determine the impact, again from 1 to 5 ranging from negligible to critical. Again for ease of ranking, determine the impact of the possible effects should the risk be actualized.
3. In assessing risk, take the 'worst likely' manifestation of the scenario not the 'most likely'.
4. Note: Only factor current action into the ranking, not planned action.

The risk criticality matrix, when complete, will show which risks require:

1. Immediate action and monitoring.
2. Action, resources, and monitoring are required, but less time critical.
3. Do not require significant attention or resources.

Risks can be categorized according to their potential to do harm, wherein, there is:

1. **Critical** - potential failure of whole system or function [after risk becomes actualized; after incident].
2. **Serious** - serious/high repair required [after risk becomes actualized; after incident].
3. Moderate - moderate repair required [after risk becomes actualized; after incident].
4. **Low** - significant repair not required [after risk becomes actualized; after incident].
5. **Negligible** - insignificant repair required [after risk becomes actualized; after incident].

Risks can be categorized according to their likelihood of occurring, wherein, there is a temporal frequency association:

1. Very high (e.g., 90% within X amount of time).
2. High (e.g., 70% within X amount of time).
3. Medium (e.g., 50% within X amount of time).
4. Low (e.g., 30% within X amount of time).
5. Very low (e.g., 10% within X amount of time).

15.3.1 Action plan decisioning

The key action(s) that form the plan to mitigate risks and solve issues due to risk is to consider what form, or forms, of action is best within the action categories of: Termination, Transfer, Treatment or Toleration. is appropriate to the risk Action plan decisioning around risk typically follows the following flow diagram:

1. Can the risk be avoided?
   A. Should the risk be avoided?
      1. Is it best to terminate the program that gives rise to the risk?
2. Can the risk be transferred?
   A. Should the risk be transferred?
      1. Is it best to transfer the risk to another "entity"?
3. Can the risk be controlled (treated)?
   A. Should the risk be controlled?
      1. Is it best to try and control the risk oneself?
4. Can the risk be lived with (tolerated)?
   A. Should "we" live with the risk?
      1. Is it best to live with the risk?

Common means of controlling for risk include, but may not be limited to (note: internal controls are designed, amongst other things, to reduce risk)

1. **Administrative controls (a.k.a., punitive controls)** – designed to ensure compliance with policies, standards, plans, rules, regulations and procedures.
2. **Preventative controls** – designed to prevent the bad event.
3. **Detective controls** – designed to identify errors, violations, or irregularities in actions/transactions already taken/processed.

15.4 Project uncertainty

> **INSIGHT:** *The paradigm of understanding that humanity creates its living algorithms from should not be deeply flawed.*

All projects exist in an uncertain environment (otherwise there would be no need, no human requirement, for a project). An 'uncertain' environment is a 'probable' (similar to 'likelihood') environment. An emergent networked [eco-societal] system is, by assumption, an uncertain environment, with the conditions of risk and constraint (on all integrations, decisions, and actions). Risk is a measure of the probability that a negative outcome will occur. Risks represent potential dis-alignment from trajectory. Risk coordination identifies the risks to safety, performance and the project (e.g., overruns, schedule delays, etc.).

Every project involves some degree of uncertainty. Before a project is started, a plan is prepared based on certain assumptions and estimates. Assumptions are documented because they will influence the development of the project's resource selection, schedule, and work scope. A project is based on a unique set of tasks and estimates of how long each task should take, various resources, assumptions about the availability and capability of those resources, and estimates of the inputs and total effects (true costs) associated with the resources and their particular flow through the system. This combination of assumptions and estimates causes a degree of uncertainty that the project objective will be completely accomplished and/or accomplished within a specified time-frame. For example, the project scope may be accomplished by the target date, but the final resource requirements may be much higher than anticipated, because of low initial estimates for the necessity of certain resources. As the project proceeds, some of the assumptions will be refined or replaced with factual information.

Someone may not absolutely know the outcomes of one's own actions, but by thinking probabilistically, can perceive a distribution over outcomes. The expected value of an action can then be computed from utility (human requirement fulfillment functions) and probability integration through computation. This cognition "entangles" the agents' betterness relations (i.e., what relationship is the better choice?) as well as the agents beliefs/values about possible outcomes.

> **INSIGHT:** *You have to accept some risk, nothing is ever going to be 100% risk free of uncertainty.*

15.5 Real problems

CLARIFICATION: *A risk is an uncertainty. When this uncertainty becomes certainty, that is, when the risk occurs, then there is a real-world problem.*

The technical procedures required in formulating [environmental] problems should, but sometimes do not, begin with the question: "Does the problem really exist?" Problems in the real-world, the designed environment, are often assumed without detailed systematic analysis, leading to problem definitions that target the wrong system, or target a system without a problem.

For a human societal system, an environmental problem exists if, and only if, a malfunction can be detected between the designed environment and the system of human behaviors. Due to the nature of human-environmental dependency, environmental problems must not only be detected, but they must be resolved, so that humans are mutually fulfilled.

Because this is a societal system development project, anything that has the potential to impact the next iteration of the societal system is a potential risk.

16 [Project] Risk coordination

A.k.a., Risk control, negative coordination and control, risk management, plan risk management, disaster recovery, business continuity, disaster recovery.

Risk coordination is an organizational (e.g., business, societal) process that all projects must undergo to protect their objectives from threats and facilitate actualization of opportunities to ease the efficiency and/or effectiveness of achieving objectives. Plan risk coordination is identical in naming for both project coordination (project management) and systems engineering. Both information sets define the risk method/strategy for determining how to conduct risk activities.

There are engineering technical risks, as well as project coordination ("oversight of the technical") risks [as all disciplines are critical to the participation in risks operations & maintenance.

The idea of 'risk' exists in a relationship between project management, systems engineering, and a dynamic environment:

1. The INCOSE Systems Engineering Handbook (SEHBK) suggests Analyze Risks includes "identification and definition of risk situations", which equates to the PMBoK Identify Risks.
2. The section of the Project Management Body of Knowledge (PMBoK) entitled "Identify Risks and Plan Risk Responses" equate to the SEHBK, "Define a treatment scheme and resources for each risk…", and is included in the SEHBK Analyze Risks activities.
3. The SEHBK adds an iteration activity: Evaluate the Risk Management Process, which should be true of all processes and activities.

To plan for risks requires to following procedure:

1. Identify what could go wrong (i.e., get the list of risks).
2. Assign likelihood and impact to each risk.
3. Develop mitigation techniques for risks.
4. Quantify impact of active mitigation techniques and responses.
5. Qualify mitigation solution.

Risks become issues to respond to once they actually occur. A risk list (risk register):

1. Records identified risks with a reference number for each risk.
   A. Describes the risk
   B. Links to a team or working group accountability for each risk

2. Identifies the likely severity of impact of each risk (likelihood).
3. Identifies the probability of occurrence of each risk (probability).
4. Identifies mitigation activities.
   A. Identify requirements for each mitigation activity.
   B. Identify probability of occurrence with each mitigation activity.
5. Identifies response procedures should the risk occur.
   A. Identify requirements for each response activity.
   B. Identify estimate of recoverability after each active response activity.

A complete risk plan includes mitigation activities and response procedures for each risk.

16.1 Plan for risk

A risk is uncertainty that affects objectives. In general, risk includes both opportunities and threats. The PMBok (2013) definition of risk makes this most clear with the words, "positive or negative effect on an objective". In common parlance, however, the term risk is generally intended to mean a negatively impactful probability. Uncertainties can affect the achievement of a project's objectives either positively or negatively. Often, the term, "risk event" is applied to both uncertainties that could hinder the project (threats, negative impacts) and uncertainties that could help the project (opportunities, positive impacts). A risk is an unplanned event that could result in harm or benefit; what unplanned event could happen that would result in harm or benefit? Risk involves future events/things that may not happen, but if they do happen, they would effect an objective. Risks matter because the effect the objectives. Risk could be viewed as uncertainty on the achievement of objectives. Risk-taking is the process of accepting risk.

Planning for risk involves thinking and acting to reduce the likelihood of harm preventative and in the case a risk incident occurs. One way to reduce risk is to increase the margin of safety. For example, having a store of some product provides a margin of safety in case the production of that product fails for some reason.

There are two core dimensions to risk:

1. Uncertainty.
   A. In a project, this is called **probability**.
2. Effect on objectives.
   A. In a project, this is called **impact (consequence)**.

Risks are uncertain 'events' or 'conditions'. Risk connects uncertainty with objectives. Uncertainty must always be connected with objectives in order to find the risks. Risk does not mean the same thing as uncertainty. All risks are uncertain, but not all uncertainties are risks. There are some uncertainties that are not risks, such that not every uncertainty in the world will be added to a risk list (or risk register). Risk is a subset of uncertainty that someone (or some population) deem of sufficient importance that they must take preparedness or mitigatory action on. More simplistically, risk is uncertainty that matters, and that likely some action may or will need to be taken upon, often, to prevent a negative impact (result or response).

In any practical dynamic environment, risks may be identified and added to a risks list, but risks are also emergent such that new risks may occur and old risks may no longer be risks. Knowable risks are exposed and listed. Risks are negative deviations from expected; wherein, an effect is a deviation from the expected - positive and/or negative. (ISO 31000:2009)

ISO 31000:2009 defines risk as:

- *Risk is the effect of uncertainty on objectives.*

Association for Project Management (APM, UK) Body of Knowledge, 2012 defines risk as:

- *Risk is an uncertain event or set of circumstances that, should it occur, will have an effect on achievement of objectives. [risk is uncertainty that matters]*

Project Management (PMBoK) Body of Knowledge, 2013, defines risk as:

- *An uncertain event or condition that, if it occurs, has positive or negative effect an objective.*

Significant risk (as, risk to objectives) determination questions include:

1. Which objective would be affected if this thing happened?
2. How uncertain is it?
3. How much does it matter?
4. All three combined determine how significant a risk is.

Fundamentally, there are two types of risk-based impacts, as assessed against objectives, that matter (and should be identified and addressed):

1. **Uncertainties that could hurt** the project (i.e., negative risk, danger).
   A. Uncertain changes or events that could harm; threats. Bad risks.
      1. In navigation, look out for, avoid, prevent, and protect against traps. What could cause us to deviate from a track or course?
2. **Uncertainties that could help** the project (i.e., positive risk, opportunity, favorable outcome).
   A. Uncertain changes or events that could be of

benefit; opportunities. Good risks.

1. In navigation, look out for, seek, and proactively make happen efficiencies and opportunities. What could help us stay on track or on course?

There are events that could happen that could be good, and there are events that happen that could be bad, and both need to be proactively identified and addressed. Events that could hurt need to be prepared for and mitigated against. Simultaneously, events that could help need to be identified and action taken to make them happen. Note that this is equally true in the personal lives of humans as it is at the societal level.

When designing systems, there are three principal design objectives that account for risk:

1. The potential to negate optimal re-solution of the design's requirements (i.e., the design requirements of the human fulfillment system).
2. The potential to hurt a human or fulfillment system.
3. The potential to cause an accident, and thus, unnecessary problems in, a human fulfillment system.

NOTE: *For every assumption, there is a corresponding constraint (i.e., probability for a problem). Similarly, for every lack of definition in an argument there is the probable creation of a space for additional error.*

16.1.1 The composition of an risk entry

A.k.a., The statement of a risk, risk statement.

A risk statement is necessarily a cause and effect statement. To develop a negative risks list, identify what might go wrong and cause harm and/or exposure to danger. A risk entry is most useful when it is contained within a structured description that separates cause, risk, and effect.

Risk can be described in three stages (Read: salient categories of meaning in relation to the achievement of a goal):

As a result of <existing condition>, uncertain event> may occur, which would lead to <affect condition> on objectives.

Simplified view of the three stages:

1. As a result of <some cause>, then
   A. a <risk> may occur, which would
      1. <affect> an objective.

16.1.2 Semantic temporality

In the English language, there are words that can be used in communication to identify the different stages or parts of a risk entry (linguistics):

1. Definite words to describe facts (to describe the **present condition; existing condition**).
   A. Is, do, has, has not ...
2. Uncertain words to describe the risk (to describe the **uncertain future; uncertain event**).
   A. May, might, possibly, ...
3. Conditional words that say, this would follow if the risk occurred (to describe the **conditional future; effect**).
   A. Would, could ...

16.1.3 The structure of a risk

A risk-based information set contains information on:

1. Risks have an individual basis:
   A. Their likelihood of occurrence.
   B. Their likelihood of impact (on all objectives).

16.1.4 Population risk types: Personal and social risk

One challenge faced by any society is when one segment of the population does not experience a problem that another segment does experience. In more individualistic societies, when one segment of the population does not experience a problem that another segment does experience, potentially, the segment of the population that does not experience the problem will not perceive social risk, and individuals therein are likely to govern their behavior only based on what they perceive their personal risk to be (i.e., they will only perceive personal risk and ignore social risk). Alternatively, individuals in a holistic society (and not individualistic society) think in terms of social risk as well as personal risk. For instance, a younger individual in an individualistic society could be generally healthy and not concerned about their personal risk after acquiring symptoms of a viral infection. This person may feel well enough to go to their job where there co-workers are present. This person may shake hands with an older colleague who has a chronic medical condition, who may become infected by the virus carried by the young co-worker who felt well enough to go to work. In such a case, the young co-worker could be responsible for that colleagues death.

In a healthy and cooperative society, it is wise for all individuals to think about their responsibility to each other when deciding their behavior. Society at large should not be thought about in terms of individuals' personal risk; instead, individuals should act collectively in a cooperative manner to reduce societal-level risks.

16.1.5 Negative deviation: Negative risks

A.k.a., Threats, detriments, losses, negatives.

In its negative context, a risk is a situation and probability involving exposure to danger (Read: harm, injury, loss, suffering, etc.), or any other negative occurrence that is caused by external or internal environment, and that may be avoided through pre-emptive action (Read: through controls on preparedness, operations, and responses). For any intentionally living system, in an uncertain environment, there is the conception of risk. In the real world, for a social populations, there are a multitude of risks. To navigate safely together, risks must be identified, prepared for, and mitigated against (i.e., protected against the danger or reduce/eliminate the danger). A negative risk is the likelihood that a loss will occur.

In the context of negative impacts, risks are potential events that could happen during the course of a project, that if they happened, they could (note: these are effects, not risks):

1. Kill or injure.
2. Lose resources, assets, or access.
3. Waste time.
4. Waste effort and energy.
5. Damage reputation.
6. Damage natural ecological cycles.
7. Harm performance.
8. Waste money *(market-State only).*

There are many types of negative risks, including but not limited to:

1. Human life risks.
2. Project risks.
3. Personnel risks.
4. Operational risks.
5. Technical risks.
6. Social risks.
7. Environmental risks.
8. Ecological risks.
9. Financial/business risks *(market-State only).*

16.1.6 Not a risk (non-risk)

Items that are certain and do not belong in a risk list include, because they are certain (and not uncertain):

1. Problems - a problem has been identified (and there are solutions to resolve the problem).
2. Issues - an issue has been acknowledge (and a process is engaged to resolve it). Issues require resolution. Issues have occurred or will imminently occur. A negative event can turn into an issue.
3. Constraints - a known limitation placed on a project/system.
4. Requirements - a known expectation from a project/system.

Risks are neither causes nor effects. However, it is easy to confuse risk with non-risk, especially cause or effect. There is a real-world, dynamic system in which risk occurs:

1. Cause (fact) - causes are not risks because they are occurring now. Causes are facts, issues, problems. Causes are not risks, because they are not uncertain.
   A. Something true today.
2. Risk (uncertainty)
   A. Something that may, or may not, happen
3. Effect (possible result) - If the risk has occurred, then it is an effect.
   A. Why something matters to the objective.

It is most useful when risk descriptions have a description of not only the risk, but also cause and effect.

16.2 [Plan] Risk coordination process

A.k.a. The risk management process.

The risk process may be simplified to:

1. Identify objectives.
2. Identify uncertainties that matter to objectives.
   A. Include in the identification threats, negative uncertainties.
   B. Include in the identification opportunities, positive uncertainties.
3. Prioritize the risks by asking, How uncertain? How much would it matter?
   A. What are the worst threats?
   B. What are the best opportunities?
4. Identify (or construct) responses appropriate to each risk.
   A. What can be done to stop threats, or continue and recover if threat occurs?
   B. What can be done to cause an opportunity to be actualized.
5. Execute the response.
   A. Preparedness and pro-active action for opportunities.
   B. Preparedness, mitigation, and operational actions for threat event.
6. Risk control.
   A. Monitor the results of all actions.
   B. Review for new risks, and repeat.

The most significant risk process questions that can be used for any project (or even any decision) include:

1. What are we trying to achieve?
   A. Set objectives.
2. What could affect us achieving it?
   A. Identifying risks.
3. Which are the most important risks on the list?

   A. Assess and prioritize risks.
4. What can be done about the risks?
   A. Planning responses.
5. When should it be done?
   A. Schedule responses and update cycle.
6. Did it work, and what has changed?

16.2.1 Organizational planning for risks

Systems engineering coordinates ("manages") technical risks within a project[-based structure].

At the project-level, the principal risk is (managing organizational risk):

- Delivering a system (Read: new system state) that does not meet organizational, orientational standards.

A the engineering-level, the principal risk is (managing technical risk):

- Delivering a system that does, or does not, meet user requirements.

In practice, a risk coordination and control system (team or working group) should account for threats and opportunities together in a single unified process, because they are both uncertainties that matter. Both threats and opportunities are types of events that may or may not happen that are likely to impact the objective. Both threats and opportunities can both be accounted for and pro-actively acted upon.

16.2.2 The risk plan (information set)

What might go wrong with the plan, and how to limit that risk with contingency planning:

1. Description of problem (risk).
2. Probability and impact of risk.
3. Workaround of problem.
4. Scope of contingency.

System safety is the accounting for observations that accidents can result "from dysfunctional interactions among system components" (e.g., bottlenecking to incident, or overshooting carrying capacity).

System 'safety' is influenced not only by the reliability and failure behavior of various subsystems and components, but also by the nature of interactions between these components, as well as their interactions with external factors (i.e., environmental conditions).

1. Safety includes human-caused incidents.
2. Safety includes environmentally-caused incidents.

16.2.3 Risk resolution coordination

The coordinated resolution of probable risk entails the analysis of risk as an information process, and the mitigation of risk as a [engineered] construction process.

Risk coordination control (risk management) refers to systematically addressing risk throughout the life cycle of a system, product, or service. Project risk coordination (management) includes the processes of conducting risk coordination planning, identification, analysis, response planning, and controlling risk on a project.

16.2.3.1 Taking proactive action as opposed to being forced to rely on reaction

In common definition:

1. Pro-active action refers to a complex of inter-actions, including but not limited to planning and monitoring in order to reduce the probability of negative consequences (i.e., reduce the likelihood or results being mis-aligned with objectives).
2. Reactive action refers to responding to a consequence without planning.

For example, the athlete gets hurt before they need "therapy", versus providing "therapy" during the athletics life cycle so they are less likely to get hurt in the future.

In a world where supermarkets are food carnivals (falsely flavored and highly palatable foods and food-like substances), filled with biologically "addictive" combinations, then the socio-economic reality is that decisions and behavior are not solved solely by personal choice, but they also necessitate as part of the solution, modifying the food environment (i.e., fixing the food environment so that it doesn't seemingly "naturally" in close proximity and access these foods, and move through environments that don't "naturally" drive individual organisms to).

16.2.3.2 Indicators and pro-active versus reactive action

Lagging indicators are used to evaluate current conditions. In order to act pro-actively, it is necessary to explore future projections in order to better guide an organization toward greater success at, or achievement of, a goal. Leading indicators give an organization the [informed] ability to think and act pro-actively, instead of reactively, which can reduce the time required to meet the goal, and in the market "save" money.

16.3 [Plan] Organizational exposure

Tracking organizational exposure through an assessment tool helps in understanding a project's exposure to a risk. The following assessment aims to support decisioning and is a definitional tool, no an explanatory one.

Table 7. Execution > Risk*: Exposure assessment including statements about aspects that may be directly or indirectly impacted by a risk.*

#	Organizational exposure	*Disagree (0); Neutral (2); Agree (4); Strongly Agree (5)*
1	The country and regional exposure to the understandings and operation of a community-type society.	
2	The organization operates in a country or region that is not of the societal type, community.	
3	The organization operates a societal interface.	
4	The organization is facing challenges regarding access to resources.	
5	The organization contributors are pessimist about the impact of an event.	
6	The organization is based or has a strong presence in production of required resources.	
7	The event will cause an unrecoverable loss.	
8	The long-term organization can be directly and negatively impacted by the event in socio-technical (micro- and macro-economic) factors.	
9	The work is located in an environment that is not of the societal type, community.	
10	The project or initiative has a large number of people working in the same location.	
11	The project or initiative has a strong need for human interaction.	
12	There will be a large negative impact if the work is reduced or ceased.	
13	Disruption to the supply chain will have a severe impact on the development of the work.	
14	The initiative or project is heavily dependent on an external societal supply chain.	
15	A member of the team is incapacitated.	
-	**If combined numerical result is low, then there is low risk; if middle, then moderate risk; if high, then high risk:** [1] If the assessed risk is low, then continue monitoring the situation and re-evaluate if results change. [2] If the risk is moderate, take moderate action. [3] If the risk is high, take rapid action.	**Combined numerical result from calculating:** *count x weight* **(of each statement)**

In order to reduce the likelihood of harm:

1. A harm reduction approach acknowledges that laws/protocols may be broken, and this can be tolerated in favor or reducing risk and increasing safety.
2. Risk review board (a.k.a., ethics review board, ERB) - when is an action decidedly available (i.e., "OK") that risks psychological and/or physical harm [in the name of science and social safety).

16.4 [Plan] Risk mitigation and remediation

Risk mitigation planning is the process of developing options and actions to enhance opportunities and reduce threats to project objectives. Risk mitigation implementation is the process of executing risk mitigation actions. Risk mitigation progress monitoring includes:

1. Tracking identified risks.
2. Identifying new risks.
3. Evaluating risk process effectiveness throughout the project.

16.4.1 [Mitigation] Narrative role model advocacy

A.k.a., Behavioral economics.

Narrative role model advocacy is the use of a storyline in media (e.g., radio, television, film) to affect change across an entire society. A storyline, for example, may have middle of the road characters designed to represent segments of the audience and be aspirational for them, but also be very similar to them. Those characters sort out conflicting advice from the positive and negative characters one finds in all melodrama, and over many episodes they gradually evolve into positive role models for the audience, and they show the audience the benefits of the new behavior and they deal with the pushback that one gets when they try anything unusual or innovative in any society's, so they show the audience how to deal with that pushback. The storyline characters ultimately become outspoken advocates for that new behavior, thus role modeling behavioral change and advocacy for the audience population. It is possible for researchers to actually to measure changes among the audience members in self-reported interpersonal communication about the issues being addressed in the storyline. Common themes in storylines include violence against one another, human issues, and family planning. The fundamental goal is to change the perception of what is normal and/or possible.

Keltner and Piff (2010, 2014) in laboratory research have found that small psychological interventions, small changes to peoples values, small nudges in certain directions can restore levels of egalitarianism and empathy. For instance, reminding people of the benefits of cooperation or the advantages of community caused wealthier individuals to be just as egalitarian as poor people.

16.4.2 [Mitigation] Understanding advocacy

Achieving full interoperability of socio-technical data is both complex and fraught with pitfalls. Users of environmental data and may be uncomfortable with geodesy, geometric transforms, dynamics modeling,

and logical reasoning. It is strongly thought that an adequate education is a necessary prerequisite to success in that endeavour. There has been a fair amount of misunderstanding when practitioners from different disciplines talk to each other about location, condition, and decision information.

There are also unspoken, and all but forgotten, assumptions made within specific disciplines that are opaque to non-specialists, and either result in miscommunication or are simply no longer appropriate assumptions to make. These problems are not unique to specialists -- they are rife within the general systems engineering and resource-based economy (RBE) organizations as well.

Given these considerations, the application of interoperability, necessarily includes significant didactic material, and generally covers four major areas, as follows (which may be seen as objectives for a learning contributor):

1. **Concept development:** This includes the reference model (RM), the scope of the reference model (RM), as well as the design criteria. It also includes the development of the concept of "pure" coordinate systems and their associated transformations from basic concepts in a database to solid and analytic geometry. Subsequently, isometric ("real world") geometry and coordinate systems are developed and extended to define the basic isometric socio-technical reference frames. Concepts associated with directions (vectors), and the corresponding orientation representations, are defined. The complexities of real-world terrain surfaces (as opposed to mathematical "smooth" approximations) are addressed.
2. **Conceptual reference frame specifications and formulations:** This covers the complete specification of the conceptual reference model (CRM) and each of the included conceptual reference frameworks (CRFs).
3. **Spatial Reference Frame Specifications and Formulations:** This covers the complete specification of the spatial reference model (SRM) and each of the included spatial reference frameworks (SRFs). Error specification and algorithmic development: This addresses how to define error specifications in reasoning and visualization (2D and 3D), and the development of efficient and accurate coordinate operations algorithms among the reference frameworks (RFs) included in the unified reference model (RM). Also included is a discussion of transitivity and chaining when converting between reference frameworks (RFs), which may use a sequence of operation steps rather than a single optimized direct conversion.
4. **Implementation, testing and application:** This involves the reduction of the developed algorithms to efficient, accurate, portable implementations that maintain the stated operation accuracies and performance. The methods used to test and verify the implementations are developed, and the results of extensive testing, are presented and reviewed. The information and material interface specification is defined, and guidelines for its use are documented.

16.5 [Plan] Risk response

NOTE: *Ignorance exist in contrast to planning for risks.*

When mitigation isn't successful, then response occurs. In order to most effectively respond to risks, a series of planned questions must be asked and answered:

1. What should be done, based on?
   A. Type and nature of risk.
   B. Controllability.
   C. Impact severity.
   D. Resource availability.
   E. Efficiency/cost-effectiveness.
2. Who, when, where, and with what tools, should it be done.

The most common categories of response to negative risks (i.e., threats) include:

1. Eliminate uncertainty - eliminate risk, kill risk, avoid risk. Note: by avoiding one risk, the solution may lead to the exposure to other risks.
   A. An avoid strategy.
2. Reduce uncertainty - reduce risk to acceptable/ controllable levels; reduce the impact or exposure of the risk; mitigate the risk. Reduce risks by reducing probability and/or impact.
   A. A reduce strategy.
3. Transfer responsibility, liability, ownership - give risk to another entity. Have an outside authority handle the risk for you. (e.g., use anti-virus software on a software operating system, use insurance use insurance; note that the presence of this type of risk is indicative of poor/non-optimal design). The asset is still being protected, it's just that you are not the one doing it.
   A. A transfer strategy.
4. Accept residual risk - accept the risk and control it as best as possible.
   A. An accept strategy.
5. Ignore risk - do nothing.
   A. An ignoring strategy.

The most common categories of response to positive

risks (i.e., opportunities) include:

1. Cause the opportunity to happen. Exploit some connection.
2. Share responsibility for making some event happen with another or others.
3. Enhance some connection to make the event more likely to occur.

Table 8. Execution > Risk*: Methodical responses for the presence of risk.*

Threat	Generic Strategy	Opportunity
Avoid (eliminate)	Eliminate uncertainty	Cause (exploit)
Transfer	Allocate ownership	Share
Reduce	Modify exposure	Enhance
Accept	Include in baseline	-

16.5.1 Risk coordination process elements

A.k.a., Risk control elements, risk coordination and control elements.

The phases of the risk coordination cycle include four main elements:

1. Risk identification - identify all potential risks.
   A. Deliver a list of risks.
2. Risk assessment - prioritize the likelihood of this risk occurring and the severity/damage impact if it occurs.
   A. Deliver a risk assessment matrix.
3. Risk control.
   A. Risk mitigation - Deliver a plan, procedure, technique, tool, or process to mitigate threat.
   B. Risk accentuation - Deliver a plan, procedure, technique, tool, or process to actualize opportunity.
4. Risk monitoring and control.

Alternatively, the phases of the risk coordination cycle could be viewed as:

1. Identify probable risks.
2. Determine probability of each risk.
3. Evaluate potential impact of each risk.
4. Develop controls and plans as responses to each risk.
5. Document responses to each risk.
6. Act on next steps (i.e., next tasks) for each risk.

The common risk coordination elements include (note that risk-coordination is a sub-type of issue-coordination, see sub-bullets):

1. **Risk planning** - planning for the avoidance of danger, or an avoidable reduction in fulfillment.
   A. Risk-type issue planning.
2. **Risk identification**
   A. Risk-type issue identification.
3. **Risk analysis** (clarifying, categorizing, and prioritizing risks, and developing controls for risks)
   A. Risk-type issue analysis.
4. **Risk mitigation** (design, select, and applying/ implement controls to mitigate risks)
   A. Risk-type issue mitigation.
5. **Risk monitoring** (assess and monitor active controls)
   A. Risk-type issue monitoring.

At a societal-level, risk coordination feeds into a societies Effectiveness Inquiry [decision system] process.

16.6 [Plan] Continuous risk analysis, coordination, and control

QUESTION: *Is it possible to engineer a system that does not posses the risk? Is it possible to analyze and design a system that is highly less likely to express the risk?*

Continuous risk analysis, coordination and control is a project engineering category with processes, methods, and tools for predicting risk-related problems and resolving them such that the project is safe, effective, and efficient.

Any useful risk inquiry identifies, what are the social mechanisms that are driving people to maintain these fixed and limiting behaviors and/or beliefs.

Risk mitigation processes include:

1. Assessing continuously what could go wrong (risks).
2. Determining the significant prioritization of risks.
3. Acting to resolve the risks:
   A. Identifying uncertainties.
   B. Identifying assumptions.
   C. Identifying problems and inquiries.
   D. Resolving the probability risk space.
   E. Resolving the decision space.
   F. Changing the system to have engineered a risk out of the system.

**All processes may occur in parallel and/or series.*

Risk analysis and mitigation involves the analysis, design, and operation of systems without the risk [to humanity and ecology]:

1. The identification of [probable] risks within:
   A. The current system.
   B. A new system state change.
2. The resolution of [probable] risks within:

A. The current system.
B. A new system state change.

In community, to minimize reliance on error-prone and time-intensive human or procedural controls, the primary means of risk mitigation involves the designing of risk out of a system (e.g., fail-safe redundancy, fault tolerance, load margins, inherent reliability, and test verification). In practice, risk reduction depends on advance knowledge of environmental conditions, performance of engineered products/systems, accurate testing, and human [response] capabilities.

Materialized systems throughout the community's habitat service system have different levels of fault tolerance. For example, locations where human safety is a critical function, normal design criteria require two-fault tolerance levels. All critical systems essential for human and ecological safety (survival) shall be designed to be two-fault tolerant [at least]. When this is not practical, systems shall be designed so that no single failure shall cause loss of the Team (or city). This requirement, as a component of [operational] maintenance, can be considered as a third level of fault tolerance (i.e., of redundancy). In community, however, functional roles are they are unified under open source engineering.

The risk analysis and mitigation sub-processes categories are (function/operation):

1. **Identify** - search for and locate risks before they become problems.
2. **Analyze** - transform risk data into decisioning information.
   A. Evaluate impact, probability, and timeframe, classify risks, and prioritize risks.
3. **Plan** - Translate risk information into decisions and mitigation actions (both present and future) and implement those actions.
4. **Track** - Monitor risk indicators and mitigation actions.
5. **Control** - Correct for deviations from the risk mitigation plans.
6. **Communicate** - Provide information and feedback internal and external to the project on the risk activities, current risks, and emerging risks. Note: Communication happens throughout all the functions of risk mitigation.

Continuous risk analysis requires answers to the following questions:

1. What proximity is required for this risk to apply?
2. How localized are the effects posed by this risk?
3. What is the recovery time if the risk was detected?
4. What are the recovery and restoration requirements if the risk is detected?
5. Impact - How serious an impact?
6. Prior - Is there evidence of this risk prior?

16.6.1 Identify

The principles applicable during the Identify function are:

1. Risks are identified as part of a continuous process, not a one-time only activity at the start of the project.
2. Risk identification must be open source to sufficiently bring forward new risks and to look beyond immediate problems.
3. Although individual contributions play a role in risk management, teamwork improves the identification of new risks by allowing individuals to combine their efforts, knowledge and understandings.

16.6.2 Analyze (Assessment)

The principles applicable during the Analyze function are:

1. Conditions and priorities often change on a project and can affect the important risks to a project—risk analysis must be a continuous process.
2. Analysis requires open communication so that prioritization and evaluation is accomplished using all known information (**safety protocol, open source protocol**).
3. A probabilistic-oriented view enables teams to consider long-range impacts of risks.
4. A global perspective and a shared societal vision allows an analysis of risks to account for the overall societal system, human needs and goals.

16.6.3 Plan

The principles applicable during the Plan function are:

1. Planning risks is a continuous process of determining what to do with new risks as they are identified, to enable efficient use of resources.
2. Integrated coordination is needed to ensure mitigation actions do not conflict with project or team plans and goals.
3. A shared product vision and global perspective are needed to create mitigation actions that ultimately benefit humankind and the ecology.
4. The focus of risk planning is to be probabilistic, to efficiently prevent risks from becoming problems.
5. Teamwork and open communication enhance the planning process by increasing the amount of knowledge and expertise that can be applied to the development of mitigation actions.

16.6.3.1 The best plan, a fail-safe plan

Fail-safe and **fail-secure** (as a task for safe systems engineering and redundancy planning) means that in the event of failure, the system responds in a way that will cause no harm, or at least a minimum of harm, to other systems or danger to individuals. Fail-safe means that a device will not endanger lives or other systems when it fails. A system's being "fail-safe" means not that failure is impossible/improbable, but rather that the system's design prevents or mitigates unsafe consequences of the system's failure.

16.6.4 Track

The principles applicable during the Track function are:

1. Open communication about a risk's status stimulates the project and risk management processes.
2. Tracking is a continuous process—current information about a risk's status is conveyed periodically to the rest of the project.
3. When project personnel review tracking data with a forward-looking view and a global perspective, they can interpret the data to reveal adverse trends and potential risks.
4. Integrated management combines risk tracking with routine project monitoring processes, creating a synergy that better predicts and identifies new issues.

16.6.5 Control

The principles applicable during the Control function are:

1. Open communication is essential for effective feedback and decisioning, a critical aspect of Control.
2. Risk control is also enhanced through integrated coordination—combining it with routine project coordination activities enables comprehensive project decisioning.
3. Shared project vision and a global perspective support control decisions that are effective for the long-term success of the project and [societal] organization.

Scholarly references (non-cited)

- Lagerholm, M. (1998). *Resource Allocation with Potts Mean Field Neural Network Techniques*. Lund University: Department of Theoretical Physics. Thesis for the degree of Doctor of Philosophy. https://pdfs.semanticscholar.org/fb5a/397e433dd16c327a5e14f5699fb180339dc1.pdf
- Lewis, B., Deatrick, J., Johnson, H. (2016). *Project Planning. Region 4. U.S. Environmental Protection Agency Science and Ecosystem Support Division*. Athens, Georgia. SEDPROC-0160R5 https://www.epa.gov/sites/production/files/2016-04/documents/project_planning016_af.r5.pdf

Scholarly references (non-cited)

- Framinan, J.M., Leisten, R., Garcia, R.R. (2017). *Manufacturing Scheduling Systems - 2014*. Springer, London. https://doi.org/10.1007/978-1-4471-6272-8
- Fanchiang, C. (2017). *A Quantitative Human Spacecraft Design Evaluation Model for Assessing Crew Accommodation and Utilization. Aerospace Engineering Sciences Graduate Thesis & Dissertation. Aerospace Engineering Sciences*. University of Colorado, Boulder. https://scholar.colorado.edu/cgi/viewcontent.cgi?article=1160&context=asen_gradetds

Book references (non-cited)

- Belzer, J. Holzman, A.G., Kent, A. (1979). *Computer Science and Technology, Vol. 13, Reliability Theory to USSR, Computing in.* Marcel Dekker, Inc. New York and Basel.
- Daylio, R. (2017). *Principles: Life and Work*. Simon & Schuster. https://www.principles.com/principles-in-action/
- Dorofee, A.J., Walker, J.A., et al. (1996). *Continuous Risk Management Guidebook*. Carnegie Melon University. SEI Joint Program Office. http://www.jodypaul.com/SWE/ContinuousRiskManagement.pdf

Online references (non-cited)

- Brautigam, B. (2017). Living Machines: Design Paradigms for Self-Aware Machines. Medium. https://medium.com/interactive-mind/living-machines-design-paradigms-for-self-aware-machines-d56ac2ac9715
- *Integrated Project Deliver: A Guide*. Ver. 1. (2007). The American Institute of Architects. http://info.aia.org/siteobjects/files/ipd_guide_2007.pdf
- *ISO 9001:2015 Requirements for a Quality Management System*. (2015). https://the9000store.com/iso-9001-2015-requirements/
- *State of Michigan Project Management Methodology.* Michigan Department of Technology, Management & Budget. 2014, May, Ver. 4.0. https://www.michigan.gov/documents/suite/SOM_PMM_Manual_456390_7.pdf
- *Project Management Fact Sheet: Language Matters*. (2017). Department of Premier and Cabinet of Tasmania. Version 1.2, November 2008. https://www.dpac.tas.gov.au/__data/assets/pdf_file/0028/108946/Language_Matters_Fact_Sheet.pdf
- Winter, D. (2012). *Project Stakeholders Gas Plant Kazakhstan*. http://www.order-efficiency.com/demo/stakeholders/LinkedDocuments/Project%20Stakeholders%20Gas%20Plant%20Kazakhstan%20[example].pdf

TABLES

Table 9. Project Approach > Coordination*: Project coordination and control tools. These essential tools represent the source of information and thought processes that are needed to effectively plan and execute a project.*

Tool	Description	Value	Application
Project charter	Initializes project	Provides integration of project into society	Projects interface
Project definition document	Defines project purpose, objectives, deliverables, completion criteria, and scope of work to be completed, explains project type	Provides boundaries and communicates understanding	Project interface
Requirements	Defines the specifications for the product/output of the project	Provides tracing of actionable information	Requirements interface
Project schedule	Shows all work efforts, properly estimated, with logical dependencies assigned to responsible resources scheduled in a calendar	Provides for coordination of the execution of activities with objects in time	Schedule interface
Status reports	Periodic or continuous reviews of actual performance versus expected performance	Provides feedback to allow for timely and appropriate identification of performance variances	Control interface
Key event chart (Milestone chart)	A summary of the detailed project schedule showing progress against key events in time	Provides a high-level project progress report on one page	Control interface
Project organization chart	Shows all project associated individuals and the working relationships among them	Provides a source for identifying the organizational structures, dynamics, and project roles	Control interface
Responsibility matrix	Defines all roles and indicates what responsibilities each role has	Provides a source of coordinated expectations, and tool for establishing and accountability	Control interface
Communication plan	Defines the how, what, when, and who regarding the flow of project information	Provides a tool for effective communication among working [team] members	Communications interface
Logistical coordination plan	Lists how project resources and humans will be acquired, when they are needed, how much are needed, and how long they will be needed	Provides for scheduling	Resources interface
Quality assurance plan	Defines the approaches and methods that will be used to resolve the quality levels of project processes and results	Provides a tool for reducing uncertainty the results of project execution	Reasoning interface
Risk coordination plan	List each identified risk and the planned response procedure for each	Provides for the communication about potential issues in advance, is a proactive measure to reduce impact to a project	Risks interface
Project plan	Formal, programmed data structure [document] used to coordinate project execution and control	Provides for an whole, unified directional information set	Plan interface
Deliverable summary	Defines and lists all deliverables to be produced by the project	Provides visibility, tracking, and reporting of deliverables	Deliverable outputs interface
Project log	Records essential information for each project risk, issue, action item, and change request	Provides visibility, tracking, and reporting of items impacting the project	Log interface
Change request form	Records essential information for any request change that impacts the scope, schedule, or resource requirements (budget)	Provides for the proper assessment and communication before a change action item is taken	Change interface
Project repository	The location where all pertinent project information is stored	Provides a single source of reference for all project information	Project database and search
Project interface (notebook)	Software tool used by a project coordinator and by project contributors to record and interface with a project	Provides the interface	Software

TABLES

Table 10. Project Approach > Coordination*: Project management standard differences.*

	PMBOK (2016)	ISO 21500
Process groups	Initiating Planning Executing Monitoring and Controlling Closing	Initiating Planning Implementing Controlling Closing
Knowledge areas (subjects or activities)	10 Knowledge Areas (KAs)	10 Subjects

Table 11. Project Approach > External Standards: *Popular established references by development category.*

Development Category	Description	Popular standard or reference
Product standards or guides	Characteristics related to quality and safety	ISO 9001 Quality Management Systems
Process standards or guides	Conditions under which products and services are produced or packaged	ISO/IEC 15288 systems and software engineering – system life cycle processes
Project management standards or guides	Helps organizations to manage their operations or project	PMBoK

Table 12. Project Approach > Contributed Deliverable Project State: *Project state variables. There are 4 variables that describe the current state of each project: Version (∞ potential values): the released version accessible to the public. Stage (4 potential values): the level of readiness/completion of the current version. Status (4 potential values): the type of activity for the current stage. Dependency (2 potential values): is development blocked by one or more dependencies.*

Version	Description
0	No released version.
1	Initial version release.
2..n.. ∞	Subsequent releases/updates.
Stage	**Description**
Development (Dev)	Active development/work of current version is underway.
Alpha	Early testing of current version is complete.
Beta	Early testing of current version is complete.
Production (Prod)	Full production release of current version is ready.
Status	**Description**
Todo	Current stage not yet started.
Active	Current stage under development.
Under Review	Current stage read to be reviewed for editing and/or testing.
Done	Current stage is complete.
Status	**Description**
Blocked	Development is blocked by a dependency.
Ready	Development is ready to continue.

TABLES

Table 13. Project Approach > Work*: Work product classification scheme.*

WP ID	Generic work product class	Generic work product description	Generic work product typical characteristics
1	Object	An entity created to serve a purpose, or created in the course of serving that purpose. Its existence is observable and rationalised by its material or behavioural characteristics. It may exist as a complete, partial or exemplifying realisation of a product, be a subordinate part of a product, be a by-product or be a part of an enabling system	identity, name of object purpose, value that caused its creation ownership and responsibility for object status, state and classification of object distinguishing observable qualities and properties functional and behavioural characteristics dimensional and parametric characteristics relationship with and dependencies on surroundings observable interactions or effects on other objects interfaces, connections to surroundings location, position in surroundings safety, security, privacy and environmental regulations
2	Description	An account or representation of a proposed or actual object or concept. It may be a textual, pictorial, graphical or mathematical representation. It may be in a standardised form for human or machine interpretation. It may be a static or dynamic model or a simulation representing reality. It may establish order, structure, grouping, or classification.	object, subject or class represented purpose and applicability of description concerned parties, viewpoints, views range of use, and validity of description accuracy, detail and abstraction level model dimensions, degrees of freedom description language, notation, nomenclature applicable standards, formats and styles representations of function, attributes, properties descriptions of architecture, arrangement, interfaces depiction of composition or form definition of classification, category, ranking, type
3	Plan	A proposed scheme or systematic course of action for achieving a declared purpose. It predicts how to successfully accomplish objectives in terms of specific actions, undertaken at defined times and employing defined resources. It may apply to technical, project or enterprise actions. At a high level of abstraction it may be a policy or, with reference to assets and their disposition, a strategy.	definition of undertaking, purpose and objectives of plan strategy and policy guiding plan plan owner, stakeholders, responsible parties and their authorities plan status, version, reviews and modifications proposed events, actions and tasks predicted timescales, durations, dates of actions assumed dependencies, conditions, constraints, risks allocated resources, labour, facilities, materials planned budget, cost, expenditures defined milestones, results and progress targets decision points and authorisation gates options and contingency actions
4	Procedure	A declared way of formally conducting a customary course of action. It defines an established and approved way or mode of conducting business in an organisation. It may detail permissible or recommended method in order to achieve technical or managerial goals or outcomes.	purpose, outcomes and results of performing actions issuing authority and controls roles, responsibilities and duties actors, their competence and proficiency dependency on requirements, standards and directives achievement, goals, completion criteria definition of transformations and their products work definitions, instructions to act progression and dependencies of action guiding method and practices enabling tools and infrastructure
5	Record	A permanent, readable form of data, information or knowledge. Accessible and maintained evidence of the existence or occurrence of facts, events or transactions. It may take the form of a journal chronicle, register or archive. It may contain the information to confirm achievement of performance, fiscal or legal conditions or obligations.	record identity or title content, description and reason for record ownership, origin and authorship practices, agreements, commitments and regulations applying to record authorities and condition of storage, retrieval, replication and deletion medium and format of record location, conditions and periods of storage applicable information privacy, security and integrity declaration of status, configuration and baseline information information on audit, validity and history

TABLES

WP ID	Generic work product class	Generic work product description	Generic work product typical characteristics
6	Report	An account prepared for interested parties in order to communicate status, results or outcomes. It is a result of information gathering, observation, investigation or assessments, and it may impart situation, affects, progress or achievement. It serves to inform so that decisions or subsequent actions can be taken.	purpose or benefit of repot source, author and authority to report interested parties, recipients, distribution knowledge, understanding communicated information, data, facts and evidence contained analysis, inspections and audits employed timing, validity, condition of information use dependence on circumstances, constraints and assumptions reported status, results, achievements, conformance, compliance or outcomes identified faults, failings or errors inferred patterns, trends or predications conclusions, recommendations, rationale
7	Request	A communication that initiates a defined course of action or change in order to fulfil a need. This may originate or control on-going action based on an agreed plan or procedure. It may result in a proposal or plan of action. It may take the form of a solicitation, requisition, instruction or demand for a resource, product, service or an approval to act.	objective, purpose or outcome of request expression of a demand, need or desire communication of enquiry, solicitation or an order to provide initiation of supply, provision or support definition of action, change or exchange identification of required products, services, capability or resources authorisation of tasking or commitments specified terms, conditions to act, agreement conveyed required availability of requested provision communicated
8	Specification	Criteria or conditions that place limits or restrictions on actions, attributes or qualities. It establishes measures or qualities for determining acceptability, conformance or merit. It may be required as part of an agreement or contract.	definition of needs, expectations and circumstances statement of requirements definition of constraints and conditions standards and regulations invoked dimensions of achievement and outcome criteria of conformance, correctness and compliance definition of measures, indicators, limitations, values, and thresholds statements of action and conduct required functions, performance, behaviour or service levels definitions of interfaces, interaction, location and connection conditions of acceptance, permissible exceptions and deviations conditions of change and variation

Approach: Engineering

Travis A. Grant,
Affiliation contacts: *trvsgrant@gmail.com*
Version Accepted: 1 April 2024
Acceptance Event: *Project coordinator acceptance*
Last Working Integration Point: *Project coordinator integration*

Keywords: engineering approach, engineering management, engineering coordination, engineering planning, engineering operations, technical development, design and simulation,

Abstract

Engineering means working to develop a specification, and then, working from that specific specification to construct and operate. In other words, engineering means to work to develop, and then follow, a single (unified) work plan. In general, engineering also carries the connotation of doing 'useful' work (as opposed to doing 'unnecessary' work). Engineering works outward from given goals and specifications, and proceeds systematically. An engineering/-ed system, is a system designed, realized, and operated through real-world process (by engineers) to achieve a particular purpose. Society may be viewed of as an engineering process and resulting deliverable (Read: societal engineering). If society is to be engineered through cooperation and contribution, then it must identify its alignable life-cycles, its life-cycle processes, its means of design and development, its fundamental[ly encodable] system conceptions, its requirements, its construction, its information-based, its informatics base, and its methods of layering and associating societally relevant information. To produce real world useful operations it is necessary to use a system of design and operation that can separate and combine conceptual and spatial information.

Any approach to state change in the material environment requires work. Useful work requires a systems approach to socio-technically coordinated state re-creation. The integration of project coordination (project management) and system engineering is projects engineering.

Graphical Abstract

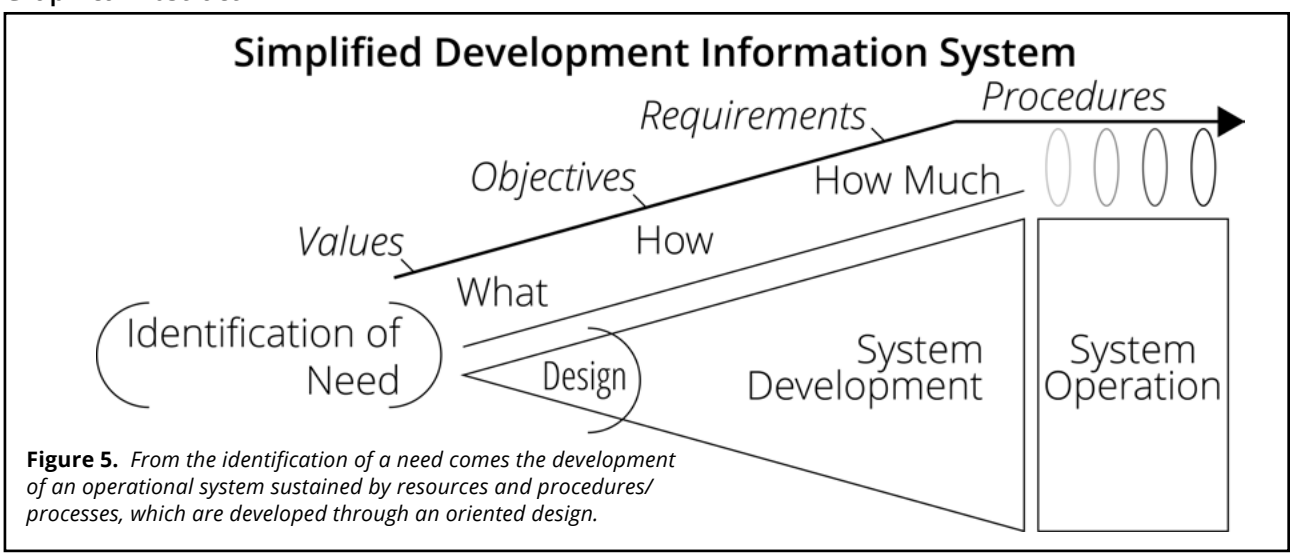

Figure 5. *From the identification of a need comes the development of an operational system sustained by resources and procedures/ processes, which are developed through an oriented design.*

1 What is engineering?

A.k.a., Unified development life cycle approach, societal designed operations approach (a.k.a., organizational design, enterprise design, business design, ...), the in[ter]-system service development approach, the construction service approach.

Engineering means working to develop a specification, and then, working from that specific specification to construct and operate. In other words, engineering means to work to develop, and then follow, a single (unified) work plan. In general, engineering also carries the connotation of doing 'useful' work (as opposed to doing 'unnecessary' work). Engineering works outward from given goals and specifications, and proceeds systematically. An engineering/-ed system, is a system designed, realized, and operated through real-world process (by engineers) to achieve a particular purpose.

Engineering is the application of the principles of science and mathematics to develop effective solutions to socio-technical problems. In society, engineering is a purposeful activity directed toward the goal of fulfilling human requirements through socio-technical [service] design; particularly, those needs that can be met by socio-technical composition. As a project cycles from an idea to the implementation, delivery, and operation of a product or service, engineering links logic and scientific discoveries to functional applications that meet individual and societal needs.

INSIGHT: *Losing function is losing the capacity to do something.*

Technology (and its operation) is the direct result of engineering. However, scientific inquiry and engineering, together, are the basis for all technology. Useful categories of objects (or systems) constructed and/or operated by engineering are called 'technology'. Technology (and its operation) is the practical application of engineering knowledge including procedural (informed by scientific inquiry).

Engineering, as an approach, is:

1. A real world, technical, problem solving activity that uses data, knowledge, and tools to materialize systems. In this sense, engineering is the materializing and materialized aspect of a societal information system.
2. The knowledge required, and the process applied, to conceive, design, make, build, operate, sustain, and/or recycle a system of technical content for a specified purpose (e.g., a concept, a model, a product, a device, a process, a system, a technology, etc.).
3. The application of knowledge and tools in the form of a process to solve discrete problems in the real world (i.e., engineering is concerned with real-world processes using scientific knowledge).
4. The design, production (development), and operation of systems that must work as expected, and hence, engineering is concerned with observable (or experienceable) outcomes (Read: knowledge applied to develop a technical solutions to a discrete problems).
5. Methodically (and systematically) conceiving and implementing viable solutions to existing problems.

Engineering necessitates organizational understanding -- the ability to organize information for a purpose. In engineering, organizational engineering requires an understanding of how to extend (i.e., enhance) the capabilities of the whole, while attempting to better understand the relationships and interactive effects among the components of the organization, and with its environment. Engineering outputs should maintain and/or improve the quality of life among a community of [technical] users. Typically, the conduct of engineering leads to systems that enable and enhance the capabilities of humans, while also responding to the needs and constraints of humans. Therein, engineering is responsible/accountable for the design, implementation, operation and maintenance of a real-world system.

To the engineer, engineering (constructing) anything into existence is based at a fundamental level on [concept- and object-systems] modeling. To the user [of technical systems], engineering represents a technical, knowledge using, life fulfillment support process.

Engineering is composed of (i.e., involves):

1. The process(es) of designing and later operating [planned production] systems based on logic and scientific principles (i.e., scientific knowledge), cycling productions through a habitat, forming a habitat service system (i.e., city) that fulfills human needs.

1.1 The core engineering processes

Engineering consists of two primary processes, each of which has multiple sub-processes:

1. **The development (including design) process** - the step-by-step development of a service or object.
2. **The operations process** - the step-by-step operation of a service or object.

The complete systems engineering life-cycle contains both a system development life cycle and a system operations life cycle. In terms of physicality, a core engineering system must account for:

1. **Flows** of physicality and stocks of physicality. A flow is a variable that measures a quantity per time

period. The motion of objects.
2. **Stock** is a variable that measures a quantity per point in time. The repository of objects.

1.1.1 The development (including design) process

Engineering to create future systems that operate in real-time. In concern to the design (and development) of systems, engineering is a design process, combining knowledge of the properties of materials, models that predict how these materials behave, and systematic thinking, to create solutions to human needs in physical matter reality (i.e., in the real world).

Early in the system development activity, a system is conceptual in nature. A system may consist of several levels where each element at each lower level may by this definition itself be considered a system (i.e., a subsystem of a large system may itself possess all of the attributes of a system).

Engineering will define:

1. The technical specification of the projected system.
2. The technical specification for the system's complete delivery, including integration and eventual de-integration.
3. The method of technology involved in executing the project.

1.1.1.1 Engineering design de-composition

The axiomatic dimensions of real world engineering (Read: development and operations of a system) by means of a project structure must account for that which existence is composed in order to bring something new (or a change to) systems in existence.

In the real world, a project has 4 axiomatic dimensions:

1. 1D - memory (knowledge).
2. 2D - direction (adds objective).
3. 3D - spatial construction (adds resources).
4. 4D - schedule (adds time).

In the market[-State], a project has 1 additional axiomatic dimension:

- Market-based 5D - transaction cost (adds market expense)

In the community, a project has 1 additional axiomatic dimension:

- Science-based 5D - environment (adds probability)

Combination:

1. 1D + 2D => information model.
2. Information model + 3D (space) => physical model.
3. Physical model + 4D (time) => service model.
4. Service model + 5D (cost) => profit model.
5. Service model + 5D (environment) => probabilities model [that humans will have the fulfillment of their needs through a service environment].

Herein, a material[-ized] service system is made up of software, hardware, and data that provides its primary value by the execution of a service for its users.

> **INSIGHT:** *To build something from the ground up "you" have to understand it in a way that "you" may not have to understand when "you" are looking at something that is already built.*

Different societal configurations have different interfaces. In the market-State, the following interfaces are required/present, which are not required/present in a community-type configuration.

State interface requirements:

1. **Contractual agreements** with an authority (jurisdictional or otherwise).
2. **Financial exchange** of currency.

Market interface requirements:

1. **Contractual agreements** with competing market entities (and an authority to enforce contract with punitive/retributive damages).
2. **Financial exchange** of currency.
3. **Demand and delivery** of object(s) or service(s).

1.1.2 The operations process (as engineering)

Engineering upon created system that operate in real-time. Note that engineering operations my involve engineering design and development. Ensuring that the [physical] behavior of the various components of a system are coordinated as required, to ensure a proper functioning of the whole system.

If there is engineering development, then there is:

1. **Development** of a new system, or
2. **Modification**, upgrade, change, iteration to existing system/product.

If there is engineering operations, then there is:

- **Operating** an actually measurable system, that can be monitored, and possibly, controlled.

1.1.3 Measurement and engineering

Engineering measurement can be categorized in two ways:

1. Direct measures - measures of the engineering process (e.g., effort, resources, and cost applied) and product (e.g., produced, lines of code (LOC), etc.).
2. Indirect measures - measures of the product (e.g., functionality, quality, complexity, etc.).

Engineering measurement requires normalization of both size-oriented and function-oriented metrics:

1. Size-oriented metrics (a.k.a., size-oriented key measures):
   A. For example, lines of code (LOC) can be chosen as the normalization value:
      1. Errors per KLOC (thousand lines of code).
      2. Defects per KLOC.
      3. Cost ($) per KLOC.
      4. Pages of documentation per KLOC.
   B. Function-oriented metrics (a.k.a., function-oriented key measures)
      1. The most widely used function-oriented metric is the function point (FP). A function point (FP) is a unit of measurement to express the amount of functionality (societal functionality, business functionality, etc.) an information system provides to a user. NESMA FPA Method: ISO/IEC 24570:2005 Software engineering - NESMA function size measurement method version. Computation of the FP is based on characteristics of the system's information and physical domains, and their complexity. To determine the number of FPs, classify a system's features into five classes:
         i. Transactions - external inputs, external outputs, external inquiries.
         ii. Data storage - internal logical files/objects and external interface files/objects.
         iii. Note: Each class is then weighted by complexity as low, average, or high. Then, the result is multiplied by a value adjustment factor (determined by asking questions based on a set number of system characteristics).
   C. Object-oriented metrics:
      1. Number of scenarios scripts (use-cases).
      2. Number of key classes.
      3. Number of support classes (required to implement system, but are not immediately related to the problem domain).
      4. Average number of support classes per key class (analysis class).
      5. Number of subsystems (an aggregation of classes that support a function that is visible to the end-user of a system).

1.2 [Systems] Usability

Systems are used by users; to be usable by a user, systems can be designed to be usable. The use of a system to is user is usability. The International Standards Organization (ISO) defines usability as "the extent to which a product can be used by specified users to achieve specified goals" (ISO-9241-11, 1998). Usability is a key element of the human-centered design (HCD) approach, and it has been shown to increase efficiency, effectiveness, and user satisfaction. Furthermore, designs with good usability can reduce errors, fatigue, training time, and overall life cycle costs. Usability is a key component of human-centered design. Human-centered design focuses on users' needs to design the system based on users' capabilities. Usability testing and evaluation methods provide user performance measures and subjective (qualitative and quantitative) comments that can be used to improve the system in question throughout the engineering design life cycle.
Usability testing and evaluation is an iterative process. Usability evaluations should be conducted several times during the life cycle of the system, and results should have a direct influence on system design, providing continuous feedback for the designers of the system. Usability should be part of the system development life cycle from the earliest stages, to make sure that users' needs, capabilities, and limitations are considered from the start of design and development.

Standards, as a control(s), make usability efficient. For example, a vehicle pedal set is standard to all vehicles. Any user can get in any vehicle and the foot pedals operate similarly, thus providing interoperability for a user.

INSIGHT: *From usability originates reusability.*

1.3 [Systems] Engineering

Technically, all engineering is "systems" engineering. In the past, many engineering organizations did not follow a systematic approach, and hence, the term 'systems' was added to engineering to emphasize its essential systematic approach. The word systems also connotes that engineering is an information-based process. If differentiated, then reasoning about systems (i.e., systems *reasoning*) is the essence of [systems] engineering. However, take note that in the market, the term 'engineering' often refers to discrete instances of the application of engineering, whereas the term 'systems engineering' often refers to the oversight of engineering at the organizational (or management) level. The term system is added to the term engineering because that which is being developed and operated through engineering is a system (pattern). Herein, systems thinking is a way of dealing with increased complexity. The fundamental concepts of systems [thinking] involve: understanding how action and decisions in one area affect another, and that the optimization of a system

within its environment does not necessarily come from optimizing the individual system components. To do system engineering, someone (or something) must understanding what a system is, its context within its environment, its boundaries and interfaces and that it has a lifecycle. Fundamentally, systems engineering is a global[ly integrated] engineering approach. Note here that because 'community' is a 'unified system', practically speaking, the terms 'engineering' and 'systems engineering' are synonymous unless specified otherwise (as would need to be specified for market-State conditions).

> **CLARIFICATION:** *Engineering complex systems necessitates a project-based approach for purposes of optimal coordination. In the [systems] engineering approach, the project, itself, is a system that applies all the principles of [systems] engineering: it has a purpose, interacts with an environment, and represents a solution to users' requirements.*

All engineering (in community at the organizational level) is, technically, systems engineering. Engineering has always implicitly drawn on systems-oriented principles and practices. However, a distinguishing characteristic of systems engineering is its continual reference and orientation towards an explicitly, developed body of systems reasoning, knowledge, experience and practice. Much of this body of knowledge has come from studies in control engineering, cybernetics, information science, biology.

> **HISTORICAL NOTE:** *The term 'system' was introduced into engineering in the 1940s, leading to the rise of 'systems engineering' in the 1950s and 1960s.*

Systems engineering is the engineering process to create and operate a system. It is a structured process based on data. Take note that there are a large range of accurate definitions in the literature for both engineering and systems engineering. In its most broad definition, [systems] engineering is the process of bringing into existence a functioning, technical system for some user.

> **CLARIFICATION:** *Engineering does not proceed by straightforward application of natural science. Constructions derived from scientific theory have to be tested (and usually, modified) in order to obtain practical, useful technology. Engineering must follow the natural laws and rely on the basic resources in nature such as materials and energy. In science, there is a discoverable, initial whole. In engineering, the whole (design-solution) does not initially exist; it is constructed.*

The following is a common list of definitions of the concept of 'systems engineering':

1. The systematic application of science, tools, and methods to find an effective solution to a problem using a quantifiable approach to create for the development, operations, and maintenance of systems.
2. The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software (ISO/IEC/IEEE 2012).
3. The systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software (IEEE 1990).
4. The interdisciplinary approach governing the total technical and managerial effort required to transform a set of customer needs, expectations, and constraints into a solution and to support that solution throughout its life (BKCASE 2017; ISO/IEC/IEEE 2010).
5. The interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining user needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem (BKCASE 2017)
6. The integration of disciplines into a team effort forming a structured development process that proceeds from concept to production to operation. (INCOSE 2012).
7. A disciplined approach for the definition, implementation, integration and operations of a system (product or service) with the emphasis on the satisfaction of stakeholder functional, physical and operational performance requirements in the intended use environments over its planned life cycle within cost and schedule constraints. Systems engineering includes the engineering activities and technical management activities related to the above definition considering the interface relationships across all elements of the system, other systems or as a part of a larger system. (NASA Systems Engineering Handbook SP-601S).

In the context of the engineering approach, systems engineering is:

1. The coordinated design, development, and operation solutions that retain optimal systems-level performance for specified objectives. Therein, in order to develop and sustain optimal performance, systems engineering uses information from a whole/unified information system.
2. The iterative and interdisciplinary processes of designing and developing new [systems] solutions to complex real-world problems by transforming

requirements into operational systems.

Systems engineering is composed of (i.e., involves):

1. A set of procedures (i.e., practices) that rely on enabling competencies (knowledge sets) and structures (organizational and procedural) at individual, team, and organizational levels to coordinate the design, development, and operation of solutions that maintain optimal systems-level performance.
2. The processes of designing, developing, and operating a system(s) embedded within a life-cycle. Thus, systems engineering is, in part, focused on the long-term and life cycle of a system, necessarily taking into account the cradle-to-grave (or, cradle-to-cradle) life of the system.

   NOTE: *In the market, systems engineering is defined as part of a continuum of business processes. In community, systems engineering is defined as part of a continuum of organizational processes.*

In general, the output of [systems] engineering entails two interrelated viewpoints:

1. The system as a created product, which is used by users.
2. The system as a delivered service, which is serviced by technicians.

The term [systems] engineering can be applied to:

1. **The system (i.e., the solution, itself)** - The [design and life cycle of the] system to be developed and operated (i.e., the technical system itself).
2. **The decision system** - the decisioning and organization that brings the system, itself, into existence. The system that controls and coordinates (i.e., decides, "manages") the development of the technical solution-system.

In general, [systems] engineering must account for:

1. The whole [systems] development process.
2. Integration of a new system (or system's state) into an environment of existing systems.
3. The life-cycle of the new system in an environment of existing life-cycles.
4. Planning the operation of, and the actual operation of, the system.

   QUESTION AND ANSWER: *What is [always] in operation? A developing system is [always] in operation.*

The word 'systems' in the term 'systems engineering' implies, in part, that the systems engineering approach is (i.e., the [systems] engineering approach maintains the following characteristic -- in order to enable the realization of successful, optimal systems, systems engineering is):

1. **Interdisciplinary / multidisciplinary** - engineering has access to all available "branches" of knowledge.
2. **Holistic / unified** - engineering has access to all available data and information while considering the full system, including any number of performance criteria, as well as potentially non-technical concerns related to human factors or societal impacts. Engineering has access to all relevant information to the problem, context and situation.
3. **Integrative (integral)** - engineering combines all available information, including that which is learned during the engineering process itself, into an optimal solution. Engineering requires the integration of multiple views and information sets. Engineering accounts for the whole, as well as the parts that makeup the whole).
4. **Completeness** - engineering completely satisfies the problem with a solution.
5. **Procedural (documented and planned process/ method)** - engineering requires identification, documentation, and improvement upon a method/process. Engineering defines methods of specification, prediction, and control.
6. **User-/Value-driven (utility)** - engineering considers the needs and interests of all users and stakeholders (of everyone impacted by the system).
7. **Collaborative** - engineering involves working with other teams and systems in a sufficiently open information space to produce a safe and reliable system. Systems are developed by teams of engineers, and everyone must be able to understand one-another's work (Read: readability/ understandability).

[Systems] Engineering enables (or, brings) to an organization the following [value] alignment characteristics:

1. **Correctability (correctness or correct alignment)** - the system is capable of adjusting action and information to a direction, standards, or need. All actions/decisions are correct according to the organization's direction. The system ensures that the correct [technical] tasks get done during development and/or operation.
2. **Validity** - the system is capable of taking decisions and actions that are correct and relevant to the problem-requirement. Given a relevant direction, every action should relate to that direction.

3. **Relevancy** - the system is capable of measuring the of pertinence to context, problems and needs.
4. **Consistency** - decisions and actions are consistent with other decisions and actions, and the organization's direction.
5. **Minimality** - the system is capable of meeting requirements exactly.
6. **Extensibility (adaptability)** - the system is capable of adapting to changing requirements.
7. **Flexibility** - the system is capable of integrating new information flows.
8. **Non-redundancy** - the system is capable of informing and acting without unnecessarily repetition.
9. **Value** - the system is capable of delivering the intended benefit to individuals and society.

[Systems] Engineering must necessarily account for:

1. **Complexity** - the interrelationship between multiple, seemingly separate, information sets (and viewpoints, which requires multi-view analysis).
2. **Uncertainty** - during a system's design, there are unknowns; and during a system's operation, there may be unknowns.
3. **Potentiality** - that [dimension] from which [axiomatically] systems ("things") can emerge.

> **CLARIFICATION:** *In some engineering design cases, given what is known and available, sub-systems and component designs may need to be sub-optimal in their designs for the who system to be optimized.?*

An organization's [systems] engineering capability may be equated to its ability to dependably conduct activities that traceably flow from a/an:

1. Knowledge base [documentation base].
2. Experience team base.
3. Competent team base.
4. Enabling systems [a set of relevant organizational assets].

There are [at least] three ways that humans can be involved in engineering systems:

1. Being the designer/developer of the system (e.g., design/development engineer).
2. Being an operator within the system (e.g., technician engineer).
3. Being a user (i.e., requirer) of the system.

Fundamental inputs of an organization's [systems] engineering capability are:

1. Information systems (organizational, decisioning, etc.).
2. Human systems (personnel with abilities).
3. Equipment systems (tools, facilities, etc.).

Engineered systems (or engineered specifications) may change given:

1. New scientific/engineering knowledge.
2. New problems/requirements.
3. New technologies (i.e., new systems).

> **NOTE:** *There is no need to put the term 'evidence-based' in front of 'engineering', because it is assumed.*

1.3.1 System of systems engineering

There is the concept of 'system of systems engineering' (SoSE), wherein, the term "system of systems" (SoS) is somewhat problematic. From a cybernetic perspective, a SoS is a meta-system, an integrated system composed of other systems. Thus, the concept of "system of systems" is tautological, since systems themselves are considered to be comprised of sub-systems, and therefore, a "system of systems" is itself just a system. In general, the term meta-system and system of systems, specifically, refers to a system with multiple embedded and inter-related autonomous complex sub-systems. These sub-systems can be diverse in technology, context, operation, location/geography, and conceptual frame. These complex sub-systems of a meta-system must function as an integrated meta-system to produce desirable results in performance and to achieve a higher-level purpose (mission, etc.) subject to constraints. In other words, a system of systems or meta-system is generally defined as an assemblage of components, themselves considered as systems, with the added distinction of coordinated and operational independence of components.

A SoS brings together systems in order to perform a higher level mission/purpose of which each member system plays an integral role. An SoS is a 'complex system', and as, such exhibits dynamic and emergent behavior and requires engineering to design and operate.

Common definitions of system of systems engineering (SoSE) include:

1. The design, deployment, operation, and transformation of a [meta]system that must function as an integrated complex system to produce desirable results.
   A. The integration of multiple, potentially previously independent, systems into a higher level system (meta-system).
   B. The functional design of a SoS that generates capabilities beyond what any of the constituent systems is independently capable of producing.

It is important to note that when previously independent

systems are integrated (i.e., at the time of integration) that there exists some degree of constraint imparted by their part/position in the larger system (i.e., in the meta-system).

1.4 The structured systems analysis and design method (the SSADM)

This *Solution Inquiry* process follows an [agile] structured systems analysis and design methodology (aSSADM). It is a systems engineering methodology and involves systems-based processes structured in such a way as to produce well-documented, accurate design outputs. It uses a formal, methodical approach toward the analysis and the design of solutions as components of systems (real world community > habitat system > habitat subsystems).

The SSADM herein follows a modified waterfall life-cycle model starting with a requirements analysis, leading to a [comprehensive] technical feasibility study (is it technically possible), and progressing through to the physical design stage of development, while accounting for qualifying requirements, protocols, tasks, and resources. One of the main features of SSADM is the intensive user involvement in the requirements analysis stage. Every good and service is designed in transparency to the entire community and the community can improve the design by discovering more about the natural environment and combining known elements in uniquely creative ways.

Engineering designs involve layers of functional diagrammatic representation. The product of a structured method is a technical design specification that can be engineered into the habitat through the re-organization of resources.

The most efficient form of action under a systems-based approach is that of the project-based approach (i.e., team-based approach). The SSADM breaks up issues into their composite projects, stages, modules, steps and tasks – as every well-applied systems/team/project approach does. A 'project' is just a collection of tasks that a team is applying effort toward with the intention of fulfilling a larger and more integrated purpose.

Remember, a system involves at least: inputs > processes > outputs. The systems and individuals involved in the design of solutions arrived at via the *Solution Inquiry* processes derive their input from a common and verifiably founded repository of data, knowledge, and values. Systems design, therefore, is the general process of defining the architecture, components, modules, interfaces, and so forth, of a system [to satisfy specified requirements using an explicit repository of information as input].

> **NOTE:** *The Solution Inquiry processes could impact the time-frame prioritization of an issue's resolution if it requires significant design time.*

1.5 Generative design

A.k.a., Algorithmic design.

Herein, the computer, as an information processing system for calculation and optimization, becomes part of the design process (with the human designer and/or user). Computers can run hundreds, millions and billions of calculations that facilitate the engineered optimization of physical objects and systems. For example, a heat exchanger may be designed by the computer (Read: machine algorithm) - the designer inputs the following data elements: here is the space I want it to occupy, here is the volume, here is the thermodynamic requirements, here are the materials, and here are the technologies (Read: material production configurations) that meet those requirements. And, the computer runs a series of calculations including predetermined engineering formula and the new data inputs, and therein, it works to optimize (to its programmed potential). The result is then tested. Given appropriate engineering rules, it is possible to produce a system, given what is knwon, that is nearly optimal in terms of its engineering characteristics (and aesthetic).

1.6 Generic design information

Engineers apply scientific and technological information in designing services, structures, and systems (i.e., structural service systems). Herein, when an engineer creates a design specification, it has the following generic metadata:

1. **The System Requirements Specification (SRS)** is a specific record of the required characteristics (functional & aesthetic) of a system. It is the characteristics that any solution is required to possess. Usually, it also includes any goals.
2. **The Operational Concept Description (OCD)** is a system-centric description with respect to:
   A. The intended users of the system (human and/or elements of technology).
   B. The intended uses of the system.
   C. How it is intended that the system be used.
   D. The conditions, external to the system, within which it is intended that the system be used.
3. **The Architectural Design Description** (ADD; e.g. SSDD, CONOPS, IEEE 1471 design description, etc.) refers to the identification of the elements of the solution, together with the key characteristics of each element and the concept of interoperation of the elements to satisfy requirements.

1.7 System engineering life-space cycles

To optimize for "environmental performance", impacts (as affects and effects) must be considered through the entire life-cycle of a good, service, or system. Here,

analytical processes measure these impacts to the best of their abilities. And, we formally determine their threshold of acceptability.

The most thorough way of assessing environmental performance factors (including an alignment with a social value set) is through the process of **life-space cycle assessment**. All iterating systems have a life-space cycle (also sometimes known as a "life-cycle"). The results of a life-cycle assessment may be compared against a benchmark, potentially a *safety benchmark*. The objective of a life-space cycle assessments is not only to identify technical feasibility, but also to identify where environmental impacts originate from and make them explicit in such a way that individuals are capable of prioritizing and setting metrics around them.

The systems engineering [inquiry] life-cycle involves the following parallel conceptual processes:

1. Discover the need for new information;
2. Discover the new information;
3. Understand and integrate the information; and then
4. Arrive at an informed and systems-based technical solution re-orientation (through novel, creative information) to the need that generated the inquiry for new information.

> **NOTE:** *In art, people often see what they want to see; in engineering precision creates operational technologies. Stated in an alternative way: in art, 'abstraction' facilitates subjective perception, and in engineering, 'specification' facilitates useful functioning.*

1.7.1 Brief technical overview of systems engineering

> **NOTE:** *In systems engineering, if it cannot be identified through an integrated visual interrelationship, then it is not understood.*

Systems engineering is an interdisciplinary field of engineering focusing on how complex engineering systems and projects can be designed and structured over their life cycles. Systems engineering involves the analysis of users' needs, the identification of required functionality, the explication (or "documentation") of requirements, then proceeding with design synthesis and system validation, all the while considering the context and root of the issue in which the problem has arisen. Essentially, systems engineering concerns the planning, analyzing, organizing, and integrating the capabilities of a mix of existing and new systems into a capability greater than the sum of the capabilities of the constituent parts. Systems engineering involves the process of designing "in-motion" systems (i.e., dynamic systems). System development often requires contribution from diverse technical disciplines. And, the result is one highly integrated information-physical design. By providing a systems (holistic) view of the development effort, systems engineering facilitates the aggregation of all technical contributors into a unified team effort, forming a structured development process that proceeds from concept to production to operation and to updating, and in some cases, to termination and to recycling.

Visualization and structural models play a principal role in systems engineering, and in the communication of experience in general. A 'model' may be defined herein in several ways, including:

1. An abstraction of reality designed to answer specific questions about the real world; or
2. An imitation, analogue, or representation of a real world process or structure; or
3. A conceptual, mathematical, or physical tool for organizing the arrival of a decision.

Systems engineering involves the use of tools and methods to better comprehend and manage complexity in systems. These tools and methods lead to information that is not open to interpretation or speculation, and can be used in engineering material systems. Engineering inquiries have right and wrong answers, and there isn't any "wiggle room" for interpretation. "True enough" isn't "good enough" in the any sciences, let alone engineering sciences. Material structures must be built with intention and accuracy otherwise they put the community of their users at risk.

Some examples of these tools, techniques and models are:

1. System modelling and diagramming.
2. Simulation modelling (i.e. modelling and simulation).
3. Systems architecture design.
4. Optimization design.
5. System dynamics design.
6. Systems analysis.
7. Statistical analysis.
8. Reliability analysis.
9. Probability analysis.
10. Technical, operational, and systems specifications (and views) for visual, blueprinted representation.

Solution Inquiry is a *structured systems* process because projects are structured into small, well-defined activities wherein the sequence and interaction of these activities is specified, and because diagrammatic and other modelling techniques give a more precise [structured] definition that is understandable by the whole community.

The three most important tools in a systematic structured solution-orientation are:

1. **Logical data modelling:** This involves the process of identifying, modelling and documenting data as a part of the gathering of system requirements. The data are classified further into entities and relationships.
2. **Data flow modelling:** This involves tracking the data flow in an information system. It clearly analyzes the processes, data stores, external entities, and the movement of data.
3. **Entity behavior modelling:** This involves identifying and documenting the events influencing each entity and the sequence in which these events happen.

Some of the important techniques and models include:

1. Logical Data Models.
2. Data Flow Models.
3. Requirements Definition.
4. Function Definition.
5. Specification Prototyping.
6. Relational Data Analysis.
7. Entity/Event Modelling (Entity Life Histories and Effect Correspondence Diagrams).
8. Technical Options.
9. Dialogue Design.
10. Update and Enquiry Process Models.
11. Physical Data Design.
12. Physical Process Specification.
13. Physical Design Control.
14. Gantt charts.
15. Critical path analysis provides a method of systematizing our knowledge so that the effect of decisions of order of action can be seen.

Common diagrams include:

1. Activity & state diagrams
2. Class diagrams
3. Sequence diagrams
4. Service diagrams
5. Operational concept and connectivity diagrams
6. Organizational relationship diagrams
7. Formulaic and matrix representations of data.

In a sufficiently technologically advanced and scaled community, computers may be utilized to:

1. Collect, process, and organize information.
2. Produce documentation.
3. Enable rapid amendment of diagrams and other structured information models.
4. Check consistency and completeness.
5. Automate activities that humans have no desire to do.

Systems engineering is a structured process requiring complete documentation and definition of all system requirements. Structured methods have the following characteristics that impact requirements specifications and systems design:

1. Structure a project into small, well-defined activities and objectives.
2. Specify the sequence and interaction of these activities.
3. Use diagrammatic and other modeling techniques.
4. Give a precise (structured) definition to all information concepts.
5. Are understandable by the population.

The general stages of a structured systems engineering process include:

1. Determining feasibility.
2. Investigating the current environment.
3. Determining systems options.
4. Defining requirements.
5. Determining technical system options.
6. Creating the logical design specification.
7. Creating the physical design specification.
8. Each of these stages applies certain techniques and a sequence of analysis. They include conventions and procedures for recording and interpreting the information with the help of diagrams and language.

The components of the structured solution process include:

1. Structures define the frameworks of activities, steps and stages and their inputs and outputs.
2. Techniques define how the activities are performed.
3. Documentation defines how the products of the activities, steps and stages are presented.
4. Inputs and outputs identify the information egress and ingress.
5. Processes define the integrated flow of information in the system. Some processes operate at the systems level and other operate at sub-levels.

The logical system specification:

1. Broad specification from systems analysis.
2. Technical solutions to the requirements are evaluated.
3. Detailed logical (non-technical) design developed which shows clearly how the new system will operate within the total system; how it will integrate.
4. Narrative and system models are used.

Physical design:

1. Logical design converted to a physical (material) one.
2. The arrangement of engineered structures into a blueprint.

Also, system disruptions must be planned for when designing an engineered system. The general process known as 'systems continuity' may also be known as disaster recovery, fail-safe recovery, system redundancy, and service continuity. An intelligently designed habitat service infrastructure would maintain distributed failsafe and the [buffered] redundancy of systems to ensure system continuity in the case of a planned or unplanned incident. Distributed centralization processes minimize the spread of damage in the case of an incident to a service system.

> **APHORISM:** *Resiliency calms economic panic. Coordination calms social panic. Empowerment calms individual panic. Awareness calms egoic panic.*

1.7.2 Defect and flaw improvement

Defect remove efficiency (DRE) can be calculated and can used at both the project and process level:

- DRE = E / (E + D), [E = Error, D = Defect]
- Or, DREi = Ei / (Ei + Ei+1), [for ith activity]
- Optimize by achieving a DREi that approaches 1

Defect removal efficiency

- DRE = E I(E+D)
- Where, E is the number of errors found before delivery of the system to the end user. These are errors because they are before delivery.
- Where, D is the number of defects found after delivery. These are defects because they are after delivery.

Flaw improvement processes generally include:

1. Error - a flaw in an [engineering] work product that is uncovered before the system is delivered to the end-user.
   A. Defect - a flaw that is uncovered after deliver to the end-user.

1.8 [System] Engineering control

Organisms must be able to keep the conditions inside their bodies stable, even when conditions in their surroundings change significantly. For example, human body temperature stays relatively steady despite changes in the environmental air temperature. The maintenance of a stable internal biological conditions is called, 'homeostasis' (also sometimes, and more accurately, known as 'homeodynamics'). Similarly, societies must be able to keep the conditions inside their habitat stable, even when conditions in their surrounding environment change significantly. At the societal level, this [ability to] control comes from specificationing and planning. For example, your access to food stays steady despite changes in the food condition of surrounding nature during winter when food is scarcer in nature. The maintenance of a stable internal economic-access condition is 'econostasis' (also sometimes, and more accurately, known as 'econodynamics'). Econostasis/-dynamics are terms used to describe an access protocol that accounts for a knowable (in this sense, static) and changeable (in this sense, dynamic) environment. All access protocols are engineered, and are forms of control. Together, through openness (Read: open source), humankind can study and develop economic access protocols that facilitate and optimize the condition of complete human need fulfillment.

1.9 [System] Types of real world engineered control

There are [at least] three types of real world (socio-technical) systems, all of which may be engineered:

1. **Social [organismal] systems** - the behavior of organisms.
2. **Technical hardware systems** (a.k.a., material [information] systems) - the behavior of material technology.
3. **Technical software systems** (a.k.a., digital [information] systems) - the behavior of digital technology.

> **HISTORICAL NOTE:** *The concept of software engineering emerged with the development of computing and information sciences (in its modern meaning) around the 1960s.*

Technology is a result of engineering - the extending of human mind-body function. Technology is the result of applied [scientific] knowledge and engineering processes. Technology refers to the technical systems that engineering designs, builds, and operates.

Technology is:

1. The useful (practical) application of knowledge.
   A. In this sense, technology is engineering operations; it describes the product of engineering.
2. Technology is a capability/function provided by the useful (practical) application of knowledge.
3. In this sense, technology is engineering design and development; it describes the engineering process itself.

1.10 [System] Engineering development levels

The development of engineered systems takes time and work. One framing of the engineering process delineates that which is being developed to be used into levels of developmental usefulness. The following categories classify technology (Read: technology products) thusly.

1.10.1 Technology readiness levels (TRL)

A.k.a., Technology maturity modeling, technology development levels.

In engineering there are technology readiness levels. Generally, there are nine [levels] of them. Technology Readiness Levels (TRL) are a type of measurement system used to assess the maturity level of a particular technology. Here, maturity is a synonym for development or readiness. Each technology (as a project) is evaluated against the parameters for each technology level, and is then assigned a TRL rating based on the projects progress. In general, a technology project is only moved to the next readiness level when the relevant environmental validation is complete for that level.

1.10.2 Level of development (LOD)

A.k.a., Level of detail, level of information detail (LID), amount of information.

Level of development (LOD) is a measure of the level of development by an object (system or element). It is not necessarily a measure of the amount of information, although there must be enough information to satisfy the LOD level of the object itself. LOD is also not a measure of the accuracy of information. Generally, the term LOD is used to refer to elements (i.e., sub-parts) of any single technological system, which itself will have a technological readiness level (TRL). LOD is a measure of "progress", which each level containing a set of parameters.

NOTE: *"Objects" only exist as information (Read: exist as concepts), before they come into being [real] 'objects' through engineering, at the expense of energy and area (Read: light and matter).*

A common level of development (LOD) scale is (the object, 'chair' as the family-type, is incorporated below as an example):

1. **LOD 100** - there is an object (Read: something that has shape and can be pointed to; a thing, product, system), a 'chair'.
2. **LOD 200** - there is a product (object) of specific dimensions -- a chair that has nominal space requirements of 400x400 units.
3. **LOD 300** - there is an object with stated functions and options -- a chair with arm rests and wheels.
4. **LOD 400** - there is an object that is numerically identifiable among other types of its object, and there is a process for producing that specific sub-type of object -- there is a model number for the chair, and a production process for that specific chair.
5. **LOD 500** - there is an object number, a production process for that specific object, and a decision to produce one (or more) of that specific object -- there is the chair's model number, the production process for the chair, and an ordered demand to produce one (or more) of that chair.

There may also be sub-levels:

1. **LOD 200** - final object specification defined.
2. LOD 290 - preliminary construction defined.
3. LOD 292 - checked for functional requirements in construction.
4. LOD 294 - checked for justice-value requirements in construction.
5. LOD 296 - checked for freedom-value requirements in construction.
6. LOD 298 - checked for efficiency-value requirements in construction.
7. **LOD 300** - final construction specification defined.

There are multiple ways of visually representing LOD information. For example (i.e., one way), an LOD numerical identifier structure, such as, "XXXX" where each digit "X" corresponds to a piece of information in the table (e.g. Description, or Width, or Height, etc...) and each of these digits would take a value between 0 and 5 (or 0 and 9 if one needs more granularity). The result would be (taking the 0 to 4 scaling):

1. "0000" means 0% information (with 0% certainty).
2. "1111" means 100% information but with low certainty/development.
3. "5550", "0005" or "5050" all mean 50% information with 25% overall certainty, BUT with clear distinction on the pieces of information that are known and to what level.
4. "9999" means 100% information with 100% certainty/development.

1.10.2.1 Level of development (LOD) sub-categorization

In concern to objects and technology, the idea of a "level of development" can be de-composed into two indices, which together represent a selectable solution:

1. **Level of Information Detail (LID)** - what level of information is present to [have the ability to] materialize the object?

2. **Degree of Certainty (DoC, Level of Certainty, LoC)** - how certain is the execution upon the information to produce the expected result? In other words, how certain are "you" that upon execution of the information the result will be as expected (predicted/specified)?

1.10.2.2 Level of uncertainty

The concept of a 'level of uncertainty' may be generally sub-divided into:

- **Level of Incompletness (LoI)** - a measure of incompleteness.
- **Level of Availability (LoA)** - a measure of what and how much information is available.

1.10.3 Level of design (LOD)

A.k.a., Level of detail (LOD).

There are several commonly identified levels of design:

1. Semantic description of system concepts (a.k.a., paper-based product concept) - these are sketches, narratives (user cases), annotated drawings, graphics, or other concept descriptions that can enable initial explorations of ideas on system functionality to be made, important usability characteristics to be identified, or walk-through studies of protocols.
2. Part prototypes or simulations - Part prototypes are used to simulate specific functional attributes of a design. They might be mock-ups of physical form, scale or mass, mechanical models, static, or animated graphics that enable people to interact with them. The prototype may look nothing like the final design, but will accurately represent those aspects under investigation.
3. Experience prototype - these are representations in any medium that help people to appreciate experiential issues beyond the purely functional attributes of a design. They are designed to include contextual and affective qualities conveyed through a relevant subjective experience.
4. Full prototypes - Full prototypes perform as the final product is intended to perform an incorporate the complete functionality and appearance of the product.
5. Complete product - complete products enable the complete user-interface to be examined. This opens the possibility of carrying out field investigations, comparative studies with other products, in-service studies, etc.

1.10.4 Level of accuracy (LOA)

Level of accuracy refers to the level of accuracy that must be achieved between interoperating models; for instance, when models are created based on a laser scan, what is the level of accuracy that the deliverable model must achieve? For instance, if a beam is (to be) warped in reality, what is the level of accuracy the model needs to achieve, can it just look like a normal beam, does it need to be need to be warped, with what precision does it need to matcb the real world object?

- **Measurement of accuracy (MOA)** - how accurate is the scan data that is being started with? This relates, in part, to the measuring tool (for example, measuring tape is less accurate than a laser).

1.10.5 Social readiness level

Just as technologies have a development readiness level, so do social [mental] models and methodologies. Today, humanity now has access to the systems methodology, and a unified, systems-based (real world) social model for iteratively integrated socio-technical design, construction and operation [of society].

In order to understand and operate complex real world systems, their methodologies (Read: the selected methods that structure the formation of complex systems) must be understood. When a population starts to view society as information, then data and processes start to structure the formation of real world systems, which may be viewed as they are, unified. Socio-individual viewpoint could be considered a new "level" of self-awareness - individuals have access to a unified information system that is pre-configured with data and processes, which are accurately alignable and intentionally programmable to complete in the iterative formation of a material hard-/soft-ware [information] system that fulfills all individual human need, which are never fully known (i.e., there is always more to know). System modeling now exists to assist us in visualizing together so that we can understand and perceive impacts of models, decisions, and actions in our common environment. In this environment, an information environment, all data is fit into a structure (e.g., data model, database) upon which processes may operate. The operation of processes on data requires a control structure to coordinate and control all data and processes. This control structure is "like" a platform, operating system, decision system, protocol, algorithm, ect. (named differently depending upon what level or scale the [whole] society is being viewed from). That control structure can be openly designed and programmed by contributing individuals (you become the ultimate relationship management site, because their reputation on their is ultimate that they would contribute freely, so greatly, which doesn't mean you can't have a secondary pay operation also, it is to say that there are multiple valuable databases here, free though, so no good, well you as source of information as value) or it can be programmed in secret.

1.10.6 BIM readiness levels

BIM readiness level (as model cooperation visualization levels) can be generally separated into:

1. Level 1 BIM is CAD separated files.
2. Level 2 BIM is 2D-3D CAD separated files.
3. Level 3 BIM is CAD with unified file directory revisioning.
4. Level 4 BIM is CAD with life-cycle integration through a unified file directory.
5. Level 5 BIM is data and process simulation, and unified directory file revisioning.
6. Level 6 BIM is societal level development and operations unification.

At the 5-6 BIM levels, the highest level societal services are: "architecture", "structure" (infrastructure), and "MEP" (maintenance, engineering, and planning) may become one integrated systems team sharing a common set of data and process, for example, as separate government and industrial entities, or local habitat sub-services entities (the later is strange to say, because it presupposes a unified system, the habitat).

One would likely rather have a proactive asset and building coordination and control system at all scales of society (like Community), rather than, a reactive one (like the cities and sprawls that early 21st century humans live in).

Possibly, when BIM is referred to in its level 5 context, by industry and government, they are in fact referring to planning (e.g., "public private ownership", etc.) at that level, in definition, as the merger of industry and government as an organization that coordinates the construction of all buildings through the control of design, construction and operation, of the information systems that produce and operate all building-related data and processes.

1.11 [System] Architectural clarifications

Architecture (noun) is defined commonly in several different ways:

1. The art and science of designing and superintending the erection of buildings and similar structures.
   A. The creativity, heuristics and engineering practice of design and technical supervision resulting in man-made systems.
2. A style of building or structure
   A. A recognisable pattern or pro forma of system composition and arrangement.
3. Buildings or structures collectively.
   A. A quality or attribute of systems that conveys composition and order.
4. The structure or design of anything.
   A. The composition and rational arrangement of a system.
5. The internal organization of a computer's components with particular reference to the way in which data is transmitted.
   A. The information technology viewpoint of a computer described according to system form and function.
6. The arrangement of the various devices in a complete computer system or network.
   B. The information technology viewpoint of a computer network as a system}' [Collins, 1991].

In concern to the semantics of these definitions:

1. Meaning 1 confirms architecture to be a body of practice. It is applied to the design and supervision of actions of particular classes of structure, such as vessels, buildings, cities.
2. Meaning 2 conveys that architecture can manifest itself as patterns of significance and value.
3. Meanings 3 and 4 convey architecture to be a collective attribute of systems.
4. In contrast, Meanings 5 and 6 present a contemporary information technology and software use of the term for computers (plus the software representations of data, processes and control that they host) when considered in system terms.

For systems engineering and, as the definitions above suggest, generally for the systems reasoning mind, it is axiomatic that architecture is an attribute of system that characterises a system's order. In the IPTL survey 67% spontaneously identified architecture with structure, with 50% referring to product structure and 17% translating this directly into consequent project structure.

Architecture is thus commonly understood as a description of the composition and structuring of a man-made system; of order that arises from intent and directed design. This is in conformity with the IEEE definition of 1990: 'the organizational structure of a system or component'. That is, a factual listing of parts and their organisation or relationship [IEEE 1990].

A decade later, however, the influential standard IEEE STD 1471 had evolved this definition into 'the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment and the principles guiding its design and evolution' [IEEE 2000]. This definition had moved beyond an objective description of a system-of-interest, extending it to include the setting, if not behaviour, in an environment of operation. It also introduced the notion of the decision (or design) rationale behind these descriptions. In doing so, it began to equate architecture to design actions and the discipline that governs them.

In ISO/IEC 15288 architecture is explicitly associated with one process: architecture design.

Architecting is an invented word to describe how architectures are created, similar to how engineering describes how "engines" and other technologies are created. Possibly, if engineering is the art and science of technical problem solving, then systems architecting occurs when the problem is not yet known. (Maier, 2009)

Yet, a systems engineering and systems architecting distinction would appear to arise partially from values, beliefs and ideas, and hence to be culturally rooted. Etymologically, the word architecture comes from the Greek word "arkhitektonike", which is a combination of two words meaning 'chief' and 'builder'. Thus, the word architect derives from the Greek for "the director of works" or "chief builder" and refers to someone who is responsible for overseeing all aspects of building, and is essentially the integrator of all aspects. Hence, an architect is associated with technical leadership and connotes seniority as much as skill. Typically, it is used in the singular form and is less prominently associated with a team activity. Architecting practitioners have thus elevated the most strategic-thinking, high-level design to be architecting, relegating all else to be termed 'design' (or, engineering), which is then a subordinate/subsequent action to that of architecting.

According to this model of architecting practice, systems engineering is concerned with the conduct of implementation-related design, and architecting with the strategic decision making across all engineering contributions. Architecting then becomes the hub of design. It is a model with seductive promise to those mired in academic complexity, but is in essence barely more than a re-titling or re-stratification of the recursive transformations described by systems engineering.

For purposes of conceptual clarification ("conceptual cleansing"), architecture may be seen as the descriptive essence of systems, and in no sense is it the system itself. Architecture could be viewed as the totality of every possible communicable view of an actual or conjectured real-world system: the summation of all possible transmissible models that inform the existence of a system as an object. It is therefore an abstract notion; a set of descriptions of the nature, arrangement, workings, holistic interaction opportunities, and additionally as preferred, the rationale for the existence of this order.

2 [Engineering] Life-cycle stages

A.k.a., The [systems] engineering life cycle, or the systems engineering method.

Each phase of the systems engineering life-cycle (process) has a similar logic[al set of objects]:

1. Definition.
2. Purpose.
3. Task(s) and activities.
4. Outcomes.

Note that the term 'requirement' is essentially the same as the term 'specification'. 'Requirements' must be sufficiently specific and detailed to allow/ensure **verification** (is the right, correct, planned "thing" being done) and **validation** (user approval).

The engineering processes of service life-cycle coordination are:

1. De-/construction (i.e., dis-/assembly, de-/equipping, etc.).
2. Maintenance (i.e., tasks that maintain a service function).
3. Operations (i.e., tasks that use a service function).
4. Monitoring (i.e., remaining aware in order to apply a control if necessary).

On-service engineering operations are systems and humans that are acting in some capacity through some task as being of service to another human or system. The incident response process, for instance, involves both data and physical incidents. When the incident response service is engaged, humans and systems become on-service to the procedural aid of other humans and systems. Maintenance is a sub-set of the life-cycle processes; it involves being of service to humans by developing and sustaining the systems that facilitate everyone's fulfillment (i.e., "service them").

The following four primary engineering process life-cycle phases:

1. Composition.
2. Maintenance *of composition.*
3. Operation *of composition.*
4. Decomposition.

2.1 Requirements of engaging in systems engineering

The primary deliverables of these systems engineering processes are/include:

1. **Requirements** engineering of the requirements specification.

2. System architecting a **logical systems architecture**.
3. **System design specification** (and standardization)
4. Integration of specification (standardization) into **habitat/information operations**.
5. **Validation and verification** of physical/information system itself is changed as expected.

The systems engineering process requires:

1. **Access** to all available knowledge (and information).
2. **Defining** user needs and required functionality.
3. **Documenting**.
4. **Design** synthesis.
5. System **validation**.
6. While considering the complete **problem**: operations, resources and schedule, performance, support, test, manufacturing, and disposal.

Engineering sub-units include (by task category):

1. **Scientific research.**
2. **Systems design and development.**
3. **Systems integration.**
4. **Systems operation.**
5. **System update and/or de-integration.**

The engineering process chain is initialized as a dynamic problem:

1. **Measure.**
2. **Identify.**
3. **Analyze.**
4. **Design.**
5. **Execute (Act).**

The engineering process chain works to materialize a solution:

1. **Design of system.**
2. **Production of components**.
3. **Assembly of system.**
4. **Testing of system.**

The modeling process for engineering a real-world system requires:

1. **Design** - integrate the concepts, principles, data, and knowledge into a structure with a logical flow.
2. **Design development** = integrate the structure into the logical flow of a specified system.
3. **Production** - apply energy through a vehicle to [effectively and efficiently] modify material or digital information into the specified system. For example, use a knife to whittle wood into a "carved" implement for eating, like a spoon or chopstick.
4. **Service integration** - materially or digitally connect the sub-system to a larger/pre-existing system.
5. **Service operation** - operate/use the system.
6. **Service testing** - of the design occurs throughout the whole process to ensure the solution is as expected by the user (i.e., meets requirements).

The modeling process for engineering a proposed societal system requires:

1. Create - Vision for society.
2. Evaluate - Individual human needs.
3. Analyze - Collect data and analyze situation.
4. Apply - Apply decided procedures.
5. Understand - Visualize results of action.
6. Update - Integrate results of action
7. Remember - Re-envision society.

2.2 The product life-cycle stages

All productions, whether they are objects or services (combinations of objects) go through the following engineered product life-cycle stages (input-outputs of a production/manufacturing/engineering system):

1. Product design (final service and/or object design).
2. Manufacturing system design (what to produce in order to produce the designed product; intermediary designs).
3. Manufacturing .
   A. Production of the manufacturing system.
   B. Production of the product from the manufacturing system.
4. Distribution and storage.
5. Product use (habitat service system operation and user access, together; intermediary and final demand).
6. Disassembly, reuse, re-manufacturing, and recycling.

2.3 The engineering life-cycle/process flows

There are multiple possible views into engineering as a system of processes. The engineering process can be viewed from multiple, correct perspectives. There is commonality between all of the possible perspectives on engineering. Therein, different engineering projects may modify the [unified information system's] common engineering process(es) accordingly.

These activities cover the "cradle-to-grave" or "cradle-to-cradle" life cycle process associated with the major functional groups that engineering provides. The following process views are in their simplified conceptual form.

2.3.1 Technical process flow views

The following are the common technical processes (technical process flows) for the realization of a solution through engineering. Note that these view all follow essentially the space system's process; they just use different terminology to describe the same process (i.e., the same structural flow of information).

The problem-solving view of the engineering process:

1. **Problem input** - initial requirements data.
2. **Analyze requirements data** - obtain answers to requirement questions.
3. **Design solution** - obtain answers to requirement questions.
4. **Test and validate solution** - produce and evaluate the design against the requirements.

The development and operations view of the engineering process:

1. **Analysis** - identify design problem.
2. **Synthesis** - identify design alternatives.
3. **Prototype** - build and test alternatives.
4. **Integrate** - integrate the best selection.
5. **Utilize** - Operate the new system.

The problem-oriented cycle view of the engineering process:

1. Have **problem**?
2. Collect **data.**
3. Design **solution**.
4. Solution **test.**
5. Solution **feedback.**
6. **Integrate** solution.
7. Have **problem**?

The engineering phases view of the material system life-cycle:

1. Conceptual phase
2. Specification and Design phase.
3. Implementation phase.
4. Operations phase.
5. Retirement phase.

The development review completion cycle view of systems engineering:

1. **System Requirements Review (SRR):** At the beginning of the project, establishes what the system will and will not do.
2. **Preliminary Design Review (PDR):** At 10% design completion, is primarily to critique the architecture of the design and critical decisions made in the design.
3. **Critical Design Review (CDR):** At 90% design completion, is primarily to make a last set of changes before the design is finalized.
4. **Validation System Review (VSR)**: At 100% operational completion the system.
5. **Verification System Review (VSR):** User feedback on issue.

The inquired action view of the engineering process:

1. **Inquire** (is a change needed; is a decision present)
2. **Problem situation** (situational analysis; requirements).
3. **Solution formulation** of relevant purposeful models and activities [accordingly, scenarios] of the perceived problem (functional and physical design).
4. **Take action [to realize formulation, reformulation of situation]** in the situation to bring about improvement (implementation, material change/construction).

The serviced view of the engineering process:

1. **Conceive** (Imagine, specify, plan).
2. **Design** (describe, define, develop, test, analyze, validate).
3. **Realize** (manufacture, make, build, procure, produce, deliver, phase-in).
4. **Service** (use, operate, maintain, support, sustain, phase-out, retire, recycle, dispose).

The actionable phase view of the engineering process:

1. **Initiation phase** - recognition of problem.
2. **Analysis phase** - understanding of problem and context.
3. **Design/synthesis phase** - specification of solution to problem.
4. **Implementation phase** - solution production, testing, training, site preparation.
5. **Operations phase** - usage of solution.
6. **Evaluation phase** - observe and review the solution and the process that created the solution.

The problem-action view of the engineering process:

1. **Problem identification** - defining.
2. **Solution abstraction** - modeling.
3. **Solution realization** - building.
4. **Solution utilization** - operating.

The problem view of the engineering process:

1. **Problem detection.**
2. **Problem definition.**
3. **Problem analysis.**
4. **System design problem.**

5. **System manufacturing problem.**
6. **System use/service problem.**
7. **System obsolescence problem.**

Then system state view of the engineering process:

1. **Problem** with world.
2. **Model current state** of world.
3. **Model new state** of world without problem.
4. **Construct new state** of world.
5. **Evaluate new state** of world.

The issue view of the engineering process:

1. **The issue problem.**
2. **The research and discovery problem.**
3. **The design problem.**
4. **The construction and integration problem.**
5. **The operation problem.**
6. **The testing and evaluation problem.**
7. **The maintenance problem.**
8. **The de-integration problem.**

The resource-based view of the engineering process:

1. **Survey** (an environment for planning).
2. **Plan** (a system for building).
3. **Build** (a system for operating).
4. **Operate** (a system for serving).
5. **Cycle** (the evolution of the operating systems).

The strategic-evaluative view of the engineering process:

1. **Planning and analysis.**
   A. Create project concept.
   B. Generate requirements.
   C. Validation.
2. **System [logical] architecting.**
   A. Functional analysis.
   B. Requirements analysis.
   C. System synthesis.
   D. Validation.
   E. Verification.
3. **System [physical] design.**
   A. Physical design.
   B. Composition analysis.
   C. Validation.
   D. Verification.
4. **Build* and test [the system itself].**
   A. System integrations.
   B. Validation.
   C. Verification.

The algorithmic life-cycle view of the engineering process:

1. **Plan algorithmic decisioning.**
2. **Design select algorithm.**
3. **Implement algorithm.**
4. **Assess algorithm.**
5. **Monitor algorithm.**
6. **Iterate algorithm.**

The design alignment view of the engineering process:

1. **Requirements.**
2. **Analysis.**
3. **Development.**
4. **Testing.**
5. **Implementation.**
6. **Support.**

The creation alignment view of the engineering process:

1. **Direction** - put together a specification of the objective.
2. **Conceptualization** - put together a specification of the system. Conceptualization involves the organizing and structuring of acquired knowledge.
3. **Implementation** - implement the concept [specification] model to create and/or operate the system.
4. **Evaluation** - evaluate by doing a technical analysis on the process and result, and correct any mis-alignment with objectives and requirements (system so that all information in all phases is more coherent and/or useful).

The system design view of the engineering process:

1. **Discover** - Why is “it” the “right” output.
   A. Research.
2. **Define** - What is the “right” output.
   A. Ideate.
3. **Design** - Design what is the “right” output.
   A. Specify.
4. **Develop -** Prototype and test the “right” output.
   A. Build and test.
5. **Deliver** - Deliver, integrate and transport, what is the “right” output.
   A. Implement and integrate.

The system integration view of the engineering process:

1. **Requirements.**
2. **Design.**
3. **Implementation, integration, transition, launch.**
4. **Verification.**
5. **Operation.**
6. **Validation.**

The system generation view of the engineering process:

1. **Conception (concept).**

2. **Development assessment.**
3. **Development demonstration.**
4. **Production manufacturing.**
5. **System transition.**
6. **Utilization (in-service operations).**
7. **Retirement (disposal operations).**

The system information view of the engineering process:

1. **Conception.**
2. **Initiation.**
3. **Analysis.**
4. **Design.**
5. **Construction.**
6. **Testing.**
7. **Deployment and release.**
8. **Operation.**
9. **Iterate and Evolve.**

The vision improvement view of the engineering process:

1. **Measure.**
2. **Analyze.**
3. **Improve.**
4. **Sustain.**

The vision to operation view of the engineering process:

1. **Vision.**
2. **Design.**
3. **Transition.**
4. **Operation.**

The system integration view of the engineering process:

1. **Need analysis.**
2. **Situation and concept exploration.**
3. **Concept definition.**
4. **Design and development.**
5. **Integration.**
6. **Operation.**
7. **Evaluation.**

The solution cycle view of the engineering process:

1. **Issue** or **change concept** (for solution).
2. **Development** (of solution).
3. **Integration** (of solution).
4. **Sustainment** (or solution).

The planning view of the engineering process:

1. **Plan.**
2. **Develop.**
3. **Test.**
4. **Deploy.**
5. **Operate.**
6. **Support.**

The project engineering view of the engineering process:

1. **Project definition.**
2. **Specification definition.**
3. **Conceptual design.**
4. **Product design.**
5. **Fabrication (manufacturing).**
6. **Assembly.**
7. **Integration.**
8. **Testing.**
9. **Evaluation.**
10. **Operation.**
11. **Iteration.**

The object (ware) view of the engineering process:

1. **Problem environment.**
2. **Design solution concept.**
3. **Design solution ware** (hardware and/or software).
4. **Construct solution ware.**
5. **Operate solution ware.**

The project transition view of the engineering process:

1. **Design.**
   A. Design assessment (design integration) - define operations and maintenance (O&M) requirements to understand the 'end state' and successful progression into efficient and effective operations.
   B. Design development - develop the [informational] design specification.
2. **Implementation/construction.**
   A. Production - produce and deliver the system to specification.
3. **Commissioning (validate and test).**
   A. Operational readiness - implement, test and validate the operations and maintenance (O&M) activities to ensure a smooth and safe transition into an operational capability. Ensure the system is at a sufficient [checklisted] state if operational readiness.
4. **Operations.**
   A. Operational evaluation - enhance, evaluate, and embed operations and maintenance practices to ensure overall efficiency and effectiveness.
5. **Operational lifespan.**

The structural-informational view of engineering:

1. **Problem (with environment)** - system design process.
   A. Requirements definition process (of system).
      1. User/stakeholder expectations definition.
      2. Technical requirements definition.
   B. Technical solution definition process (of system)

    1. Logical decomposition.
    2. Design solution definition.
2. **Solution (for environment)** - system realization process.
    1. Design realization process (of system).
    2. Integration process (of system into environment).
    3. Evaluation process (of system operating in environment).
3. **Planning (of environment).**
    1. Technical planning process.
    2. Technical control process.
    3. Technical assessment process.
    4. Technical decision analysis process.

The system materialization view of the engineering process (Read: Levels of materialization):

1. **Concept refinement phase** - refine the initial problem/issue/concept/situation into a direction, approach, and orientation [to the state of the environmental societal system, as the solution]. Conceive of why the system needs to be changed and what changes are required.
2. **System development** - Develop a new system, sub-system, or capability (object or service) aligned with the direction, orientation, and approach. Develop the new system state to align with the refined conception.
3. **System deployment** - Achieve a transitional operation of the actual material system that satisfies the refined conception of a direction, orientation and approach (as given in the concept refinement phase).
4. **System operation** - Execute a support program that meets operational support performance requirements and sustains the system over the time of its life-cycle.

The service life-cycle view of engineering (New service life-cycle phases):

1. **Service need.**
    A. Concept studies.
2. **Concept definition.**
    A. Concept and technology development.
3. **Design specification.**
    A. Preliminary design, engineering model (final design), and technology completion.
4. **Production (fabrication).**
    A. Assembly, integration, and testing.
5. **Operation.**
    B. Operations and sustainment.

The situational systems view of the engineering process:

1. **Analyse situation.**
2. **Develop requirements** for system.
3. **Design system** based on requirements.
4. **Build system** based on design.
5. **Use and maintain system** based on design.
6. **Re-cycle system** based on design.

The constructional view of the engineering process:

1. **Informational** (conceptual, object-process).
2. **Virtual** (simulation).
3. **Live** (actualized, material, physical).

The measurement view of the engineering process:

1. **Do a cause-and-effect analysis** - to understand the current situation.
2. **Identify objectives** - to set the purpose for changing the current situation.
3. **Identify requirements** - to set the precise structural outcome(s).
4. **Quantify** - to specify the precise outcome.
5. **Measure the build** - build the precise outcome.
6. **Measure the result** - determine if the build meets the specifically defined quantifications.
7. **Repeat.**

The engineering design view of the engineering process:

1. **Concept studies.**
2. **Concept development.**
3. **Preliminary design.**
4. **Detailed and final design.**
5. **FAIT or SAITL** (FAIT - fabrication, assembly, integration, transition; SAITL - system assembly, integration, testing, launch).
6. **Verification.**
7. **Operation.**
8. **Validation.**

The product plan view of the engineering process:

1. **Concept design.**
2. **Product development.**
3. **Product production.**
4. **Product utilization.**
5. **Product support.**

The coordination view of the engineering process:

1. **Planning:**
    A. Site survey.
    B. Resource survey.
    C. Feasibility analysis/study.
2. **Engineering:**
    A. Process design.
    B. System design.

C. Sub-system designs.
3. **Procurement:**
   A. Acquisition.
   B. Logistics.
   C. Inspection.
4. **Construction:**
   A. Construction planning.
   B. Schedule control.
   C. Construction tasks.
   D. Construction validation.
5. **Service:**
   E. Operations (and maintenance).

The development view of the engineering process:

1. **Research and discovery** (problem inquiry and situation analysis) - identify a problem/issue for which a solution is to be designed.
   A. Identify the problem.
   B. Document and analysis of problem, situation, and prior solution attempts.
   C. Determine solution requirements.
   D. Root cause analysis (process) - similar to that used in solving quality-related problems, can be used to categorize risks according to their source, to list risks in each category, and then to propose preventive actions to prevent these risks, or to develop countermeasures or risk responses if they happen to occur. It can be used as part of brainstorming, the first technique listed, to identify risks.
2. **Design** - develop multiple solution possibilities and through the use of feedback and data, select the best potential solution to pursue.
   A. Generate design concept, analysis, selection.
   B. Application of STEM principles and practices.
   C. Determine design viability.
3. **Prototype and test** - create a testable prototype and unbiased testing plan based on the defined design requirements to determine the effectiveness of the solution created.
   A. Construction of a testable prototype.
   B. Prototype testing and data collection plan.
   C. Testing, data collection, and analysis.
4. **Evaluation of project and process** - seek and document feedback.

The service system existence activities view of the engineering process:

1. **Development (design and testing)** - the activities required to create/evolve the system from user needs to product or process solutions.
2. **Production and construction (create final solution)** - the activities necessary to create the completed solution.
3. **Deployment (fielding of final solution)** - activities necessary to initially deliver, transport, receive, process, assemble, install, checkout, train, operate, house, store, or field the system to achieve full operational capability.
4. **Operation (of final solution)** - the user function and includes activities necessary to satisfy defined operational objectives and tasks in peacetime and wartime environments.
5. **Support (of operational solution)** - the activities necessary to provide operations support, maintenance, logistics, and material management.
6. **Disposal/evolution (of operational solution)** - the activities necessary to ensure that the disposal of decommissioned, destroyed, or irreparable system components meets all applicable regulations and directives.
7. **Training (on operational solution and learnings)** - the activities necessary to achieve and maintain the knowledge and skill levels necessary to efficiently and effectively perform operations and support functions.
8. **Verification (of operational solution)** - the activities necessary to evaluate progress and effectiveness of evolving system products and processes, and to measure specification compliance.

The technical system design realization view of the engineering process:

1. **Systems design processes:**
   A. User expectations defined (imperatives/ objectives).
   B. Technical requirements definition (requirements definition process).
   C. Logical system decomposition.
   D. Design solution definition (solution definition process).
2. **System realization processes:**
   A. System implementation process.
   B. System integration process.
   C. System validation process.
   D. Requirements validation process (technical evaluation process).
   E. System verification process (technical evaluation process).
   F. System transition (transition to user process).
   G. System maintenance process.
   H. System disposal process.
3. **Technical coordination processes:**
   A. Planning coordination process.
   B. Imperatives coordination process.
   C. Requirements coordination process.

D. Resource and tool coordination process.
E. Assessment process.
F. Control process.
G. Risk coordination process.
H. Data coordination process.
I. Interface coordination process.
J. Decision analysis process.

2.3.1 The basic process view of engineering

All engineering project follow essentially the same structure:

1. **Goal** - purpose for the project's [working] existence.
2. **Functional requirement (a.k.a., requirements encoding mechanistic/causative functioning, technical requirements, functional objectives)** - addresses one specific aspect or required performance of a system to achieve the stated goal (note that other functional requirements may contribute to achieve the same goal).
3. **Non-functional requirement (a.k.a., non-functional objectives, requirements encoded values, orienting principles)** - addresses one specific property (characteristic/objective) that is to be achieved by executing the project (note that other objectives may contribute to achieve the same goal).
4. **Operative requirement (a.k.a., performance requirement)** - actual requirements, in terms of performance criteria or expanded functional description.
5. **Verification requirement** - instructions and tools for verification of performance.

The basic process view of engineering involves an information loop:

1. **Requirements analysis** - Requirements analysis is used to develop functional and performance requirements; that is, customer requirements are translated into a set of requirements that define what the system must do and how well it must perform. The systems engineer must ensure that the requirements are understandable, unambiguous, comprehensive, complete, and concise. Requirements analysis must clarify and define functional requirements and design constraints. Functional requirements define quantity (how many), quality (how good), coverage (how far), time lines (when and how long), and availability (how often). Design constraints define those factors that limit design flexibility, such as: environmental conditions or limits; defense against internal or external threats; and contract, customer or regulatory standards.
2. **Functional analysis and allocation** - Functions are analyzed by decomposing higher-level functions identified through requirements analysis into lower-level functions. The performance requirements associated with the higher level are allocated to lower functions. The result is a description of the product or item in terms of what it does logically and in terms of the performance required. This description is often called the functional architecture of the product or item. Functional analysis and allocation allows for a better understanding of what the system has to do, in what ways it can do it, and to some extent, the priorities and conflicts associated with lower-level functions. It provides information essential to optimizing physical solutions. Key tools in functional analysis and allocation are Functional Flow Block Diagrams, Time Line Analysis, and the Requirements Allocation Sheet.
   A. Here, it is important to consider under what conditions an existent function may not be wanted by a user, and hence, should be disableable (Read: disableable by the user).
3. **Requirements loop** - Performance of the functional analysis and allocation results in a better understanding of the requirements and should prompt reconsideration of the requirements analysis. Each function identified should be traceable back to a requirement. This iterative process of revisiting requirements analysis as a result of functional analysis and allocation is referred to as the requirements loop.
4. **Design synthesis** - Design synthesis is the process of defining the product or item in terms of the physical and software elements which together make up and define the item. The result is often referred to as the physical architecture. Each part must meet at least one functional requirement, and any part may support many functions. The physical architecture is the basic structure for generating the specifications and baselines.
   A. Design deliverable (noun) - A design [specification] is a visualization (sometimes, plan) that shows (through to demonstrates via simulation) some combination of function ("workings"/mechanism), performance, and interface of future system.
   B. Design process (verb) - To design means the decisioning processes (groups) that model, determine, and select the function, performance, and interface to be recorded as the executable design, a valid design for integration).
5. **Design loop** - Similar to the requirements

loop described above, the design loop is the process of revisiting the functional architecture to verify that the physical design synthesized can perform the required functions at required levels of performance. The design loop permits reconsideration of how the system will perform its mission, and this helps optimize the synthesized design.

6. **Verification** - For each application of the system engineering process, the solution will be compared to the requirements. This part of the process is called the verification loop, or more commonly, Verification. Each requirement at each level of development must be verifiable. Baseline documentation developed during the systems engineering process must establish the method of verification for each requirement. Appropriate methods of verification include examination, demonstration, analysis (including modeling and simulation), and testing. Formal test and evaluation (both developmental and operational) are important contributors to the verification of systems.
   - Inspection is one method of verification.

2.3.2 The project engineering process

A descriptive view of project engineering includes:

1. Engineering life-cycle:
   A. Engineering.
   B. Pre-concept.
   C. Concept.
   D. Prototype.
   E. Evaluate.
   F. Produce.
   G. Operate.
   H. Maintain.
   I. Cycle.
2. Engineering process stages:
   A. User need definition.
   B. System requirements definition.
   C. Detailed system design.
   D. Prototype, test and acceptance.
   E. In-service feedback.
3. Engineering design and development process:
   A. Needs identification.
   B. Literature/background study.
   C. Task requirements and specifications.
   D. Definition of the goal/purpose of the design.
   E. Ideation and invention.
   F. Analysis.
   G. Selection.
   H. Detailed design.
   I. Prototyping and testing (including validation, certification and standardization as applicable).

2.3.3 The basic concept view of engineering

A descriptive view of engineering includes:

1. **Conceptual design** - the formal transition from the user-issue organization level to the engineering level. In other words, a decision space has now opened an engineering solution space, which the first deliverable of which includes a set of engineering requirements that align with the decision space's resolution objective(s). Traceability from the user-issue with a complete logical description of the system-of-interest into measurement statements (i.e., requirements) for designing and operating a user system without issues. This deliverable set ensures the proper definition/identification/development of the system requirements. This phase has two primary functions: (1) more likely that a solution will optimally resolve a problem; (2) more likely that effectiveness inquiry (a core decisioning process) will return an accurate result, such that HSS operational process will operative effectively due to accurate data, and unsafe projects will be correctly identified and removed from active decisioning, placing them into issue holding.
   A. Concept stage - encompasses all analysis and planning to establish the valid need for a new system. Why does the user need the new system? Establish possibility/feasibility of an architecture (system) that is realizable (based on society's value set alignment.
      1. Valid need - establish that there is a valid need, that the system will be used (in the market, market feasibility - someone will buy the system)
      2. System concepts - exploring potential system concepts/formulations along with valid sets of system performance requirements.
      3. Selection - selection of the most optimal (best fit) system concept (matching). Define the functional characteristics of the optimal (best fit) system concept so that the selected system concept definition can be used to make engineering, productions, and operations plans.
      4. New technology development - certain times, the newly envisioned system will require the development of new technology (because of non-existent technology) - hence, develop necessary technology and technology needed for the system concept, and validate the technology.

B. In engineering, the need associated with a critical gap constitutes the start of the systems engineering lifecycle and the initiation of a conceptual design solution. As mentioned earlier in the process lifecycle, conceptual design includes:
   1. Define organizational needs and requirements
   2. Define stakeholder/user needs and requirements
   3. Define system requirements
   4. Conduct system-level synthesis - this will allow for the selection of the optimal ("preferred") system-level solution (configuration).
   5. Conduct system design review (evaluation)
   6. The output of 5 then becomes the Preliminary Design

2. **Requirements activities** - The 3 types of requirements that form [part of] the system's logical design (requirements flowdown):
   A. **Societal-level organizational needs and requirements** - the value system [engineering] requirements for materialization (their alignment):
      1. In the societal information system, this is represented by: the parallel decision inquiry processes.
      2. The organizational requirements (parallel value-alignment decision inquiry process) ensure feasibility of the solution (i.e., that the solution is a feasible undertaking and integration by the societal system). These requirements control (and guide) the development of engineering requirements for the system. The likely options to a problem (i.e., the possible solution spaces). What society requires from the ultimate solution when it is deployed.
      3. Societal-level requirements activities are also known as the societal requirements specification (or business requirements specification including organizational/ business needs and requirements).
      4. Imperatives and directives such as mission, vision, goals, objectives, needs, etc.
      5. In human terms, tasks (execution) herein are completed by members of InterSystem teams, whereas, because the societal information system involves both tasks by 'individuals' and tasks by 'engineering'.
   B. **User (Read: stakeholder) needs and requirements** (may or may not describe a system's structure and/or behavior, user specification) - How the user describes what is required? How the user determined their [logical] path to arrival at what is required? A description of an experience that has resulted or may result in a lack of alignment with a visualizable objective experience?
      1. In the societal information system, this is represented by: the articulation and recognition decision inquiry processes
      2. User requirements breakdown structure
      3. In human terms, tasks herein could be completed by anyone (either accessing as a community individual or accessing as a member of the community InterSystem team)
   C. **Engineering requirements (system requirements specification)** - the technical system [engineering] requirements for materialization (their alignment):
      1. In the societal information system, this is represented by: the solution decision inquiry process, as a description from broad to specific of the functional and non-functional [technical] design of the system.
      2. Establish a system level analysis - what must the system do to satisfy user requirements.
      3. Deliver system requirements specification (in the form of, for example, a physical document, spreadsheet, database, or model of desired system illustrating the desired system by a simulation) including requirements breakdown structure.
         i. Determine functional requirements - what does the system need to be able to do. Determine performance requirements associated with functional requirements (how well does the system need to be able to perform those functions) - define performance levels
         ii. Non-functional requirements - what other characteristics are required of the system.
         iii. External interface requirements - what other systems require interface with the system
         iv. Under what conditions is the system expected to operate
         v. Verify the system performance against the requirements. Verification - to confirm system performance against specified requirements (has the system been built right for the user?). Confirming a system as aligning with its requirements. How would I confirm this requirement? Assign rationale. Do not duplicate or repeat requirements in the same document, which will result in conflict in the future.
         vi. In human terms, tasks (development

and execution) herein are completed by members of InterSystem teams.

3. **Preliminary design** - convert the logical architecture described by the engineering requirements into a secondary description of the [digital, physical] sub-systems (the upper-level architecture) that will meet the system requirements. Develop preliminary design based on chosen system concept while considering production, integration, and operational service life-cycle.
   A. Translating the concept (the logical design) into the digital and/or physical design (i.e., the logical design is translated into digital/physical design).
   B. The result (deliverable) of the preliminary design is the allocated baseline (visualizing functionality of the system now allocated to sub-system level (physical or digital "building blocks") groupings, known as configuration items as logically composed in the design/development specification.
      1. Sub-system level specifications for each configuration broken down by development item (or module)
   C. The focus shifts from the [engineering] problem domain to the [engineering] solution domain. Translating the concept into the
   D. Preliminary design verification (review) - was the study and design effort prior (the integration of information) appropriate? Will this design be technically adequate? What are the technical risk?
4. **Detailed design** (and development, prototyping) - "traditional" engineering, where sub-systems are broken down, understood, developed and integrated into existential operation.
   A. The complete engineering design specification that goes to makeup the system.
   B. Engineering of proto-types of sub-systems that make-up the system.
   C. Engineering for prototype system - that satisfy (fulfill) performance (required), reliability (required), life cycle safety and maintenance (required).
   D. Engineering for manufacturability - ensuring resource efficiency (cost affordability).
   E. Test and evaluation of prototypes - confirm system design by means of analysis of tests; design construction test; review, evaluate the expressed design's alignment with requirements.
   F. By end of this stage here is a digital, physical system.
   G. The habitat service system baseline deliverable - the societal level service baseline (a.k.a, product baseline, PBL). As the system now defined by numerous services (products, sub-systems, assemblies) as well as the materials and processes for manufacturing and construction of the total system :: materials, processes, people in time to complete tasks.
   H. Critical design evaluation (review) deliverable - the last point at which the information is in documentation form before transfer to memory and/or execution on the design. Here, the design is fully and officially accepted by all of the inquiry processes: solution (technical) and parallel, organizational value decisioning.
   I. Evaluates [solution] design in terms of readiness for production and construction; asking, Is everything a go, or not, for production and integration into operation. This evaluation process ensures the design is compatible with the societal organizational system (given what is known), or otherwise mis-alignment with a determined value orientation. This includes a detailed understanding of all of the internal and external interfaces.
5. **Operations (engineering) activities -** Operation[al design] (including, construction and production) - Produce and operate components in accordance with the detailed design specifications. Here, at the societal level, a design configuration is selected and integrated into [HSS operations] materiality (software or hardware, digital or spatial) as a 'construction' and/or 'production' (more precisely, 'service' or 'service object', for which there is InterSystem, Community/Commons, and Personal access). Components are developed, produced, and integrated in accordance with the detailed design specification in its final form, and the system is ultimately construction and operational (as an 'HS service' or 'HS service object').
   A. Formal qualification evaluation (review) - the user accepts the system from the InterSystem Team.
   B. All activities beyond system development.
   C. Post-development stage has all activities, but systems engineering is necessary in supporting user.
   D. Solve unanticipated issues where resolution is necessary to ensure the continued usage of the system.
   E. Testing and evaluation of system in its operational environment.
   F. Acceptance stage (because user accepts the digital/physical design that was the translation of the system concept.

1. After the development of the system where activities production, operation, deployment, system support, etc. are accomplished during useful life of the system. The system is doing what it is supposed to do.

6. **Utilization of service** (a.k.a., application, post-development, operations) - operational use and system support through engineering as a deployed or transitioned system. System support (life-cycle) - supported during utilization.
   A. Support Operations (support maintenance) - use, wherein issues become capability (and quality) gaps.
   B. The fulfillment of a need means the closing of a capability gap in the environment through systems engineering and life-cycle operation.

2.3.4 The risk-oriented engineering view

A.k.a., Safe human integration and human factors engineering.

In systems engineering, the human element is often called the human factor. Humans can come to harm, and because humans can come to harm, engineered systems should be designed while accounting for risk *to the human factor*.

The NASA Human Research Program architecture (Read: the development process) is an example of risk-oriented engineering. The human engineering research development cycle (NASA Human Research Program) is:

1. **Evidence** - Reviews of the accumulated evidence from human records, habitat operations, and research findings are compiled into NASA Human Research Program Evidence Reports. These findings provide the basis for identifying the highest priority human risks (in space exploration) and are a record of the state of knowledge for each risk in the program requirements document (PRD). The Evidence Reports are available to the scientific community and general public [https://humanresearchroadmap.nasa.gov/evidence/]. The Evidence Reports receive outside independent review and are updated as needed. If new evidence indicates that a risk should be retired or that a new risk should be added, the Human Research Program (HRP) will, after thorough review with the HSRB, take the appropriate action to modify the PRD and update the Evidence Reports accordingly.
2. **Risks** - Identifies relevant risks, including risks to the health and human performance of the exploration program based on current evidence. Each risk is assigned a risk rating as a tool to communicate to the seriousness of a risk to crew health and performance when applied to the mission* architecture and/or mission characteristics defined for each Design Reference Mission (DRM). The PRD, however, does not establish priority for the risks.
3. **Gaps** - Identifies gaps in knowledge about the risk and the ability to mitigate the risk. The degree of uncertainty in understanding the likelihood, consequence and/or timeframe of a particular risk as well as its criticality to the mission(s) are the major factors that drive the priority of the research gaps listed in the Integrated Research Plan (IRP). Gaps should represent the critical questions that need to be answered in order to significantly reduce the risk. Gaps could change over time based on research progress, current evidence, and mission planning scenarios. In some cases, a gap can address multiple risks.
4. **Tasks** - Defines the tasks that will provide the deliverables required to fill the gaps. Tasks are listed in the Integrated Research Plan (IRP). The IRP describes a plan of research that addresses both human physiology, human performance and the interconnected system of the human and spacecraft in a highly integrated manner. The HRP Elements identify specific research tasks that are targeted at better characterizing a risk or developing mitigation capabilities to reduce the risk to an acceptable level.
5. **Deliverables** - Each task or progression of tasks is designed to ultimately culminate in deliverables or products that range from risk characterization to prototype technology or countermeasures.

** A 'mission' is a type of 'project' with a human factor.*

Human Research Program (HRP) deliverables are generally:

1. **Knowledge** - deliverables that add to the body of knowledge regarding the risk or concern.
2. **Countermeasures** – preventative and treatment actions taken to address a risk,.
3. **Technology development** - hardware and software that enable risk monitoring, prevention or treatment.
4. **Operational protocols** - operational procedures and methods that define a technique or process for mitigation of the risk.
5. **Guidelines, requirements, and standards** – information that defines the acceptable levels of risk. Information generated by HRP that can inform the status of the risk and anticipated mitigations are documented in the HSRB Risk Summary.

A socio-technical [human] research program may have

the following deliverable categories:

1. **Requirement** or **guideline** - The "Requirement or Guideline" deliverable is chosen when a task will result in information that is relevant to a requirement (or requirements set) or guideline associated with a higher decision set.
2. **Technology** or **tool** - The "Technology or Tool" deliverable covers a broad spectrum of developments that includes hardware, software, systems solutions, new processes, new systems and machines (inventions), new methods and procedures (innovative methods), collaborative design tools, databases, computational models, or systems simulations.
3. **Countermeasure** - A "Countermeasure" deliverable is a specific protocol that is developed and validated to prevent or reduce the likelihood or consequence of a risk [of acceptable level]. Countermeasures may be medical, physical, or operational entities, such as a pharmaceutical or nutritional supplement, hardware or software (prototype and fully integrated), or specific exercise/training, entrainment routines, respectively. A countermeasure deliverable is usually specific and extensive enough to require validation in habitat service operation.
4. **Standard** - A working group integration of all feedback and discovery to which operation conforms by threshold (i.e., by degree). Discovery workgroups may result in a recommendation for a new or updated standard. Standards working groups integrate discoveries into the next iteration of the societal-habitat system, through project-engineering and cooperation-coordination.

2.3.5 The asset coordination life-cycle view of engineering

The asset management lifecycle (a.k.a., asset lifecycle management, ALM) is a process for coordinating ("managing") the usage (and maintenance) of a support service 'asset' (or 'object') throughout its lifetime (or period of service). Each assets lifecycle is defined by a series of stages:

1. **Procurement/access coordination:**
   A. Set requirements for purchasing the asset
      1. Based on inventory, consumption, and labor rates.
         i. Make purchase order.
            1. Track purchase until delivery.
2. **Inbound/outbound service coordination:**
   A. Inbound services.
      1. Receive shipment.
      2. Unpackage shipment.
      3. Reconcile shipment with purchase order (itemized checklist) to ensure accuracy.
      4. Tag asset for tracking in system.
   B. Outbound services.
      1. Package asset (for deliver to end location).
3. **Inventory coordination:**
   A. Storage - Assets not yet ready to be delivered to an end location are deposited in an inventory (organized for streamlined storage and retrieval).
      1. Inventory cycle counts (inventory surveys) ensure that min/max levels are maintained and accurately reflected in the asset management database.
4. **Deployment coordination:**
   A. Request/demand - an asset is requested for use.
   B. Retrieval - an asset is retrieved from inventory
   C. Provisioning ("to make something available or ready to use") - final configuration of system for specific use.
   D. Access - by user for usage.
5. **Re-assignment coordination:**
   A. Return access -
      1. Still has useful life?
      2. Does not still have useful life?
      3. Event.
         i. Lost.
         ii. Return to inventory.
         iii. Return for de-composition.
         iv. Return for repair.

More simply, the asset lifecycle consists of

1. **Asset objective** - communicate and plan asset
2. **Asset model** - design asset.
3. **Asset construction** - code/build and test of asset.
4. **Asset deployment** - integration and operation
5. **Asset usage** - the asset is used by the user and maintained/operated by asset technicians.
6. **Asset return** - the return of the asset to inventory, or for de-cycling.

2.3.6 Asset life-cycle software

Current asset lifecycle software solutions include:

1. Autodesk fusion product lifecycle.
2. Service life-cycle coordination (application, asset management lifecycle).

3 [Engineering] Life-cycle processes

The socio-technical engineering process is a multi-stage method that results in a highly predictable societal design materialization.

3.1 Initiation and planning stage

The output work products for this stage are:

1. Issue articulation [initial].
2. Project coordination plan [initial].
3. Project charter [initial].
4. Maintenance plan [initial].
5. Configuration coordination plan [initial].

3.2 Requirements definition stage

The requirements definition phase starts with establishing a functional baseline from which to do future work.

3.2.1 Establish functional baseline

A.k.a., System requirements baseline.

The functional baseline is the main technical work product of the Requirements Definition stage of an engineering project. The system requirements are baselined after the Project Team's formal approval of the Requirements Specification. Once the requirements are baselined, any changes to the requirements must be coordinated under change control procedures.

Clarification: *To be "baselined" means to have been formally determined (or, selected).*

The output work products for this stage are:

1. Project coordination plan [revised].
2. Requirements specification [initial].
3. Requirements traceability matrix [initial].
4. Maintenance plan [revised].
5. Configuration coordination plan [revised].
6. Organizational continuity plan [revised].
7. Data dictionary [revised].

3.3 Functional design stage

During the functional design stage, the overall structure of the product is defined from a functional viewpoint. The goal of this stage is to define and document the functions of the product to the extent necessary to obtain the system owner and users understanding and approval and to the level of detail necessary to build the system design.

The deliverable of the functional design stage is the Functional Design [Document].

The high-level activities are presented in the sections listed below.

1. Determine system structure.
2. Design content of system inputs and outputs.
3. Design user interface.
4. Design system interfaces.
5. Design system security controls.
6. Build logical model.
7. Build data model.
8. Develop functional design.
9. Select system architecture.

The output work products for this stage are:

1. Project coordination plan [revised].
2. Functional design document [final].
3. Maintenance plan [revised].
4. Requirements specification [final].
5. Requirements traceability matrix [revised].
6. Configuration coordination plan [revised].
7. Organizational continuity plan [revised].
8. Data dictionary [final].

3.3.1 The functional design specification

The functional design process maps the "what to do" of the Requirements Specification into the "how to do it" of the design specifications. The functional design describes the logical system flow, data organization, system inputs and outputs, processing rules, and operational characteristics of the product from the user's point of view. The functional design is not concerned with the software or hardware that will support the operation of the product or the physical organization of the data or the programs that will accept the input data, execute the processing rules, and produce the required output. The focus is on the functions and structure of the components that comprise the product. The functional design describes how the product will be structured to satisfy the requirements identified in the Requirements Specification. It is a description of the structure, components, interfaces, and data necessary before development can begin.

The functional design is a model or representation of the system that is used primarily for communicating design information to facilitate analysis, planning, and coding decisions. It represents a partitioning of the system into design entities and describes the important properties and relationships among those entities. Design descriptions may be produced as documents, graphic representations, formal design languages, and records in a database.

Within the functional design, the design entities can be organized and presented in any number of ways. The goal of this activity (Read: develop the functional

design) is to compile the design entities and their associated attributes in a manner that facilitates the access of design information from various viewpoints (e.g., project coordination, engineering development, quality assurance, and testing). Also, the design entities and their attributes must be described in terms that are understandable to the system users.

Prototyping of system functions can be helpful in communicating the design specifications to the system users. Prototypes can be used to simulate one function, a module, or the entire product. Prototyping is also useful in the transition from the functional design to the system design.

3.3.2 Determine system structure

A hierarchical approach is useful for determining the structure and components of the system. System decomposition is one hierarchical approach that divides the system into different levels of abstraction. Decomposition is an iterative process that continues until single purpose components (i.e., design entities or objects) can be identified. Decomposition is used to understand how the product will be structured, and the purpose and function of each entity or object.

The goal of the decomposition is to create a highly cohesive design. A design exhibits a high degree of cohesion if each design entity in the system unit is essential for that unit to achieve its purpose.

Several reliable methods exist for performing system decomposition. Select a method that enables the design of simple, independent entities. Functional design and object-oriented design are two common approaches to decomposition. These approaches are not mutually exclusive. Each may be applicable at different times in the design process.

3.3.2.1 Tasks to determine system structure

The system decomposition activity includes the following tasks.

1. Identify design entities.
2. Identify design dependencies.

3.3.3 Identify design entities

Design entities result from a decomposition of the system requirements. A design entity is an element (or object) of a design that is structurally and functionally distinct from other elements and is separately named and referenced. The number and type of entities required to partition a design are dependent on a number of factors, such as the complexity of the product, the design method used, and the development environment. The objective of design entities is to divide the product into separate components that can be coded, implemented, changed, and tested with minimal effect on other entities.

3.3.3.1 Attributes of design entities

A design entity attribute is a characteristic or property of a design entity. It provides a statement of fact about an entity. The following are common attributes that should be considered for each design entity.

1. Assign a unique name to each entity.
2. Classify each entity into a specific type. The type may describe the nature of the entity, such as a sub-program or module; or a class of entities dealing with a particular type of information.
3. Describe the purpose or rationale for each entity. Include the specific functional and performance requirements for which the entity was created.
4. Describe the function to be performed by each entity. Include the transformation applied to inputs by the entity to produce the desired output.
5. Identify all of the external resources that are needed by an entity to perform its function.
6. Specify the processing rules each entity will follow to achieve its function. Include the algorithm used by the entity to perform a specific task and contingency actions in case expected processing events do not occur.
7. Describe the data elements internal to each entity. Include information such as the method of representation, format, and the initial and acceptable values of internal data. This description may be provided in the data dictionary.

3.3.4 Identify design dependencies

Design dependencies describe the relationships or interactions between design entities at the module, process, and data levels. These interactions may involve the initiation, order of execution, data sharing, creation, duplication, use, storage, or destruction of entities.

Identify the dependent entities of the system design, describe their coupling, and identify the resources required for the entities to perform their function. Also define the strategies for interactions among design entities and provide the information needed to perceive how, why, where, and at what level actions occur.

Dependency descriptions should provide an overall picture of how the product will work. Data flow diagrams, structure charts, and transaction diagrams are useful for showing the relationship among design entities.

The dependency descriptions may be useful in producing the system integration plan by identifying the entities that are needed by other entities and that must be developed first. Dependency descriptions can also be used to aid in the production of integration test cases.

3.3.5 Design content of system inputs and outputs

Design the content and format for each of the product

inputs and outputs based on the system input and output requirements identified during the Requirements Definition Stage. Involve the system users in the design process to make certain that their needs and expectations are being met.

Document the design for the system inputs and outputs in accordance with the project design standards. Discuss the designs with the system owner and users and submit completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document.

3.3.6 Design user interface

Design a user interface that is appropriate for the users, content, and operating environment for the product. Determine interface levels for all categories of users. For interactive user environments, prototype the user interface. Arrange for users to experiment with the prototypes so that design weaknesses in the interface can be identified and resolved early. Use prototypes to gain user acceptance of the interface.

3.3.7 Design system interface

Develop a design depicting how the product will interface with other systems based on the system interface requirements identified in the Requirements Definition Stage. Submit the applicable interface designs for review by the system owner or system administrator for each system that will interface with the product. Any incompatibilities with the interfaces will be identified early in the design process and corrective actions can be initiated to assure each interface is properly designed and coded.

3.3.8 Design system controls

Design the access (security) controls that will be incorporated into the product based on the access requirements identified during the Requirements Definition Stage.

3.3.8.1 Design system controls procedure

Use the following procedure to implement the design process.

1. Identify the users and organizations that will have access to the product. Indicate what access restrictions they will have. All persons in a work area may not have the same access level. Controls should be implemented to assure that materials and systems requiring protection are not accessed by unauthorized individuals.
2. Identify controls for the product, such as the user identification code for system access and the network access code for the network on which the product will reside.
3. Identify whether access restrictions will be applied at the system, subsystem, transaction, record, or data element levels. Sensitive information must be protected in accordance with directives.
4. Identify physical safeguards required to protect hardware, software, or information from natural hazards and malicious acts.
5. Identify communications access (security) requirements.

3.3.9 Build logical model

The logical model defines the flow of data through the system and determines a logically consistent structure for the system. Each module that defines a function is identified, interfaces between modules are established, and design constraints and limitations are described. The focus of the logical model is on the real-world problem or need to be solved by the product.

A logical model has the following characteristics:

1. Describes the final sources and destinations of data and control flows crossing the system boundary rather than intermediate handlers of the flows.
2. Describes the net transfer of data across the system boundary rather than the details of the data transfer.
3. Provides for data stores only when required by an externally imposed time delay.

When building a logical model, the organization of the model should follow the natural organization of the product's subject matter. The names given to the components of the model should be specific. The connections among the components of the model should be as simple as possible.

The logical model should be documented in user terminology and contain sufficient detail to obtain the system owner's and users' understanding and approval. Use data flow diagrams to show the levels of detail necessary to reach a clear, complete picture of the product processes, data flow, and data stores.

Maintain the logical model and data flow diagrams for incorporation into the Functional Design Document. Keep the logical model and diagrams up-to-date. They will serve as a resource for planning enhancements during maintenance, particularly for enhancements involving new functions.

3.3.10 Build data model

A data model is a representation of a collection of data objects and the relationships among these objects (i.e., representation of information about a form or a process).

The data model is used to provide the following functions:

1. Transform the sense entities into data entities.

2. Transform the socio-technical rules into data relationships.
3. Resolve the many-to many relationships as intersecting data entities.
4. Determine a unique identifier (key) for each data entity.
5. Add the attributes (facts) for each data entity.
6. Document the integrity rules required in the model.
7. Determine the data accesses (navigation) of the model.

The data dictionary is developed in this stage. Its purpose it to catalogue every known data element used in the user's work and every system-generated data element. Data elements are documented in detail to include attributes, known constraints, input sources, output destinations, and known formats.

The data dictionary can serve as a central repository of information for both developers and end users. The dictionary can include business rules, processing statistics, and cross-referencing information for multiple vendor environments.

To expand the data dictionary, define, analyze, and complete data definitions using the following steps.

1. Identify data needs associated with various system features.
2. Match (verify) data needs with the data dictionary.
3. Match the data dictionary with specific data structures.
4. Create data record layouts.
5. Ensure that all data can be maintained through add, change, or delete functions.

3.3.11 Develop functional design

Major work products are the Functional Design and the revised Requirements Traceability Matrix. Each requirement identified in the Requirements Specification must be traceable to one or more design entities. This traceability ensures that the product will satisfy all of the requirements and will not include inappropriate or extraneous functionality. Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the functional design to the requirements.

It is relevant to note here that technical design reviews occur during the system engineering lifecycle. These reviews can be supported by peer reviews, which are deeper technical reviews by technical experts in the subject matter to be reviewed. Design reviews are generally conducted when the system under development meets a milestone or level of development/ construction during the product design through to realization process. Formalized design reviews have entry and exit criteria, and/or acceptance criteria.

The following tasks are involved in developing the functional design:

1. Develop Functional Design Document
2. Conduct Functional Design Review

3.3.11.1 Develop functional design document

The Functional Design Document defines the functions of the system in user terminology and provides a firm foundation for the development of the system design. The Functional Design Document should be written from the system users' perspective. This document provides the users with an opportunity to review and provide input to the product design before system design work is completed.

3.3.11.2 Conduct functional design review (technical design review)

The Functional Design Review is a formal technical review of the basic design approach. The primary goal of the Functional Design Review is to demonstrate the ability of the system design to satisfy the project requirements. The review may be a series of presentations by the project team to the system users, functional area Team members (a.k.a., points-of-contact). Vendors may be invited to participate in the Functional Design Review when an off-the-shelf software product or hardware item is being considered for the system architecture.

The work product is the Functional Design Document. The review of this document will result in one of the following outcomes:

1. **Selection (a.k.a., approval)** - indicates that the functional design is satisfactorily completed.
2. **Hold selection (a.k.a., hold approval, contingent approval)** - indicates that the functional design is not considered accomplished until the satisfactory completion of identified action items.
3. **Non-selection (a.k.a., disapproval)** - indicates that the functional design is inadequate. Another Functional Design Review is required, once specified changes to the functional design are completed.

Conduct the Functional Design Review to perform the following verifications:

1. Evaluate the progress, technical adequacy, and risk mitigation of the selected design approach. Determine whether the design approach is being followed by the project team.
2. Evaluate the progress, technical adequacy, and risk mitigation of the selected test approach. Review the following items:
   A. System test requirements from the requirements specification document.

B. Organization and responsibilities of group conducting tests.
C. Planned format, content, and distribution of test reports.
D. Planned resolution of problems and errors identified during testing.
E. Retest procedures.
F. Change control and configuration management of test items.
G. Special test tools not required as deliverables.

3. Evaluate the techniques to be used to meet quality assurance requirements.
4. Establish the existence and compatibility of the physical and functional interfaces.
5. Determine whether the functional design embodies all of the product requirements.
6. Verify that the design represents a system that can meet the functional, data, and interface requirements.
7. Demonstrate any rapid design prototypes used to make design decisions.
8. Identify potential high risk areas in the design and any requirements changes that could reduce risk.
9. Review to assure that consideration has been given to optimizing the maintainability and maintenance aspects of the product.

The following items should be considered for review and evaluation during the Functional Design Review:

1. **Functional flows**: Indicate how the system functional flows map the software and interface requirements to the individual high-level components of the product.
2. **Storage allocation data**: Describe the manner in which available storage is allocated to individual components. Timing, sequencing requirements, and relevant equipment constraints used in determining the allocation should be included.
3. **Control functions**: Describe the executive control and start/recovery features of the product.
4. **Component structure**: Describe the high-level structure of the product, the reasons for choosing the components, the development technique that will be used within the constraints of available computer resources, and any support programs that will be required in order to develop and maintain the product and allocated data storage.
5. **Security**: Identify the security requirements and provide a description of the techniques to be used for implementing and maintaining security within the product.
6. **Information systems engineering facilities**: Describe the availability, adequacy, and planned utilization of the information systems engineering facilities including both Government-provided and commercially available facilities.
7. **Information systems engineering facility versus the operational system**: Describe any unique design features that exist in the functional design in order to allow use within the information systems engineering facility that will not exist in the operational product. Provide information on the design of support programs not explicitly required for the operational system that will be generated to assist in the development of the product.
8. **Development tools**: Describe any special tools (e.g., simulation, data reduction, or utility tools) that are not deliverables, but are planned for use during systems development.
9. **Test tools**: Describe any special test systems, test data, data reduction tools, test computer software, or calibration and diagnostic software that are not deliverables, but are planned for use during development.
10. **Commercial resources**: Describe commercially available computer resources, including any optional capabilities (e.g., special features, interface units, special instructions, controls, formats). Identify any limitations of commercially available equipment (e.g., failure to meet user interface, safety, and maintainability requirements) and identify any deficiencies.
11. **Existing documentation**: Maintain a file and have available for review any existing documentation supporting the use of commercially available computer resources.
12. **Support resources**: Describe the resources necessary to support the product during engineering, installation, and operational state (e.g., operational and support hardware and software personnel, special skills, human factors, configuration management, testing support, documentation, and facilities/space management).
13. **Standards**: Describe any standards or guidelines that must be followed.
14. **Operation and support documentation**: Describe the documentation that will be produced to support the operation and maintenance of the product.

3.3.12 Select system architecture

When the system architecture for the product has not been predetermined by the existing environment of the system users, evaluate system architecture alternatives to determine which one best satisfies the project requirements. Select the specific design based on the pre-determined value conditions.

The following tasks are involved in selecting a system architecture:

1. Evaluate system architecture alternatives
2. Select system architecture

3.3.12.1 Evaluate system architecture alternatives

Consider system architecture alternatives within the organizations architecture guidelines and standards conditions that enable the project objectives and requirements to be achieved.

The following procedure provides one approach for evaluating the architecture alternatives:

1. Conduct an analysis to determine the most effective and conditionally aligned alternative.
2. Create and evaluate a data flow diagram for each alternative.
3. Identify how users would interact with the features associated with each alternative.
4. Create a list of the risks associated with each alternative and develop a plan for mitigating each risk.
5. Compare the performance capabilities of each alternative.
6. Follow the societal-level decision system.

3.4 System design stage

The goal of this stage is to translate the user-oriented functional design specifications into a set of technical, realization-oriented system design specifications; and to design the data structure and processes to the level of detail necessary to plan and execute the Construction and Implementation Stages. General module specifications should be produced to define what each module is to do, but not how the module is to be coded. Effort focuses on specifying individual routines and data structures while holding constant the structure and interfaces developed in the previous stage. Each module and data structure is considered individually during detailed design with emphasis placed on the description of internal and procedural details. The primary work product of this stage is a system design that provides a specification (blueprint) for the materialization (i.e., [en] coding) of individual modules and elements.

The following items provide input to this stage:

1. Functional design.
2. Maintenance plan.
3. Requirements specification.
4. Requirements traceability matrix.
5. Software configuration management plan.
6. Project coordination plan.
7. Access plan.
8. Data dictionary

The high-level activities for this stage are:

1. Design specifications for modules.
2. Design physical model and database structure.
3. Develop integration test considerations.
4. Develop system test considerations.
5. Develop conversion plan.
6. Develop system design.

The output work products for this stage are:

1. Project coordination plan [revised].
2. Conversion plan [initial].
3. Maintenance plan [revised].
4. Requirements traceability matrix [revised].
5. Configuration management plan [final].
6. System design document [final].
7. Test plan [initial].
8. Test type approach and reports [initial].
9. Test cases [initial].

3.4.1 System design

The system design is the main technical work product of the System Design Stage. The system design translates requirements into precise descriptions of the components, interfaces, and data necessary before coding and testing can begin. It is a blueprint for the Construction Stage based on the structure and data model established in the Functional Design Stage.

Once the system design is baselined, any changes to the design must be managed under change control procedures. Approved changes must be incorporated into the System Design Document.

It is important for the system users to understand that some changes to the baselined system design may affect the project scope and therefore can change the project resources, schedule, etc. It is the responsibility of the project coordinator and team to identify system user requested changes that would result in a change of project scope; evaluate the potential impact to the project elements (resources, schedule, etc.); and notify the system user of the project planning revisions that will be required to accommodate their change requests.

3.4.2 Design specifications for modules

Expand the functional design to account for each major action that must be performed and each data object to be managed. Detail the design to a level such that each sub-system represents a function that a developer will be able to develop.

The following procedure facilitates in designing the module specifications:

1. Identify a structure for each action needed to meet

each function or requirement in the Requirements Specification and the data dictionary.
2. Identify any routines and structures that may be available as reusable objects.
3. Identify structures that must be designed and developed (custom-built). Assign a name to each structure and object that is functionally meaningful. Identify the system features that will be supported by each structure.
4. Specify each structure interface. Update the data dictionary to reflect all program and object interfaces changed while evolving from the functional to the system design.
5. Define and design significant attributes of the structures to be custom-built.
6. Expand the structure interfaces to include control items needed for design validity (e.g., error and status indicators).
7. Combine similar structures and objects. Group the design entities into modules based on closely knit functional relationships. Formulate identification labels for these modules.
8. Show dependencies between data structures and physical structures.
9. Change the design to eliminate features that reduce maintainability or reusability (i.e., minimize coupling between programs and maximize the cohesion of programs).

Document the system design primarily in the form of diagrams. Supplement each diagram with text that summarizes the function (or data) and highlights important performance and design issues.

When using structured design methods, the design diagrams should:

1. Depict the product as a top-down set of diagrams showing the control hierarchy of all programs to be implemented.
2. Define the function of each structure.
3. Identify data and control interfaces between programs.
4. Specify files, records, and global data accessed by each program.
5. When using object-oriented or data-centered design methods, the design diagrams should:
6. Show the data objects to be managed by the product.
7. Specify the program functions to be included within each object.
8. Identify functional interfaces between objects.
9. Specify files and records comprising each object.
10. Identify relationships between data files.

Standards for specifications may be provided by government agencies, standards organizations (SAE, AWS, NIST, ASTM, ISO, CEN, US DoD, etc.), trade associations, corporations, and others.

The following British standards apply to specifications:

- BS 7373-1:2001 Guide to the preparation of specifications [4]
- BS 7373-2:2001 Product specifications. Guide to identifying criteria for a product specification and to declaring product conformity [5]
- BS 7373-3:2005, Product specifications. Guide to identifying criteria for specifying a service offering
- The following NIST standards apply [nist.gov]:
- IEEE P7001 - Transparency of autonomous systems
- IEEE P7003 - Algorithmic bias considerations
- IEEE P7007 - Ontological standard for ethically driven robotics and automation systems
- IEEE P7008 - Standard for ethically driven nudging for robotic, intelligent and autonomous systems
- IEEE P7009 - Standard for fail-safe design of autonomous and semi-autonomous systems
- IEEE P7010 - Well-being metrics standard for ethical artificial intelligence and autonomous systems

3.4.3 Design physical model and database structure

The physical model is a description of the dynamics, data transformation, and data storage requirements of the system. The physical model maps the logical model created during the Functional Design Stage to a specific technical solution.

3.4.4 Develop conversion plan

A.k.a., Develop transition plan.

If the product will replace an existing system, then develop a Conversion Plan. The major elements of the Conversion Plan are to develop conversion procedures, outline the installation of new and converted structures, coordinate the development of structural-conversion, and plan the implementation of the conversion procedures.

System conversion should include a confirmation of file integrity. Determine what the output in the new system should be compared with the current system, and ensure that the files are synchronized. The objective of file conversion is new files that are complete, accurate and ready to use.

Many factors influence conversion, such as the design of the current and new systems and the processes for input, storage, and output. Understanding the structures function in the old system and determining if the function will be the same or different in the new system is of major importance to the Conversion Plan.

The structure of the system to be converted can limit the development of the system and affect the choice of structure.

Consider the following factors during the development of the Conversion Plan:

1. Determine if any portion of the conversion process should be performed manually.
2. Determine whether parallel runs of the old and new systems will be necessary during the conversion process.
3. Understanding the function of the structure in the old system and determining if the use will be the same or different in the new system is important.
4. The order that information is processed in the two systems influences the conversion process.
5. User work and delivery schedules, timeframes for reports and end-of-year procedures, and the criticality of the data help determine when conversion should be scheduled.
6. Determine whether availability and use should be limited during the conversion.
7. Plan for the disposition of obsolete or unused structure that is not converted.

3.4.5 Develop system design

Major work products include the System Design Document and the updated Requirements Traceability Matrix. Each requirement identified in the Requirements Specification must be traceable to one or more design entities. This traceability ensures that the product will satisfy all of the requirements and will not include inappropriate or extraneous functionality. Revise the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the system design to the requirements.

The following tasks are involved in developing the system design.

1. Develop System Design Document.
2. Conduct System Design Review.

3.4.5.1 Develop system design document

The System Design Document records the results of the system design process and describes how the system will be structured to satisfy the requirements identified in the Requirements Specification. The System Design Document is a translation of the requirements into a description of the structure, components, interfaces, and data necessary to support the construction process.

3.4.5.2 Conduct system design review (technical design review)

The System Design Review is a formal technical review of the system design. The purpose of the review is to demonstrate to the system users that the system design can be implemented on the selected platform and accounts for all requirements and accommodates all design constraints (e.g., performance, resource, and reliability requirements). The design review should include a review of the validity of algorithms needed to perform critical functions.

3.5 Construction stage

The goal of this stage is to translate the set of technical system design specifications into a language the constructor can understand and execute. Construction may involves materializing, coding, validation and unit testing by a developer. Plans are developed for the installation of the operating environment hardware and software. A training program is designed and a Training Plan that describes the system is produced.

The activities in this stage result in the transformation of the system design into the first complete executable (operatable) representation of the product.

The high-level activities for this stage are:

1. Establish development environment
2. Develop programs
3. Conduct unit testing
4. Establish development baselines
5. Plan transition to operational status
6. Generate operating documentation
7. Develop training plan
8. Develop installation plan

The output work products for this stage are:

1. Project coordination plan [revised]
2. Maintenance plan [revised]
3. Requirements traceability matrix [revised]
4. Conversion plan [revised]
5. Test type approach and reports [revised]
6. Test cases [revised]
7. Transition plan [initial]
8. Installation plan [initial]
9. Training plan [initial]
10. Operating documentation [initial]
    A. Users manual
    B. Developer's reference manual
11. System units and modules [initial]

3.5.1 Establish Development Environment

Establishing the development environment involves assembling and installing the hardware, software,

equipment, databases, and other items required to support the construction effort.

Before being integrated into or used to support the product, vendor products should be tested to verify that the product satisfies the following objectives:

1. The product performs as advertised/specified.
2. The product's performance is acceptable and predictable in the target environment.
3. The product fully or partially satisfies the project requirements.
4. The product is compatible with the project team's other hardware and software tools.

Time should be planned for the project team to become familiar with new products. Ensure that the project team members who will use the hardware or software obtain proper training. This may involve attendance at formal training sessions conducted by the vendor or the services of a consultant to provide in-house training.

3.5.2 Conduct unit testing

Unit testing is used to verify the input and output for each module. Successful testing indicates the validity of the function or sub-function performed by the module and shows traceability to the design. During unit testing, each module is tested individually and the module interface is verified for consistency with the design specification. All important processing paths through the module are tested for expected results. All error handling paths are also tested.

Unit testing is driven by test cases and test data that are designed to verify requirements, and to exercise all program functions, edits, in-bound and out-bound values, and error conditions identified in the program specifications. If timing is an important characteristic of the module, tests should be generated that measure time critical paths in average and worst-case situations.

Plan and document the inputs and expected outputs for all test cases in advance of the tests. Log all test results. Analyze and correct all errors and retest the unit using the scenarios defined in the test cases. Repeat testing until all errors have been corrected.

While unit testing is generally considered the responsibility of the developer, the project coordinator or lead developer should be aware of the unit test results.

Completion of unit testing for a component signifies internal project delivery of a component or module for integration with other components.

3.5.3 Establish development baseline

A development baseline is an approved "build" of the product. A build can be a single component or a combination of components. The first development baseline is established after the first build is completed, tested, and approved by the project manager or lead developer. Subsequent versions of a development baseline should also be approved. The approved development baseline for one build supersedes that for its predecessor build.

Conduct internal build tests such as regression, functional, performance, and reliability. Regression tests are designed to verify that capabilities in earlier builds continue to work correctly in subsequent builds. Functional tests focus on verifying that the build meets its functional and data requirements and correctly generates each expected display and report. Performance and reliability tests are used to identify the performance and reliability thresholds of each build.

Once the first development baseline is established, any changes to the baseline must be managed under the change control procedures. Approved changes to a development baseline must be incorporated into the next build of the product and revisions made to the affected work products (e.g., Requirements Specification, System Design Document, and Program Specifications).

Document the internal build test procedures and results. Identify errors and describe the corrective action that was taken. Place a copy of the internal build test materials in the Project Test File.

Maintain configuration control logs and records as required. Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage.

3.5.4 Plan transition to operational status

Successful transition from acceptance testing to full operational use of the product depends on planning the transition long before the product is installed in its operational environment. In planning for the transition, quantify the operational needs associated with the product and describe the procedures that will be used to perform the transition.

Rely on experience and data gathered from previous, similar projects to define these needs. Develop a Transition Plan that describes the detailed plans, procedures, and schedules that will guide the transition process. Coordinate development of the plan with the operational and maintenance personnel. The following issues should be considered in the preparation of a Transition Plan:

1. Develop detailed operational scenarios to describe the functions to be performed by the operational support staff, maintenance staff, and users.
2. Document the release process. If development is incremental, define the particular process, schedule, and acceptance criteria for each release.
3. Describe the development or migration of data, including the transfer or reconstruction of historic data. Schedule ample time for the system owner and user to review the content of reconstructed or migrated data files to reduce the chance of errors

or omissions.
4. Specify problem identification and resolution procedures for the operational product.
5. Define the configuration management procedures that will be used for the operational product. Ideally, the methods defined in the Software Configuration Management Plan that were employed during product development can continue to be used for the operational product.
6. Define the scope and nature of support that will be provided by the project team during the transition period.
7. Specify the organizations and individuals who will be responsible for each transition activity, ensuring that responsibility for the product by the operations and maintenance personnel increases progressively.
8. Identify products and support services that will be needed for day-to-day operations or that will enhance operational effectiveness.

3.5.5 Generate operating documentation

Plan, organize, and write the operating documentation that describes the functions and features of the product from the users point-of-view. The different ways that users (including system administration and maintenance personnel) will interact with the product must be considered. The needs of the users should dictate the document presentation style and level of detail. Responsibilities for changing and maintaining the documents should be described in each document.

The following are typical operating documents for a large project:

1. Users manual/online help screens.
2. Developer's reference manual.
3. Intersystem team manual (a.K.A., Systems administration manual).
   A. Database administration manual.
4. Operations manual.

It is recommended that a technical writer be involved in the generation of all operating documents. A technical writer works closely with the project team to ensure that documents are grammatically correct; comply with applicable standards; and are consistent, readable, and logical.

Use the following procedure to develop the operating documentation.

1. Identify the operating documents that need to be developed. Determine if any of the documents can be combined or delivered as multiple volumes.
2. Determine whether the documents should be provided as printed material, standalone electronic files, online documentation accessed through the product, or a combination.
3. Determine the best presentation method or combination of methods required for each of the documents, such as a traditional manual, quick reference guide or card, or online help.
4. Identify all of the features of the user interface and the tasks users will perform.
5. Identify the users' needs and experience levels to determine:
   A. The amount of user interaction, level of interaction, and whether the interaction is direct or indirect.
   B. The appropriate level of detail (e.g., the Users Manual should not contain highly technical terms and explanations that may confuse or frustrate a user).
6. Determine the document content and organization based on whether the document will be used more as an instructional tool or a reference guide.
7. Develop descriptions of each function and feature of the product and organize the information to facilitate quick, random access.
8. Provide appropriate illustrations and examples to enhance clarity and understanding.
9. Establish a schedule for the documents to be reviewed after the product goes into production. Operating documents must be kept up-to-date as long as the product remains in production.

The following tasks describe the minimum requirements for operating documentation.

1. Produce Users Manual
2. Produce Developer's Reference Manual

3.5.5.1 Produce users manual

The Users Manual provides detailed information users need to access, navigate through, and operate the product. Users rely on the Users Manual to learn about the product or to refresh their memory about specific functions. A Users Manual that is organized functionally so that the information is presented the same way the product works helps users understand the flow of menus and options to reach the desired functions.

Different categories of users may require different types of information. A modular approach to developing the Users Manual to accommodate the needs of different types of users eliminates duplication and minimizes the potential for error or omission during an amendment or update. For example, separate general information that applies to all users from the special information that applies to selected users such as system administrators or database administrators. The special information can be presented in appendixes or supplements that are

only provided to the users who need the information.

Write the draft Users Manual in clear, non-technical terminology that is oriented to the experience levels and needs of the user(s).

For very small projects, a quick reference guide or card may be more appropriate than a full-scale Users Manual.

For projects of any size, a quick reference card may be developed as a supplement to more detailed user documentation.

The following are typical features of a users manual.

1. Overview information on the history and background of the project and the architecture, operating environment, and current version or release of the product.
2. Instructions for how to install, setup, or access the product.
3. Complete coverage of all functions, presented in a logical, hierarchical order.
4. Accurate pictures of screens and reports, ideally with data values shown, so the user can easily relate to examples.
5. In-depth examples and explanations of the areas of the product that are most difficult to understand.
6. Clear delineation of which features are accessible only to specific users.
7. Instructions on accessing and using online help features.
8. Procedures for data entry.
9. Descriptions of error conditions, explanations of error messages, and instructions for correcting problems and returning to the function being performed when the error occurred.
10. Instructions for performing queries and generating reports.
11. Who to contact for help or further information.

3.5.5.2 Produce developer's reference manual

The Developer's Reference Manual contains information about program development used by the maintenance staff to maintain the programs, databases, interfaces, and operating environment. The Developer's Reference Manual should provide an overall conceptual understanding of how the product is constructed and the details necessary to implement corrections, changes, or enhancements.

The Developer's Reference Manual describes the logic used in developing the product and the functional and system flow to help the maintenance staff understand how the programs fit together. The information should enable a developer to determine which programs may need to be modified to change a system function or to fix an error.

Use appendixes to provide detailed information that is likely to change as the product is maintained. For example, a list of program names and a synopsis of each program could be included as an appendix.

The following are typical features of a Developer's Reference Manual.

1. A description of the technical environment, including versions of the development language(s) and other proprietary software packages.
2. A brief description of the design features including descriptions of unusual conditions and constraints.
3. An overview of the architecture, program structure, and program calling hierarchy.
4. The design and coding practices and techniques used to develop the product.
5. Concise descriptions of the purpose and approach used for each program.
6. Layouts for all data structures and files used in the product.
7. Descriptions of maintenance procedures, including configuration management, program checkout, and system build routines.
8. The instructions necessary to compile, link, edit, and execute all programs.
9. Manual and automated backup procedures.
10. Error-processing features.

3.5.6 Develop training plan

A Training Plan defines the training needed to implement and operate the product successfully. The Training Plan should address the training that will be provided to the system users, and InterSystem Team Operators and Maintenance personnel. When new hardware or software is being used, affected personnel will need hands-on experience before bringing the new system (equipment and/or software) into daily operation.

Training must address both the knowledge and the skills required to operate and use the system effectively.

Complete the Training Checklist to ensure that all activities and work products are complete.

Place a copy of the initial Training Plan and completed Training Checklist in the Project File. The plan will be reviewed and updated during the Testing Stage.

Design the training to accomplish the following objectives:

1. Provide trainees with the specific knowledge and skills necessary to perform their work.
2. Prepare training materials that will sell the product as well as instruct the trainees. The training should leave the trainees with the enthusiasm and desire to use the new product.
3. Account for the knowledge and skills the trainees bring with them, and use this information as a transition to learning new material.
4. Anticipate the needs for follow-on training after

the product is fully operational, including refresher courses, advanced training, and repeats of basic courses for new personnel.
5. Build in the capability to update the training as the product evolves.

The Training Plan should address the following issues:

1. Identify the organization's training policy for meeting training needs.
2. Ensure InterSystem Teams have received orientation on the training.
3. Ensure training courses prepared at the organization level are developed and maintained according to organizational standards.
4. Ensure a procedure for required training is established and used to determine whether individuals already possess the knowledge and skills required to perform in their designated area.
5. Ensure measurements are made and used to determine the status of training activities.
6. Ensure that training activities are reviewed on a periodic basis.
7. Ensure the training is independently evaluated on a periodic basis for consistency with, and relevance to, the organization's needs.
8. Ensure the training activities and work products are reviewed and/or audited and the results are reported.
9. Ensure training records are properly maintained.

Prepare a draft Training Plan that describes the training and at a minimum addresses the following issues.

1. Identifies personnel to be trained. Review the list of trainees with the system owner and users to ensure that all personnel who should receive training have been identified.
2. Defines the overall approach to training and the required training courses.
3. Establishes the scope of the training needed for users, management, operations, and maintenance personnel.
4. Defines how and when training will be conducted. Specify instructor qualifications, learning objectives, and mastery or certification requirements (if applicable).
5. Identifies any skill areas for which certification is necessary or desirable. Tailor the training to the certification requirements.
6. Establishes a preliminary schedule for the training courses. The schedule must reflect training requirements and constraints outside the project. Schedule individual courses to accommodate personnel who may require training in more than one area. Identify critical paths in the training schedule such as the time period for the product's installation and conversion to production status.
7. Defines the required course(s), outlines their content and sequence, and establishes training milestones to meet transition schedules.
8. Tailors the instruction methods to the type of material being presented. Include classroom presentation, interactive computer-assisted instruction, demonstrations, individual video presentations, and hands-on experience, either live or simulated.
9. Identifies trainers who are technically knowledgeable and were involved in the design and development of the system. For projects with extensive and formal training requirements, it may be necessary to provide training for the trainers.
10. Consider availability of the following: users, system-tested software, training rooms and equipment, and the completion of system documentation and training materials.

3.5.7 Develop installation plan

The Installation Plan is prepared to specify the requirements and procedures for the full-scale installation of the developed product at the system users' work sites. The plan also addresses the installation of any hardware, software, firmware, and equipment needed to operate the product at each site. In developing an Installation Plan consider each site's requirements for continuity of operations, level of service, and the needs of the project team, users, maintenance personnel, and coordination.

Work closely with the system owner and representatives from the user sites to assure that all site-specific hardware, software, and communications installation requirements are addressed in the Installation Plan.

Ensure any special requirements are adequately documented. Place a copy of the initial Installation Plan in the Project File.

Develop an initial Installation Plan that addresses the following issues:

1. Schedule of all installation activities.
2. Items to be delivered to each installation site.
3. Number and qualifications of personnel performing installation.
4. Equipment environmental needs and installation instructions.
5. Hardware, software, firmware, tools, documentation, and space required for each installation.
6. Special requirements governing the movement of equipment to each site.

7. Communications requirements.
8. Dependencies among activities affected by installation.
9. Installation tests to assure the integrity and quality of the installed product.

3.6 Testing stage

In this stage, components are integrated and tested to determine whether the product meets predetermined functionality, performance, quality, interface, and security requirements. Once the product is fully integrated, system testing is conducted to validate that the product will operate in its intended environment, satisfies all user requirements, and is supported with complete and accurate operating documentation. User Acceptance Testing (UAT) follows System Testing, and requests-accepts feedback from users to make any final adjustments to the system before releasing the product for implementation.

Refer to the Testing Process Manual for more information regarding testing.

The high-level activities for this stage are:

1. Conduct integration testing.
2. Conduct system testing.
3. Conduct user acceptance testing.

The output work products for this stage are:

1. Project coordination plan [revised]
2. Maintenance plan [revised]
3. Requirements traceability matrix [final]
4. Conversion plan [revised]
5. Test type approach and reports [final]
6. Test cases [final]
7. Transition plan [revised]
8. Installation plan [final]
9. Training plan [final]
10. Operating documentation [final]
    A. Users manual
    B. Developer's reference manual

3.6.1 Testing

Testing activities focus on interfaces between and among components of the product, such as functional correctness, system stability, overall system operability, system control, and system performance requirements (e.g., reliability, maintainability, and availability). Testing performed incrementally provides feedback on quality, errors, and design weaknesses early in the integration process.

3.6.2 Conduct integration testing

Integration testing is the first activity in the Testing Stage and requires special attention to preparation. The Pre-Acceptance Checklist, Integration and System Test Checklist, and Testing Package Checklist each provide the necessary steps for their preparation.

During integration, the components constructed by the development personnel, vendors, and reusable code or modules obtained from other sources are assembled into one product. Each assembly is tested in a systematic manner in accordance with the Integration Section of the Test Plan. An incremental approach to integration enables verification that as each new component is integrated, it continues to function as designed and both the component and the integrated product satisfy their assigned requirements.

Given the incremental nature of the Testing Stage, completion and sign-off of the Integration Section of the Integration and System Testing Checklist is required prior to moving on to System Testing.

Refer to the Testing Process Manual for more information regarding integration testing.

Each requirement identified in the Requirements Specification must be tested during integration testing. This traceability ensures that the product will satisfy all of the requirements and will not include inappropriate or extraneous functionality. Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the integration test to the requirements. Place a copy of the expanded matrix in the Project File.

At the completion of each level of integration testing, a test report is written. The report documents test results and lists any discrepancies that must be resolved before the tested components can be used as the foundation for another integration level. Place a copy of all integration test materials in the Project Test File.

A final test report is generated at the completion of integration testing indicating any unresolved difficulties that require management attention. Place a copy of the final Integration Test Report in the Project File.

Sign-off of the Integration section of the Integration and System Checklist signifies completion of the Integration Testing activities.

A formal reporting system by which detected errors and discrepancies are recorded and fully described is recommended. These reports will help to confirm that all known errors are fixed before delivery of the completed product. Error reports also help to trace multiple instances of the same error or anomalous behavior, so that error correction and prevention assignments can be implemented. The Quality Assurance representative assigned to the project can provide assistance in developing and using an error reporting/tracking system.

3.6.2.1 Integration testing

Integration testing is a formal procedure that must be carefully planned and coordinated with the completion dates of the unit-tested modules. Integration testing begins with a structure where called sub-elements are

simulated by stubs. A stub is a simplified program or dummy module designed to provide the response (or one of the responses) that would be provided by the real sub-element. A stub allows testing of calling program control and interface correctness. Stubs are replaced by unit-tested modules or builds as integration testing proceeds. This process continues one element at a time until the entire system has been integrated and tested.

Integration testing may be performed using "bottom up" or "top down" techniques. Most integration test plans make use of both bottom-up and top-down techniques. Scheduling constraints and the need for parallel testing will affect the test approach.

The bottom-up approach incorporates one or more modules into a build; tests the build; and then integrates the build into the structure. The build normally comprises a set of modules that perform a major function of the system. Initially, the function may be represented by a stub that is replaced when the build is integrated.

In the top-down approach, individual stubs are replaced so that the top-level control is tested first, followed by stub replacements that move downward in the structure. Using top-down integration, all modules that comprise a major function are integrated, thereby allowing an operational function to be demonstrated prior to completion of the entire system.

3.6.3 Conduct system testing

A.k.a., Conduct system verification testing.

During system testing, the completely integrated product is tested to validate that the product meets all requirements. System properties and the functional accuracy of logic and numerical calculations are verified under a variety of possible conditions (e.g., both normal and high-load conditions). All operating documents are verified for completeness and accuracy.

System testing is conducted on the system test bed using the methodology and test cases described in the System Test Requirements section of the Requirements Specification document. The system test environment should be as close as possible to the actual production system environment. Either the project team or an independent test team conducts system testing to assure that the system performs as expected and that each function executes without error. The results of each test are recorded and upon completion included as part of the project test documentation.

Note that regression testing is a critical aspect of system testing. It is performed in order to verify that system modifications have not caused unintended effects and that the software or system component still complies with its specified requirements.

When errors are discovered, they should be reviewed by the test team leader to determine the severity and necessary subsequent action. If appropriate, minor problems can be corrected and regression tested by the project team developers within the time frame allotted for the system test. Any corrections or changes to the product must be controlled under configuration management. Major problems may be cause to suspend or terminate the system test, which should then be rescheduled to begin after all of the problems are resolved.

Users may be encouraged to participate in the system tests to gain their confidence in the product and to receive an early indication of any problems from the user's perspective. Inform users that errors and discrepancies may occur during testing and explain the error correction, configuration management, and retest processes.

Refer to the Testing Process Manual for more information regarding system testing.

Review the draft versions of the operating documents, Training Plan, and Installation Plan. Update the documents as needed. Deliver the final versions of the operating documents, Training Plan, and Installation Plan to the system owner and user for review and approval. Place a copy of the approved documents in the Project File.

Place a copy of all system test materials (e.g., inputs, outputs, results, and error logs) in the Project Test File.

Sign-off of the Integration and System Testing Checklist and the Pre-Acceptance Checklist signifies completion of the System Testing activities.

Generate a test report at the conclusion of the system test process. The report documents the system test results and lists any discrepancies that must be resolved before the software product is ready for acceptance testing. Place a copy of the report in the Project File.

3.6.4 Conduct user acceptance testing

A.k.a., Conduct user validation testing.

Acceptance of a delivered product is the ultimate objective of a development project. Acceptance testing is used to demonstrate the product's compliance with the system owner's requirements and acceptance criteria.

At the system user's discretion, acceptance testing may be performed by the project team, by the system owner and users with support from the project team, or by an independent verification and validation team. Whenever possible, users should participate in acceptance testing to assure that the product meets the users' needs and expectations. All acceptance test activities should be coordinated with the system user(s), operations personnel, and other affected organizations.

Acceptance testing is conducted in the test environment using acceptance test data and test procedures established in the Acceptance Test Requirements section of the Requirements Specification. Testing is designed to determine whether the product meets functional, performance, and operational requirements. If acceptance testing is conducted on an incremental release basis, the testing for each release should focus on the capabilities of the new release while verifying the correct operation of the requirements incorporated in the previous release.

If the project team is not conducting the User Acceptance Test (UAT), training may be required for the personnel performing the testing. The acceptance test participants and their experience with the product and the operating environment should have been identified in the Acceptance Test Requirements within the Requirements Specification.

Acceptance testing usually covers the same requirements as the system test. Acceptance testing may cover additional requirements that are unique to the operational environment. The results of each test should be recorded and included as part of the project test documentation.

UAT is typically the final phase in a software development process in which the software is given to the intended audience to be tested for functionality. UAT is done by making the software available for testing by an in-house testing panel comprised of users who would be using the product in real-world applications. UAT is done in order to get feedback from users to make any final adjustments to the programming before releasing the product to the intended user community.

The level of training will depend on the testers' familiarity with the product and the platform on which the product will run. The advantage of having users acceptance test the product is that they are the experts most familiar with the information flow and how the product works.

It is recommended that the operating documents and other test materials be distributed to the test team prior to the actual start of the acceptance test training. This will give the test team time to become familiar with the product and the test process and procedures.

Subject the test environment to strict, formal configuration control to maintain the stability of the test environment and to assure the validity of all tests. Review the acceptance test environment, including the test procedures and their sequence, with the system owner and user before starting any tests.

Testing is complete when all tests have been executed correctly. If one or more tests fail, problems are documented, corrected, and retested. If the failure is significant, the acceptance test process may be halted until the problem is corrected.

In order to complete the testing process the following additional elements must be complete:

1. Completion and sign-off of User Acceptance Testing is required prior to moving on to the Implementation Stage.
2. Refer to the Testing Process Manual for more information regarding user acceptance testing.
3. Sign-off of the User Acceptance Checklist and the Testing Package Checklist signifies completion of the Testing Stage.
4. Prepare a formal Acceptance Test Report. Summarize the test procedures executed, any problems detected and corrected, and the projected schedule for correcting any open problem reports.
5. Place a copy of all acceptance test materials in the Project Test File.

3.7 Implementation stage

Implementation of the product is initiated after all testing has been successfully completed. This stage involves the activities required to finalize the install (or conversion) the system and activate the system's operation. The activities associated with this stage should be performed each time the system is installed at a site.

User training may be required to complete the implementation process. A description of the training necessary for developers, testers, users, and operations staff is provided in the Training Plan.

The high-level activities for this stage are:

1. Perform installation activities.
2. Conduct installation tests.
3. Transition to operational status.

The output work products for this stage are:

1. Project coordination plan [final].
2. Maintenance plan [final].
3. Conversion plan [final].
4. Transition plan [final].
5. Installation test materials [final].
6. Operating documents.
7. Operating system.

3.7.1 Perform installation activities

The installation process involves installing, loading, copying, or migrating the system to the production platform and the provision of operating documentation and other support materials at each site.

At each installation site, inspect the facility to assure that site preparation is complete and in accordance with the Installation Plan. Initiate any actions that are needed to complete the preparations. Conduct an inventory of all vendor provided hardware, software, firmware, and communications equipment.

Follow the procedure specified in the Installation Plan when installing. Monitor all installation activities including those performed by vendors.

Use the following procedure to perform the installation activities.

1. Coordinate the installation with the system users, operations personnel, and other affected organizations.
2. Ensure that any necessary modifications to the

physical installation environment are completed.
3. Inventory and test the hardware that will support the product. This inventory should be performed in advance of the planned installation date to allow time for missing hardware to be obtained and malfunctioning equipment to be replaced or repaired.
4. If the product requires an initial data load or data conversion, install and execute the tested programs to perform this process.
5. If the product requires, then install the software product onto the hardware platform.

3.7.2 Conduct installation tests

Ensure the integrity and quality of the installed product by executing the installation tests defined in the Installation Plan. Testing is performed to verify that the product has been properly installed and is fully operational and in production.

The installation test(s) are designed to validate all functions of the product and should specify a standard set of test results and tolerances. If the product being installed is a modification to an existing system, all remaining functions that may be affected by the new product should be tested.

Document any problems and identify corrective action. Select a diagnostic package that will pinpoint problems quickly and allow for timely corrections. Retest all equipment and software after a repair, replacement, or modification.

When installation is complete, rerun a portion or all of the system test and dry-run the acceptance test procedures to verify correct operation of the product.

Place a copy of all Installation Test materials in the Project File.

3.8 Transition to operational status

The transition of the product to full operational status begins after the formal acceptance by the system owner. Use the procedures described in the Transition Plan to implement the transition processes. Conduct or support stress tests and other operational tests. Determine product tolerances to adverse conditions, failure modes, recovery methods, and specification margins. Complete any training and certification activities. Ensure that support to be provided by contractors begins as planned.

The project team is usually expected to provide operational and technical support during the transition. Identify project team personnel with a comprehensive understanding of the product who can provide assistance in the areas of installation and maintenance, test, and documentation of changes. Technical support may involve the analysis of problems in components and operational procedures, the analysis of potential enhancements.

Transition to full operational status should be an event-oriented process that is not complete until all transition activities have been successfully performed. Withdraw the support of the project team personnel in a gradual sequence to ensure the smooth operation of, and user confidence in, the product.

All Project File materials, operating documents, a list of any planned enhancements, and other pertinent records should be turned over to the maintenance staff. Access rules must be modified to provide access to the product by the maintenance staff and to remove access by the project team and other temporary user accesses. Programs, files, and other support software should be in the production library and deleted from the test library, where appropriate.

For major systems involving multiple organizations and interfaces with other systems, a formal announcement of the transition to production is recommended. The announcement should be distributed to all affected groups. The names and contact details of the team should be included.

The system is transitioned into operational status. Project File materials, operating documents, and other pertinent records are turned over to the maintenance staff.[Engineering] Systems design
System(s) is a word that takes on relatively distinct meanings in different contexts. In the context of design, a system can be defined as an emergent or designed network of interconnected functions that fulfil an intended unit of satisfaction (system outcome or result). Additionally, system(s) has been described as a holistic, embodied way of thinking about reality. Accordingly, the term system(s) represents both a way of inquiry and an object of inquiry. In the engineering context, system(s) embodies both a way of designing and an object of design.

> **NOTE:** *The socio-technical perception (or, nature) of reality assumes that the real-world comprises systems that can be 'designed'. Therefore, it implies that models of those systems can be made and their behavior can be simulated.*

Human-oriented design is a unique form of inquiry and action that aims to create and transform systems to fulfil human needs. Therein, systems thinking provides a base (approach) for synthesizing knowledge of how humans may live optimally together in a common ecology.

Here, a systems design approach refers to the mental model (or, approach methodology) through which designers "frame" (understand and construct) the world, sometimes referred to as a perspective or "paradigm". Systems design is an approach that guides designers in their visualization and resolution of complex problem situations. Systems design is an approach to creating better systems for humankind.

Both the need to support increasing changes in the scale of the challenges facing the development of society's infrastructure and resource limitations, have led to the emergence of a common and unified field

of 'system' design. The implementation of a systems approach to societal [engineering] design is optimal, and its result is, the visual-materialization of the context of a socio-technical network of habitat infrastructures. These habitat infrastructures are designed and materially developed. Herein, the systems design approach seeks the integration and unification of all human-oriented information in order to pre-determine an optimal structural re-orientation for the next iteration of the societal system. This "learning" and consciously-integrating (i.e., emerging) approaches necessarily recognizes the need for a unified societal perspective that considers the capacity of [common] design to improve everyone's well-being by meeting (completing, fulfilling), currently, everyone's basic and full opportunity needs of existing generations, while sustainable [habitat] 'construction' for future generations.

Obviously, all 'human' groups must be open to contribute to the whole system design (WSD to be executed), the service-product 'habitat' system (SPS/PSS).

From a systems thinking perspective, problem solvers are the whole individual societal system, which is a networked community of human conscious-organismal entities. Community, humanity in this context, may obviously, and only solve its global societal problem situations by identifying and reasoning ("discussing" and "conversationling") at length the relationship between design and the materialization. That designed system, which is designed to be commonly optimal for all, must highlight openly difference in social relations of information and power in order to optimize a system that is commonly fulfilling for all, given all that is known. The distribution of information from 'unified' (commonly open to all) to 'centric' (power-over-others) may be visualizes as an information-based socio-technical system. There is the composition and decomposition of information, the discovery of available information (i.e., search), the controlled inquiry into new reality information (i.e., consciousness sciences), computation of data (i.e., hardware-software, interface-conditional programming). Systems engineering is a method of designing systems. Logic, as expressed by consciousness, is otherwise known as, critical thinking. Logic, at the individual (and hence, societal level) is necessary to understand the idea that an information system structures everything experienced, and that experiential objectives can oriented the structure of the next iteration of the societal system. Critical thinking is necessary to develop technology and social organization is necessary to enable its socially-effective and full application as an optimized habitat system. A unified approach necessitates integration of the 'positive' approach of extensionable compassion, and the revealing 'negative' approach of socially visualizing an open network of power, control, domination, and oppression, to reduce social information and spatial sets that reduce optimal and common, individual human-fulfillment.

3.9 Whole systems design

Whole system(s) design is a collaborative and integrative approach that enables a common (i.e., collective) response to socio-technical (i.e., complex real-world) problems). Whole systems design is required for solutions that scale optimally for all of humanity. Through a systems approach, designers take social and technical [parallel] decisions on what systems methodologies and design tools to use, based on their unified understanding of each problem situation.

> **INSIGHT:** *Society is a whole system, and its engineering needs to be re-solved as a whole [integrated] system.*

Note that in software systems, the whole system(s) design uses conditional programming (i.e., procedural software, learning principles) to produce holistic solutions (i.e., to produce solutions that account for the whole of humanity and the environmental ecology).

A community project-systems design approach is a co-participatory approach to every [human-involved] problem situation, where solutions should not be imposed. Rather, stakeholders should be empowered to understand and participate in the functioning of the system (Note: this idea is expounded upon in the Lifestyle System Specification). Moreover, stakeholders actively participate in the conceptualization and implementation of the iteratively ("newly") designed societal system.

In the market, however, there is great confusion over the application of the systems methodology. Some of the "stakeholders" are non-existent entities "institutions, market enterprises (businesses; market coercion), and government enterprises (State coercion; states where 'leaders' control populations through relationships of power-over-others, rather than, coordination for all); which, together form the idea commonly referred to as the 'structural violence', as the common experience of most of modern 21st century society. Layers of confusion and abstraction will limit simplex, higher-system [synthetic] thought (i.e., the perceive that the optimal is to cooperate in the moment toward the fulfillment of all individuals for one's own fulfillment in common with those whom share its conscious, experiential [cosmic] environment.

3.9.1 Living systems design

Living systems design involves ways in which a designer can look at the patterns and life principles that are found within living systems that humanity operates in, and then, apply these patterns and principles to the products, and processes. Living systems design has different names depending upon the disclipine, including biomimicry, permaculture, closed-loop economics, circular economy, and waste equals foods.

3.10 Systems-oriented design

Systems-oriented design (SOD) is an applied knowledge-based (i.e., skills, lifestyle training) approach intended to develop better designs, visualizations, and systems practices. Systems-oriented design, as a holistic approach, has a requirement for a project-based information set, because it accounts for the design of executable interactions in time (in a real-, spatial-world). It must consider different network types and boundaries within a particular socio-technical system to ensure the system functions for all common individual human needs. Systems-oriented design exists as a tool to design of a coherent combination of processes and service-product (or product-service) combinations that together can fulfil the function of the system as specified by humanity in common.

The core design output of a human systems-oriented design is the generation of socio-technical models that are large and information-dense diagrams that act as a bridge between inquiry and design. These models, are visualization maps used to synthesize and interrelate knowledge, and they become a commonly shared understanding of the system among stakeholders.

3.11 What is human systems engineering?

A.k.a., Human system integration (HSI), human engineering [criteria], human engineering standards, human systems integration, human factors, human ergonomics, user-centered design.

Human systems engineering is the process of developing and operating a socio-technical system expressed as a habitat service system and composed of a real world information model that iteratively resolves a higher potential for everyone, given the ability to become more knowable. Human systems engineering is the engineering of systems to meet human requirements (i.e., human needs). Human systems engineering integrates an understanding of human capabilities and human needs into a systems design using an iterative model of systems engineering development.

Human systems engineering (a.k.a., human systems integration) is a structured systems approach to the designing and development socio-technical systems that will involve humans, ensuring alignment of the final system with their requirements, capabilities, and limitations. When perceived from a life cycle viewpoint, human systems engineering is the activities involved throughout the system life cycle that address the human element of system design (one of the first international technical standards for this idea is IEEE 1220-1994, 1998). In other words, human system engineering creates socio-technical life-cycling systems that have the potential to function effectively for humans.

Human systems integration (HIS, NASA terminology circa. 2010) emphasizes human considerations (requirements) as a/the top priority in systems design to reduce life cycle issues and optimize system performance and usability when humans are present. Essentially, human systems integration is the relationship between humans and their environment – particularly how systems are designed and used relative to that relationship – with the goal of ensuring a safe and effective environment that meets human requirements.

Human systems engineering is a comprehensive engineering methodology for integrating human requirements as part of an overall system solution. The goal of human systems engineering is to optimize the total system performance by accounting for both the human and technological components, and their integration.

Human systems engineering starts with an accurate representation of human needs/requirements, which allows for the development of an effective system (i.e., a system that effectively meets those inputs in its operation). Human systems engineering provides the potential for optimizing the interface between the human and his/her environment or work processes.

Similarly, human factors engineering is the application of information on physical and psycho-sociological characteristics (as requirements) of humans to the design of devices and systems for human use. Note that the terminology here can be confusing, because it could be said that simply accounting for 'human factors' (a.k.a., human requirements) in engineering is human systems engineering, and thus, there is no need for a special label when the approach is unified. And, the approach is necessarily unified when engineering a unified societal service system for humankind.

The term 'social engineering' is associated with many negative connotations. In common parlance, social engineering refers to the design and influence of social organization and social behavior. It brings up visions of advertising, propaganda, manipulation, and scamming. These associations with the term social engineering are applicable under market-based conditions, but may not be applicable to other societal types. Note that terms like social science and systems science are also used when applying the systems approach to social systems.

There is a substantial body of knowledge in both human factors, ergonomics, performance, and usability demonstrating how user-centered design can be organized and applied effectively.

Human systems integration design criteria, principles, and practices for standards:

1. Improving performance of personnel (users).
2. Enhance the usability, safety, acceptability, and affordability of technology and systems.
3. Achieve the required reliability and productivity of personnel-equipment combinations.

Human system integration design engineering (human-centered design):

1. Understand the user and environment.
2. Develop concept of operation.
3. Allocate function between user and system.
4. Perform user task analysis.
5. Conduct requirements analysis.
6. Visualize and produce design solutions.
7. Evaluate designs and iterate solutions.

Areas of technical expertise necessary for proper HSI include, but are not limited to:

1. Human factors and human engineering (including crew workload and usability, human-in-the-loop evaluation, and human error analysis).
2. Crew health and countermeasures.
3. Environmental health (including radiation, toxicology, and other areas).
4. Safety.
5. Systems engineering.
6. Architecture.
7. Crew functions and habitability functions (including nutrition, acoustics, water quality and quantity, etc.)
8. Crew interfaces and information management.
9. Maintenance and housekeeping.
10. Ground maintenance and assembly.
11. Extravehicular activity physiology.
12. Mission operations.
13. Training.

User-centered design (UCD) is a well-established design approach that concentrates on developing usable systems by focusing on the system users, their needs, and requirements. The approach applies principles of human factors and ergonomics, as well as usability knowledge and techniques.

The goals of the user-centered design approach are to:

1. Enhance effectiveness and efficiency.
   A. Improve human well-being.
   B. Increase user satisfaction.
   C. Improve accessibility and sustainability.
   D. Counteract possible adverse effects of use on human health, safety, and performance.

3.11.1 User-centered design

User-centered design is formalized by multiple standards and standards setting bodies:

- ISO 9241-210 - provides requirements and recommendations for user-centered design principles and activities throughout the life cycle of computer-based interactive systems.
- ISO/IEC TR 25060 - describes a potential family of International Standards, named the Common Industry Formats (CIF), that document the specification and evaluation of the usability of interactive systems. The Technical Report focuses on documenting design and development elements of usable systems. It does not prescribe a specific process and is intended for use with ISO 9241 standards.
- ISO/IEC 25062 - standardizes the types of information captured with user testing. The level of detail allows the same or another organization to replicate the test procedure. Major variables include: user demographics, task descriptions, test context (including the equipment used, the testing environment, and the participant and test administrator's interaction protocol), and the metrics chosen to code the study findings.
- NISTIR 7889/7990/7934 - Human Engineering Design Criteria Standards
- MIL-HDBK-759C (07/31/1995) - Handbook for Human Engineering Design Guidelines
- ISO 9241 (06/01/1997)1 - human centered design and human-computer interaction
- Section 508 of the Rehabilitation Act of 1973 (08/07/1998)2 - accessibility by those with disabilities
- ISO/IEC TR 25060 (09/01/2006) - Systems and software engineering – Systems and software product Quality Requirements and Evaluation (SQuaRE)
- Ministry of Defence Standard 00-250 (05/23/2008) - Human Factors (HF) and Human Factors Integration (HFI) requirements
- NASA/SP-2010-3407 (01/27/2010) - Human Integration Design Handbook (HIDH)
- ISO/IEC 25062 (07/15/2010) - Software engineering – Software product Quality Requirement and Evaluation (SQuaRE)
- MIL-STD-1472G (01/11/2012) - human engineering design criteria, principles and practices
- ASTM F1166 (06/28/2011) - the design and evaluation of human-machine interfaces

3.12 Service product design

A.k.a., Product–service systems (PSSs), or more accurately, service-product systems (SPSs).

In the literature, the integration of product and services is most often called a PSS. Naturally, a human designed (and oriented) 'habitat' system exists to provide functions that fulfil human needs through service, and then product, combinations. Obviously, this conceptualization is found across different "professional-market" disciplines, such as Operational Research, Information Systems, Systems Engineering, Software-Hardware Systems, Politics, Business Management, Marketing, and Self-development.

A systems thinking perspective on SPS/PSS is fundamental for a commonly aligned conceptualization and in-depth understanding of the socio-material system as it is currently in place, and simultaneously, possibly in place in the future. SPS/PSS design is an effective form of conceptualization of complex societal systems.

Through the understanding that all types of 'human' society represent an understandable and unifiable societal information systems, that will express for that human-society an observable decisioning-materializing system. That system can be visualized before it is generated, a process necessarily open to every individual, if the orientation of the next societal generation is to be a more optimal form of free and open access system as the desired design result. SPS/PSP is as a tool for analyzing and synthesizing (integrating) causal loops (e.g., systemic relationships, procedural decisioning) among community users and operating designers, among a unified habitat. Computationally, SPS/PSS simulates the dynamic behavior of systems quantitatively, because it is a visualization of a materializing system (i.e., a system that is sensibly quantifiable; i.e., can be expressed in numeric pattern, fractally). The SPS/PSS is the operational system; it is the user-interface conceptualized as a socio-technical 'societal' system.

3.13 Engineering service operations

Engineering operations is the application of knowledge and technical design to fulfill a requirement formulated as a problem. In order to solve a problem in engineering, the cause (or "root") of the problem, and its context, must be understood. The result of the process of engineering is the construction and/or continued operation (or recovery operation) of a technically existent (and experiencable) [societal] system; hence, the dual life-cycle phases of a **unified engineering approach** (with construction and operation information sets):

1. **Construction (and Re-construction) of system to specification:**
   A. Design feedback integration.
   B. Construction preparation.
   C. Construction (building structures, building sub-structures, and equipping).
2. **Operation of system at specification:**
   A. Operation preparation.
   B. Operation (automation and/or human event involvement).
   C. Monitoring (quality assurance and operational feedback).
   D. **Deconstruction of system to specification:**
      1. Deconstruction preparation.
      2. Deconstruction.
      3. Re-integration.
      4. Monitoring (quality assurance and operational feedback).

A clarification must be made here. Take, for example, a chef and a waitress. Each is equally maintaining and operating a service system. A mother feeding a child is maintaining and operating a service system [for the child]. Someone feeding themselves through food acquired anywhere is operating (and maintaining, or not) a service system. There is a constructed and iterating foundation to the system that structures the operation and maintenance of fulfillment to humans (Read: that which humans really do require, and have specified) as a set of societal-level requirements traced to their societal-level [support] services (a.k.a., the Service Systems: life-support service, decision service, facility service systems, etc.).

Service activities within the service system(s) may be automated and/or maintain human involvement, where desired(/-able) to humans. That which is desirable to humans is an objective to which humans may (or may not) align the next iteration of their societal-life system (society, civilization). In the social information set of the real world information model, 'objectives' are [expressed] 'values' to which decisions are [objectively] aligned or not.

3.13.1.1 The determination of a societal [service] system design structure

The design and the service system of which the design is a part, is engineered into existence and continued operation as a solution to the requirements that humans have set for its continued operation.

The design of a real [world, societal-level] system:

1. **The Unified [Societal] Information System:** social; decision; lifestyle; material. Social processes; decision processes; lifestyle processes; material processes.
   A. Open source collaborative development effort.
   B. Common and InterSystem Team effort.
2. **The Physical [Societal] System:** Actualized configuration in formation of a Habitat service system.
   A. Open source collaborative development effort.
   B. InterSystem Team effort.

A real-world, societal-level system may sub-composed of habitat service sub-systems (at the material level of experience of a conscious user). The design, generation and otherwise execution upon and within of these societal systems may be aligned (or not) to explicit[ly human] requirements (or not).

The materialized instantiation of these [required] habitat service systems necessitates the three 'material' information categories of materialization (other conceptions of which include: construction, creation, generation, etc):

1. **Specification**, which involves conceptual through

to physicalizable designs of a system that meet the requirements. The specifications are a set of visualizations forming a conception through to technicalization of a system, using tools and techniques (i.e., processes).

   A. Knowledge encoded by engineering information groups becomes visualized and tested.

2. **Operation (construction and de-operation)**, which involves a set of activities (operations, tasks performed in a [pre-]controlled structure) necessary for the materialization (creation and re-creation) of the system.
   A. **A non-prior system** will have its first constructed instantiation, and then life-cycle thereform.
   B. **A prior existing system** will have a module life-cycle. A set of unique operations exist for each habitat service system. All systems provide material and informational services directly and indirectly to humans. Indirectly means that the service is that of the lifecycle of the systems themselves.
3. **Validation to specification**, which involves a set of validation activity **tests for alignment** with requirements, imperatives, and the user. Here, experience generates feedback and revised integration directionally-intentionally evolves the system, or generates an entirely new one to replace the last. The design of a new system, communicated as a delivered specification, must be validated to sufficiently meet operational expectations as defined by the user requirements.
   A. Without validation there is no feedback valuable for alignment, and without feedback valuable for alignment, there is no re-alignment to a direction set by an objective.

4 [Engineering] Design and development

> **NOTE:** *If design is political, then debating and obfuscation are design skills. Yet, neither obfuscation nor debating facilitate functional design. Design is not political.*

The determination of a societal-level solution is, in part, through a common engineering design and development process. Design through to execution becomes the fundamental engineering [development] activity (process). The engineering of systems revolves around the problem solving of design. Design consists of a sequence of stages starting from the perception of need and terminating in a final (end/firm) description of a particular design configuration. Each stage is in itself a design process and is an iterative sequence of sets. Design fills the gap, the difference or separation between what is to be done (and why), and how to do it.

Both objects and processes (given an environment) can be designed. The societal information system contains both information objects as well as information processes, given an information-based environment. Therein, the habitat service system is a combination of both objects and processes with material reference. For example, the energy sub-system is a supra-process life-support category, containing object-assets cycled through materiality as part of service system processes to meet human energy [operational] requirements. The energy sub-system delivers service (asset-object) types, including power generation and storage systems (e.g., wind turbine electric generators and batteries).

> **NOTE:** *Since engineering quality is only as precise as its tools, the quality of the design tools (e.g., representations, conventions, and applications) has a direct impact on the quality of the result.*

Engineering design, as an activity, produces an "engineering design" as a product, which is delivered to be acted upon and through. This information product is a representation of the physical product to be produced and/or operated (and is variously called the designed service object, system model, product model, engineering drawing, etc.).

A design is a solution for a given set of problems. A design is a solution towards a problem experienced by a real user; the design solely relies on the "user" for it to be appreciated. In other words, designing is the process of problem solving.

It's not a canvas for personal expression. Design is not personal, like someone's preference in music or art. A quality designer takes design decisions based on user research, knowledge, and best practices, with a focus on communicating clearly and achieving human goals. The process of design must be complementary with the objectives. This means the design and implementation

process is critical. If flexibility and participation are the objectives of an organization's design, then the question must be asked, how might an organization be designed so that it is flexible, interactive and participatory.

Design is a continuous commitment, a re-iterative process. A design is a solution, which inevitably has to be changed, therefore it is critical to build learning and change ability into the organization that produces designs. In concern to feedback on designs, feedback should be linked to goals. Designers are tasked with generating creative and unique solutions, following a process that builds upon logic, observation, knowledge, and feedback to arrive at an optimal output.

> **APHORISM:** *"A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away." -Antoine de Saint-Exupéry*

The quality of any engineering design - whether it's a commercial product or a data model - is a direct function of the ability of the design system to access and codify the knowledge of the users, and systematically translate that knowledge into a model of the desired product/system.

> **NOTE:** *It is the design element in the practice of engineering development that distinguishes engineering as an activity from the sciences.*

In design, engineering is a:

1. Deliverable (noun): The production of a real-world, hardware (object) and/or software (concept) system.
   A. A design describes a deliverable (noun, object). A design is a visualization that shows the final object/system.
2. Process (verb): The decisioning processes that determines the function, performance, parameters, interface, etc., to be delivered and used/operated.
   B. A plan describes how the deliverable was designed and materialized. A design is a plan that shows how the final object/system will be created and operated.

Engineering service design involves, at least:

1. Service concept.
2. Services.
3. Processes.
4. Tasks.
5. Roles & technologies.
6. Equipment and computation.
7. Resources and conscious motive.

Engineering design is:

1. A verb, a logical sequence of activities and decisions that transforms an operational need into a description of system performance parameters and an optimal system configuration.
2. A comprehensive, iterative and recursive problem solving process, applied to transform needs (and requirements) into a new system (or system state).
3. A standardized, disciplined process for the development of system solutions that provide a system that meets user needs in an environment of uncertainty.
4. The process of selecting the means and contriving the elements, steps and procedures for producing what will adequately satisfy some need.
5. Design is founded on the consistent, directed resolution of a system into reality. Design is founded on decisioning.
6. To create order, structure and/or pattern as an outcome [of the process of designing]. Crucially, it is the order, structure and patterns in design actions that are the source of these attributes of design. Design, even in nature, is not an outcome of anything other than a highly structured causative sequence of actions (intentional or not).
7. A problem-solving activity. Wherein, problem solving is that form of activity (or action) in which an organism intends to realize a goal, a gap in the 'route' to the goal, and a set of alternative mans, none of which are immediately and obviously suitable. It is a path of resolution followed in conformity with the guiding criteria of a goal (the user requirements/user needs/intervention intention) subject to the constraints of viability (the opportunities and reality of implementation technologies). Design is action requiring the mind to examine (process) each and every item (of information), which pertains to the design in a continuous and uninterrupted process, including all of these in an adequate and orderly enumeration.
8. A resolution arising out of a sequence or iteration of process transformations or work products outputs. Engineering design is a matter of recursion that resolves transformations and work products definition at different scales.

The design process is repeated sequentially in a number of stages, proceeding from a global view of the system, to progressively more localized considerations, and from an abstract and fluid description to a concrete, physicalizable one. Therein, abstraction forms functional descriptions and material detail forms implementation descriptions.

> **NOTE:** *To build and test is to construct the whole from the parts (to piece parts together, to join into one that operates together within a boundary).*

The processes of design build models of that realisation and future reality, as descriptions of:

1. Intended intervention.
2. The function of some intervening agent.
3. The physical composition and ordering of that agent.

These models are progressively resolved, each with the other according to ever greater detail, until the risks of achieving a viable and valid outcome diminish to an inconsequential level. Thus, design resolution is often recursive in practice.

The concordant resolution of models follows a sequential decision-resolution path; one that is continually revisiting different levels of modeling detail, past decisions and preceding lines of decision rationale.

The design resolution path forms in linear time from a process of execution concurrency relating to the following three information sets:

1. Structural detail (as system architecture - function, form, and effect domains). Elemental connectivity must exist between all domains.
   A. Design rational (logical domain).
2. Solution progression (as organizational processes and work products - material, energy, information domains).

Both objects and processes can be designed. For example, the habitat service sub-system is a combination of both designed objects and processes. Therein, the energy sub-system is a supra-process life-support category, containing object-assets cycled through materiality as part of service system processes (Read: the operational processes). The energy sub-system delivers asset-object types including power generation and storage systems (e.g., wind turbine electric generators and batteries).

Regardless of what is being designed, design involves several core information processes; design is:

1. Generative (i.e., involves creating, analytical-synthesis) of some new information set. Some thing new is the output of design.
2. Iterative (i.e., involves repeated cycles of trial, error, and learning).
3. Representational (i.e., visualizations, models, and prototypes document and communicate a design) with many potential views, given inquiry intent.
4. Collaborative (i.e., there is fulfillment in optimizing common designs for fulfillment).
5. Complex, probabilistic, and emergent.

The primary [systems] engineering tasks include:

1. Develop the total system design solution.
2. Develop and track technical information needed for decisioning.
3. Test the system.
4. Verify that technical solutions satisfy user requirements.
5. Operate the system.

4.1 The design phase

APHORISM: *If you want to be a powerful creator, become good at systems [thinking] and understanding and solving structural problems.*

Design for (i.e., the common elements of the design phase are):

- Function - the "means" by which (how) the system operates for user fulfillment.
- Interface - the "means" by which (how) which two systems interact (Read: share information).
- Performance - the evaluated quality "means" by which (how in alignment):
  - Information is shared between systems (per requirements).
  - The function operates for user fulfillment (per requirements).

Service system engineering life cycle:

1. Service operation.
   A. Issue inquiry for service.
2. 2. Service integration.
   A. Organizational value-alignment inquiry.
   B. Solution engineering inquiry.

The contextual elements of use for any particular product (which can be a technology, system, device, piece of equipment, or process) include, but are not limited to:

1. The intended user(s).
2. Their goals and tasks.
3. Associated equipment.
4. The physical and social/informational environment in which the product may be used.
5. Note: A product could be viewed of a as a set of preplanned tasks.

4.1.1 Define the conceptual system [a phase]

The conceptual formation of a projected system may be initialized through a set of imperatives. Often, a sufficiently developed imperative structure is composed of all of the following:

1. **Project [societal] imperatives:**
   A. **A strategic direction**, which is a description of progress along some identified alignment with which humans seek to move or progress. A strategic direction is described by concepts.

1. **Define the mission, vision**, and other strategic directional or outcome descriptives.

B. **Goals (human aspirations)**, which involves a set of criteria representing a list of axiomatic outcomes (conditions) that are to be realized under a given strategic direction of intention.

1. A **societal goal** is a particular category of goals, which are universalizable to a society composed of organisms. Generally, societal goals categorically express the necessary conditions for avoidance of serious harm (survival) and the expressed facilitation of fulfillment.

C. **Needs (human needs, which are objectives)**, which involves a list representing a set of criteria that are of imperative importance to fulfill, including processes and states, for humans to not only survive, but thrive at their fullest potential.

Goals and needs are both:

1. **Measurable**, at minimum through progress on subsidiary goals/objectives, but preferably also directly. Here, measurable refers to that which is independent of personal sensitivity, capable of experience by some population with common senses.
2. **Completion** of goal and **fulfillment** of a need represents significant alignment with or progress toward the strategic direction.

A conceptual design specification includes:

1. A **conceptual specification** (or, Unit; Conceptual specification) is the design for an instance (potential/existent instantiation state) of a community-type society.

A. The **functional specification** (Functional specification) is the Unified Societal System Specification detailing the functional elements of that societal instance.

1. A **technical component specification** (Technical specification) is a set of sub-system iteration states.
2. What is needed here is resources, time, financial budget, ...
3. Feedback from experience.

4.1.1.1 Process and technology mapping

Process and technology mapping is the process of collecting and associating all configuration and connectivity parameters for hard[ware] (material) and soft[ware] (information) systems. In some disciplines, the total set of process and technology mappings for an entity is called a configuration item (CI), and configuration items are used for operations (including production/ fabrication and construction/integration). Process mapping creates an image (diagram, visualization, description) of each organizational (such as, societal or business) process, and what would be needed to continue the process in the absence of any or all of its informational and material (Read: IT, information technology) resources.

In concern to system development, process and technology mapping allows easy duplication of a system. In concern to disaster recovery and system continuity, this mapping (assuming it is backed up) allows operators to re-prioritize, move, and most importantly, restore systems.

4.1.2 Define the technical system [a phase]

An engineered system, is a technical or socio-technical systems system, which is the "subject" of a systems [control] engineering life cycle. Systems engineering is the approach, involving a set of processes, which realize (materialize) a system that accomplishes, fulfills, and completes the imperatives [of the projected system].

In part, the technical system is an expressed (or, express-able) visualization.

4.1.2.1 Information visualization function(s)

An information visualization function is the description of an operation that allows information to be more coherently understood and integrated by a visualizing user. An information visualization function adds shape, an object, geometrical relationship to an information set. Here, a 'function' is an information process or information operation that allows for a clearer and larger expression of what is possibly available, because it conveys the greater -ability to connect the user to the usable system.

4.1.2.2 Traceability

Traceability is a principal information visualization objective. Traceability is the ability to describe and follow information in both forward and backward direction. Full traceability is the ability to explain and visualize the flow of information in forwards and backwards directions, fully. For example, a complete (full) visualization of requirements expresses traceability wherein a requirements statement at any level can be related to any other level, including its source (e.g., human issue, intention, problem) and destination (e.g., output, result, system).

4.1.2.3 Requirements traceability (RT)

Requirements traceability (RT) as a principal objective of project coordination is the ability to describe and follow information about [the life of] a requirement in both forward and backward directions, completely (i.e., without gaps or "jumps"). In order to integrate feedback coherently within a project, there must be traceability

of outputs to inputs. Traceability assures everyone that all requirements can be accounted for in the design at any stage and that no unnecessary requirements are included (probably, unnecessary work). Traceability supports configuration control (if a requirement needs change, its related information flows and impact are visible).

Requirements traceability is a feature (quality, characteristic, attribute, objective) of a system's unified (top-down) design approach, which "guarantees" (Read: makes objectively measurable) that requirements can be identified and inquired into (satisfied) at any stage of a project.

Traceability ensures data on:

1. Where a requirement came from?
2. What requirements are related to it?
3. What requirements were derived from it?

There are sub-conceptions to traceability:

1. **Forward traceability** is required so that design decisions can be traced from any given system-level requirement down to a detail design decision.
2. **Backward traceability** means that any lower-level requirement is associated with at least one higher-level requirement.

4.1.2.4 Requirements use case

A use case (or user story) is the sequence of events to explain your requirement.

4.1.3 Modeling (visualizing-simulating) requirements as mathematical associations

There are two mathematically aligned characteristics when using the systems approach to modeling (by the engineering system) the intention of a 'requirement', or even prior, an 'issue'. The two axiomatic mathematical associations are that of variables and

1. **Variables or Non-functional requirements:** A variable is the way in which an attribute or quantity is represented or expressed. Non-functional requirements may be transposed for variables, which describe an attribute or quantity. How much durability and reliability do you want in the design of your system (its a variable option)? How much autonomy do you want in the design of your society (its also a variable option)? Technological obsolescences as a value, and then a variable, in a societal resolution equation has a different outcome than reliability as a value, and then a variable in a societal resolution equation.
2. **Constants:** Functional requirements become absolute quantities. Whereas, the functions are the parameters, normally a constant in an equation describing a model. For instance nutrition is a constant (not optional) need. In a natural environment of scarcity with an inability to design non-basic technologies, then the nutritional constants are pre-determined by scarcity in the environment and organismal sensation. Therein, human needs may be considered a constant.
   A. **Parameters or Functional requirements:** In an environment where needs may be fulfilled via designed and varied methods, then human needs (which are human requirements to engineering) are more akin, conceptually, to parameters. When the environment determines fulfillment, there is very little that can be done in terms of changing (by choice and design) fulfillment. However, when intellect, resources, and designability are present, then the fulfillment (felt and objective) is not fixed.

4.1.4 Engineered system characteristics

Systems engineering involves the processes of designing and constructing additional possible 'function' into the material and/or informational world, through a particular design. In a purposeful (purposive) context, a design[ed] system has the following kinds and sub-kinds of characteristics:

1. **Physical characteristics (physical properties, physical requirements)**: its materials, structures, and motions.
2. **Operational characteristics (operational properties, operational requirements)** - are properties that are designed into a system, and expressed in the systems operation or state of being. There are two types of operational properties:
   A. **Functional characteristics (capability requirements)**: what it is for; what service(s) it performs. These are the first type of 'ability, functional abilities or capabilities. A description of *functions* of what precisely the system will do. Functional requirement - state what the system will do. Describe how it will behave; what is its specific behavior and functions? A functional requirement is a specific need or desired behavior as seen by an external user of the system. The required capability or function must be delivered by a system through one or more of its components.
   B. **Non-Functional characteristics (a.k.a., dispositional properties and non- functional requirements)** that one possible assembly

produces over another for the same function. Non-functional requirements are the conditions under which the system should perform. These are otherwise known as 'abilities'. These are the second type of 'ability, non-functional abilities or objectives. A description of *features* of what precisely the system will do. Non-functional requirement state what the system will be. The criteria for evaluating the operation of the system, rather than specific behaviors. These requirements cannot be categorized in to function, data, or process (both process and data) requirements.

1. **Execution qualities** - Execution qualities form an interface and [critical] decision path between a user with needs and a [societal] service system that provides access to a capability that meets those need. Execution qualities are often visible during operation (at run time) of the [societal] system itself. Such as: durability, automaticity, and optimization, which are observable during operation (at run time) of the [societal] system itself.
2. **Evolution and availability qualities** - are embodied in the static structure of the system itself. Such as: testability, maintainability, extensibility, and scalability, which are embodied in the static structure of the system itself.

3. There is also, for every system, a [negative] **efficiency characteristic (efficiency requirements)** for all expressed properties: given what is known and what is possible, how off alignment from optimization of materials, structures, motions, and attributes (capability and dispositional) is the system[s design and operation]?

> **CLARIFICATION:** *Broadly, functional requirements define what a system is supposed to do and non-functional requirements define how a system is supposed to be.*

The overall properties of the resulting system commonly mark the difference between whether the development project has succeeded or failed.

4.1.5 Define system non-functional requirements (a.k.a., objectives orientational needs)

> *A.k.a., Design goals, dispositional properties, dispositions, non-functional requirements, non-functional needs, system quality requirements, system performance requirements, performance needs, qualitative requirements, objectives, system control objectives, quality objectives, quality attributes, quality goals, quality service requirements, constraints, features, and values.*

Non-functional requirements (a.k.a., objectives) constrain functional requirements. Non-functional requirements specify under what constraints the functional[ly required] system should function. Objectives are the orientational component of the imperative structure. Defining non-functional-requirements is an orienting process. Objectives are orientational because they predetermine one assembly of components for a given function (or service), versus another assembly for fulfillment of the same function. In other words, a design may be categorized under a specific conceptual state (or condition) given its composition, over a design to fulfill the same function, but with a different assembly.

> **NOTE:** *In a decision system, values are non-functional requirements, which are the social[ly viable] conditions for creation and operation of a stable human society.*

In system requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to evaluate the operation[al performance] of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behaviors or functions.

> **NOTE:** *'Sign' is the way in which a design executes a desired function.*

Simply, objectives are top-level project requirements of a system that identify what its design **should be (non-functional requirements)**, as opposed to what the design should do **(functional requirements)**. Objectives are design goals (a.k.a., non-functional requirements) that describe the desired attributes, qualities, or features the design will have. Objectives allow for exploration of a design and decision space where an optimal selection among alternative options occurs. **Objectives**, which involves a set of criteria representing a concrete, measurable output or outcome to become a requirement for the goal's fulfillment. Embodied consciousness has a set of abilities available to it; and it can extend its abilities through a systems process to create newly available technological functions. The extending of single function can occur in multiple different ways, each expressing a different objective (dispositional property). These newly available technological functions allow us to integrate and automate our functional habitat service systems into a network of resource-access sharing for each and everyone's fulfillment.

In contrast to functional requirements, non-functional requirements are in the form of, "system shall be <requirement>", an overall property of the system as a whole or of a particular aspect and not a specific function. Objectives are expressions of desired attributes and behaviors that the system will express. The system must maintain some [conceptual through to actual] ability in

its performance. Here, objectives specify -ability inquiries for decision (as the planned solution) selection.

Objectives are characterized as:

1. Expressed using the verb, to be. These are "be" words ("be" words include: is, am, are, was, were, be, being, becoming, been).
2. Qualities that the system (object) should have.
3. Measurable, which senses and inputs measures or sources of data for system progression (i.e., evolvement, optimization, or betterment), whether quantitative, qualitative, or both.
4. Logically relevant to the applicable goal.

In order to bring a material system into existence, there is an existent material reality that must be worked with and through. The system which is to be brought into existence is composed of a set of requirements. There are relationships between requirements and the extant material reality. In common parlance, these relationships are called tradeoffs. Objectives are concepts that are encoded into the design, and eventual operation, of the system in order to resolve these relationships toward some particular alignment. These concepts are dispositional properties (-abilities) designed into the system, in expectation of expressing a particular function during the systems lifespan. Here, disposition refers to the arrangement of material reality into a system expressive of a particular technical function, previously conceived of as a objective or dispositional property. Material systems can be arranged in different ways to perform the same service. Dispositional properties prescribe (on the design front-end) and describe (on the operational user-end) the functional expression of a material systems arrangement.

Objectives are expressed as 'abilities' (a.k.a., concepts of operation). An 'ability' is the capacity to form and resolve a process in a categorically pre-defined manner, a dispositional property -- a category of technical action. Specific categories of 'ability' are labelled with that term as a suffix. In other words, an objective defines the pre-defined manner of desired functioning.

An objective/requirement represents a measure of specified change, in order to bring about the achievement of the goal. The attainment of each goal may require a number of objectives to be reached. There is often much confusion between goals and objectives. Whereas as a goal is a description of a destination, an objective is a measure.

Systems engineering depends on the ability to perceive the [multiple] possibilities for action within a environment, so that a system's movement within the environment can be appropriately coordinated. Herein, objectives design *time* into *being*, which is the fundamental principle of concurrent engineering. There are relationships between a designed system, its environment, and the concepts applied to its assembled functioning.

Objectives enable the selection among design alternatives [for the one(s) that express the greatest alignability with the strategic imperatives]. Here, objectives represent constraints which decidedly orient the a functional system:

1. Constraints enable the rejection of unacceptable alternatives [that express dis-alignment ability].
2. Constraints are typically framed as a binary yes or no choice.
3. Constraints establish the design space.
4. Constraints are fixed under consideration of Design Decision Standardization and user requirements.

> *Note: Feature trees are high-level models organizing features into feature groups capturing the entire scope of a project in a single model. They show the relationships between features.*

Here, values become objectives useful for life fulfillment. The encoding of those objectives is likely to produce an economy where only useful things are served; an economy that serves the process of human and ecological fulfillment.

4.1.6 Define system functional requirements (capabilities)

> *Functional requirements area also known as: operational goals, functional attributes, capabilities, capability requirements, requirements, physical objectives, system operational requirements, system performance requirements.*

In contrast to non-functional requirements, functional requirements are usually in the form of "system shall do <requirement>", an individual action or part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or input>process>output model. A [functional] specification describes the necessary functions at the level of unit(s) and components; these specifications are typically used to build the system.

Capabilities occur in pairs in which some property of the environment (e.g., climb-able) is related by a property of the being's or system's capability, known as an it's effectivity (e.g., to climb or walk).

> **CLARIFICATION:** *In common technical parlance, a "value driver" is another term for a primary function, and expresses how to create "value" for the human in line with its objectives.*

> **PROJECT COORDINATION:** *Needs must be appropriately matched with abilities (as in, ability to do), forming a technical, functional system.*

Herein, a 'capability' is the (physical or informational) ability [of a system] to execute a specified course of action, as originating from some source. In engineering design, a 'capability spread' includes the follow capability elements:

1. **Capability Gap (or Gap)** – The inability to execute a specified course of action.
2. **Capability Requirement** (also called "need" or "requirement") – A capability is required to meet [human] needs, current or future.
3. **Capability Solution** – A materiel or non-material solution to satisfy one or more capability requirements (or needs), and reduce or eliminate one or more capability gaps.
4. **Capability Production** - The materialization of the material or non-material (e.g., digital) solution.

The planned, functional design characteristics of a system are otherwise known as a system's functional attributes. Functions are the behaviors expected from the design. A function is an activity that the system should perform or support. A design should perform certain functions for conversion of a given input into a required output. Functions are often expressed as verb-object pairs. Functions describe what the design (or, more often, an object within the design) will "do" or accomplish, with an emphasis on input-output transformations. Something is expected to occur due to the system's existence in the real world. When denoted, functions are arranged in a hierarchy to express their relationship to the project objectives.

The statement of a function typically couples an action verb to a noun or object (e.g., lift a book, support a shelf, transmit a current, measure a temperature, or switch on a light). For instance, "Measure weight of objects up to 120 kg"; "Support weight up to 70 kg and Hold on wall without failure"; and, "Control pointer on a computer".

Even abstract requirements like 'proximity to transport' may be expressed as functions, such as: enable "easy" access to public transport; whereupon, "easy" is defined specifically and numerically.

The design of a system must account for several types of function:

1. The primary function(s).
2. Desirable secondary functions.
3. Undesirable secondary functions.

For example,

1. Project images (for a projector).
2. Generation light (desirable).
3. Generation of heat (undesirable).

Additional examples of function include:

1. The function of a bicycle brake is stop the wheel when applying the brake lever by means of frictional force between rim and brake pad.
2. The function of a hydraulic lift is to elevate heavy weight by means of pascals law.
3. The function of a speaker is to produce sound by means of electro-magnetic induction.

A quality/efficiency spectrum from optimization of the functional and non-functional attributes of the service to highly sub-optimal (i.e., a near zero efficiency rating to negative efficiency).

Service product[ion] functioning includes:

1. The functioning of a product can be described as follows:
   A. Form (Structure) / Characteristics / Function / Values / Needs.
2. The design process follows this sequence in reverse:
   A. Needs / Values / Function / Characteristics / Form (Structure).

5 [Engineering] System concepts

A.k.a., Concept of operations (ConOps), operations concepts (OpsCon), system operational concept (OpsCon).

System concepts bridge the gap between product scope and technical requirements. System Concepts are plain-language descriptions of user-product/system interactions throughout the life of your system from the perspective of all the key stakeholders. How it will be manufactured, tested, installed, used, maintained, stored, and decommissioned.

When developing the system concepts, users describe a day in the life of the product, for all life-cycle stages, and addressing both nominal as well as off-nominal situations. These descriptions are told from the users' perspective describing their expectations for the system's functionality, performance, capabilities, and quality. These expectations are in the context of meeting need, goals, and objectives within the context of the defined drivers and constraints. A complete system concepts information set helps prevent both missing and incorrect requirements. System concepts will help establish a shared vision for the system and facilitate acquisition of the knowledge needed to define a clear, complete, correct, and concise set of requirements. The 'system concept' is the basis for system functional and performance requirements.

System concepts exist within the systems engineering domain, and it is the responsibility of systems engineering (or whomever is responsible for the technical expertise of the system) to develop and maintain 'system concept' information set (document).

5.1 Relationship between Concept of Operations and Operational Concept

Both system concept documents, 'concept of operation' and 'operations concepts' are developed in exactly the same way, and many organizations combine the two information sets into one. Both information sets define capabilities, functionality, performance, and quality needed in the system – just from a different perspective:

1. **OpsCons** focuses on the system under development from a user/operator perspective. Describes the way the system works from the user/operators perspective.
2. **ConOps** focuses on how the system fits into the bigger system of which it is a part and will be developed, tested, and operated. Describes the way the system works from a socio-technical organizational perspective.

More specifically,

1. **Concept of Operations (ConOps, ConOp, CONOP):** A verbal and graphic statement, in broad outline, of a socio-technical organization's assumptions or intent in regard to an operation or series of operations. The concept of operations frequently is embodied in long-range strategic plans and operational plans. In the latter case, the concept of operations in the plan covers a series of connected operations to be carried out simultaneously or in succession. The concept is designed to give an overall picture of the socio-technical operations.
2. **Operational Concept (OpsCon):** A verbal and graphic statement of an socio-technical organization's assumptions or intent in regard to an operation or series of operations of a system or a related set of systems. The operational concept is frequently developed as part of a system development or acquisition project. The operational concept is designed to give an overall picture of the operations using one or more specific systems, or set of related systems, in the enterprise's operational environment from the users' and operators' perspective.

Both information sets address user needs, the life-cycle, and nominal and off-nominal situations. Both ConOps and OpsCons involve the telling of stories, scenarios, or use cases. Both align the users to a common vision, are used to define a feasible approach to meeting the overall needs, goals, and objectives, and are used to further define the various development elements involved in the project.

Documenting both perspectives as 'System Concepts' results in addressing the traditional benefits and outcomes of both a ConOps and an OpsCon thereby avoiding confusion in having to distinguish between whether it is a ConOps or an OpsCon.

5.1.1 OpsCon in brief

A system OpsCon document describes what the system will do (not how it will do it) and why (rationale). An OpsCon is a user-oriented document that describes system characteristics of the to-be-delivered system from the user's viewpoint. The OpsCon document is used to communicate overall quantitative and qualitative system characteristics to the acquirer, user, supplier and other organizational elements.

An Operational Concept Document (information set) is a document for recording an Operational Concept. It is prepared at the acquisition organization and developer level to describe how a particular system (new, modified or existing) will be operated to satisfy its user and operator needs. The description is independent of specific design solutions, although it will make reference to a possible design solution at the highest level of

abstraction. The Operational Concept Document is not a requirements document. It describes the system operational intent and context, and is used to derive needs and requirements.

In order to avoid inclusion of solution-specific information in the initial Operational Concept Document, system operational behavior should be described in the form of capabilities and outcomes. Initially, any reference to an architectural or detailed solution should be minimized. As the system is realized and the Operational Concept Document is revised throughout the product life cycle, references to the specific architectural features of the solution are incorporated.

5.1.2 ConOps in brief

The ConOps, at the organization level, addresses the user's intended way of operating the organization. It may refer to the use of one or more systems as black boxes to forward the organization's goals and objectives. The ConOps document describes the organization's assumptions or intent in regard to an overall operation or series of operations within the organization in regards to the system to be developed, existing systems, and possible future systems. This document is frequently embodied in long-range strategic plans and cyclical (e.g., annual) operational plans. The ConOps document serves as a basis for the organization to direct the overall characteristics of the future organization and systems. A concept of operation phase defines a need or gap to be filled by a system.

A Concept of Operations document (information set) is a document for recording a Concept of Operations. It is developed at the organization (enterprise) level, independent of any specific system solution, to describe how the organization (enterprise) will operate to execute strategy and doctrine. The Concept of Operations Document is not a requirements document. It describes the organization (enterprise) operational intent and context, and is used to derive needs and requirements.

'Concepts of operation' (ConOps) are defined as operational design elements that guide the organization and flow of project elements, including hardware, software, personnel, communications, and data products through the course of a project objective implementation. Conops are the organizational design elements; how people and robots work together; how they flow through different pathways as they accomplish different tasks. Here, the term 'capabilities' is defined as specific functional mission aspects that can take the form of hardware or software. Additionally, capabilities may be high-level ("architecture level") such as high-bandwidth communications or can be lower level such as pan-tilt-zoom capabilities on a camera. Capabilities are the functional aspect looking at hardware and software. What is required to support humans and robots?

By learning which ConOps and Capabilities are enabling or enhancing (and which are not) early on in the development process, NASA's limited resources are better managed towards value-add systems and support technologies.

5.1.2.1 [System] Concept of operation

A.k.a., ConOps, CONOPS, system concept, system concept of operation, operational concept description, operational concepts, operational scenarios, system concepts, use cases, user needs.

Concept of operation (ConOps) is a formal statement (visual and/or linguistic) of the intended operation of a system. A concept of operation is a user-oriented conceptual formalization that describes the characteristics for a proposed system from a user's perspective. A ConOps describes the proposed system in terms of the user needs it will fulfill, its relationship to existing systems or procedures, and the ways it will be used. A ConOps may focus on communicating the user's needs to the developer or the developer's ideas to the user and other interested parties. The main objective of a ConOps is to communicate with the end user of the system during the early specification stages to assure the operational needs are clearly understood and incorporated into the design decisions for later inclusion in the system and segment specifications.

In general, a [system] 'concept of operation' formalization contains the following:

1. Define the environment in which the system will operate.
2. Define the high-level system concept and reason (provide rationale and justification) that it is superior to the other known alternatives.
3. Provide high-level requirements.
4. Provide the criteria to be used for validation of the completed system.

The common deliverables for a 'concept of operation' formalization are:

1. Statement of the goals and objectives of the system (what is important?):
   A. Identify direction (e.g., expected impact as human fulfillment, desired result optimal quality of life, high-level conception of operation as habitat service system using a unified societal system.
   B. Establish objective priorities (e.g., establish societal-human priorities; identify human needs and requirements).
   C. Identify objective dependencies (e.g., identify the dependencies between conceptual objects, material objects, and their interrelationships through time as a matrix of dependencies, or dependency flow models as input-output

service modeling to fulfill all human needs).
2. Identify constraints affecting the system ("externally" environmental):
   A. Ecological [service flows and capacities]
   B. Resource [service flows from market, ...]
   C. Jurisdictional [service flows from State, ...]
3. Organizations, activities, and interactions among team.
4. Clear statement of roles and accountabilities ("responsibilities").
5. Specific operational processes for fielding the system.
6. Processes for initiating, developing, maintaining, and retiring the system.

Additional ConOps objectives include:

1. Provide end-to-end traceability between operational needs and captured source requirements.
2. Establish a high-level basis for requirements that supports the system over its life cycle.
3. Establish a high-level basis for test planning and system-level test requirements.
4. Support the generation of operational analysis models (use cases) to test the interfaces.
5. Provide the basis for computation of system capacity.
6. Validate and discover implicit requirements.

5.2 System concepts standards

The principal standards defining 'System Concepts' (as ConOps) are:

- **ISO/IEC/IEEE 29148-2018:** Systems and software engineering - Life cycle processes - Requirements engineering.
- **ISO/IEC/IEEE 15288:2015:** Systems and software engineering - System life cycle processes.
- **IEEE Std 1362-1998:** IEEE Guide for Information Technology—System Definition—Concept of Operations (ConOps) Document [standards.ieee.org/standard]
- **ISO/IEC/IEEE 29148:** Systems and software engineering. Life cycle processes. Requirements engineering
- **ANSI/AIAA G-043A-2012:** Guide to the Preparation of Operational Concept Documents
- **NASA NPR 7123.1B:** US NASA Systems Engineering Processes and Requirements (here, the definitions of ConOps and OpsCon are closely aligned with BSR/AIAA G-043A).
- **DI-IPSC-81430:** US DoD data item description for CONOPS document.

NOTE: *The first commonly known standard that defined the idea of a formalized Concept of Operation was IEEE 1362-1998 - IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps).*

The principal standards guiding the development of OpsCon are:

- **IEEE Standard 1362:1998:** IEEE Guide for Information Technology – System Definition – Concept of Operations Document
- **ISO 14711:2002(E)** - Space systems – Unmanned mission operations concepts
- **FHWA-HOP-07-001:2005** - Developing and Using a Concept of Operations in Transportation Management Systems, US Federal Highway Administration

5.2.1 Standards descriptions of ConOps

System concepts are defined via the aforementioned standards in the following ways:

- **ISO/IEC/IEEE 29148:** The ConOps, at the organization level, addresses the leadership's intended way of operating the organization. It may refer to the use of one or more systems, as black boxes, to forward the organization's goals and objectives. The ConOps document describes the organization's assumptions or intent in regard to an overall operation or series of operations of the business with using the system to be developed, existing systems, and possible future systems. This document is frequently embodied in long-range strategic plans and annual operational plans.
  The ConOps document serves as a basis for the organization to direct the overall characteristics of the future business and systems, for the project to understand its background, and for the users of [ISO/IEC/IEEE 29148] to implement the stakeholder requirements elicitation.
- **IEEE Std 1362-1998:** A Concept of Operations (CONOPS) is a user-oriented document that describes systems characteristics for a proposed system from a user's perspective. A CONOPS also describes the user organization, mission, and objectives from an integrated systems point of view and is used to communicate overall quantitative and qualitative system characteristics to stakeholders.
- **Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th Edition, INCOSE:** A Concept of Operations (ConOps) document is produced early in the requirements definition process to describe what

the system will do (not how it will do it) and why (rationale). It should also define any critical, top-level performance requirements or objectives (stated either qualitatively or quantitatively) and system rationale. Describes the way the system works from the operator's perspective. The ConOps includes the user description and summarizes the needs, goals, and characteristics of the system's user community. This includes operation, maintenance, and support personnel.

5.3 Concept of operations

A.k.a., Concepts of operation, ConOps, CONOPS.

Concept of Operations (ConOps) is a description of how the system will operate to meet user (operator) expectations for anything being conceptualized for the purpose of transforming that concept into reality. The ConOps includes the user description and summarizes the needs, goals, vision, and characteristics of the system's user. ConOps includes a description of the operation, maintenance, and support for the system. The ConOps describes the characteristics of a proposed system from the viewpoint of its users (and in a unified system, its operators also). It is used to communicate the quantitative and qualitative system characteristics to all stakeholders and serve as a basis for stakeholder unification on the issue. It often conveys a clearer statement of intent than the requirements themselves. The concept of operation describes the concept of the solution to meet the requirements. ConOps is a formal description of the likely operation of a future or existing system in the terminology of its users, providing important information for the development (or acquisition) of that system.

A ConOps makes all project team members aware of the different types of users of the system and activities those users will perform. This allows everyone who uses the document to get an idea of who is performing what task and in what order they are performing those tasks.

> **NOTE:** *A ConOps is a living document that is updated a changes occur. A ConOps is a directional document which can be used to compose an executive summary.*

ConOps is the first step in the engineering life-cycle for a new project [to develop a system]. The ConOps is a starting point for the more detailed description of the system. A system functional requirements specification follows the ConOps. Although the Concept of Operations is not a requirements document, a well-formed concept of Operations will be a primary source of information used to create the initial high-level functional requirements.

ConOps is part of the systems engineering life-cycle process, as seen below:

1. ConOps.
2. Requirements (high-level to detailed).
3. Design (high-level to detailed).
4. Implementation.
5. Integration and testing system verification.
6. Operation and maintenance.
7. Assessment.

The principal goal of the ConOps is to provide high-level definition of the system, including:

1. Identifying the required vision (mission) in functional terms.
2. The major parts within the envisioned system.
3. The flows of information among those parts, the information flow to any entities external to the system.
4. The high-level capabilities of the system.

Each capability in the ConOps needs units of measure meaningful for decisioning:

1. **Measures of effectiveness** - Operational measures of success that are closely related to the achievements of the vision or operational objectives evaluated in the operational environment, under a specific set of conditions.
2. **Measures of performance** - Characterize physical or functional attributes relating to the system operation, measured or estimated under specific conditions.
3. **Key performance parameters** - Capabilities and characteristics so significant that failure to meet them can be cause for re-evaluation, re-assessing, or termination of the project.

The ConOps (with OpsCon) provides the following:

1. An analysis and information set that bridges the gap between the users' needs and visions, and the developer's technical specifications.
2. A means of describing a user's operational needs without detailed technical issues that shall be addressed during the systems design analysis activity.
3. A mechanism for documenting a system's characteristics and the user's operational needs in a manner that can be verified by the user without requiring any technical knowledge beyond that required to perform normal job functions.
4. A place for users to state their desires, visions, and expectations without requiring the provision of quantified, testable specifications. For example, the users could express their need for a "highly reliable" system, and their reasons for that need, without having to produce a testable reliability

requirement.
5. A mechanism for users and operators (and buyers in the market) to express thoughts and concerns on possible solution strategies.

A ConOps document can be separated at a high-level into four major sections (or stages):

1. Describe current system (description).
2. Describe changes to make (description).
3. Describe proposed system (description).
4. Analyze proposed system (explanation).

In concern to the questions that provide a common context for any system, a ConOps/OpsCon answers the following:

1. **Who:** These describe the interactions among the various human elements within the system including their interfaces with people external to the system. The document and related scenarios should also identify decision points to include the organizational entity with authority to make those decisions. Other systems with which this system interoperates are also identified.
2. **What:** These are the known components or elements and top level capabilities required of the system, at its highest level of abstraction, to perform the necessary functions. The components are described from an operational point of view. Necessary mission phases or modes may also be described here. The Whats also include descriptions of the external systems with which the system of interest interfaces, and the external interfaces. Principal internal interfaces are also described.
3. **When:** These describe activities, tasks, flows, precedence, concurrencies, and other time / sequence related elements necessary to achieve the mission objectives in each of the various mission modes and conditions. Whens may also include information as to system development and operational availability dates, such as project milestones.
4. **Where:** These are the environments, such as geographical and physical locations of user's facilities and interfacing systems, within which the capabilities are required to be performed and supported. A description of the nature of the interfaces with other systems, organizations and the environment is also needed.
5. **How:** These tie together the other elements (the what, where, when, who, and why) to describe how the system is expected to be used, operated, maintained and, ultimately, retired in the given environment, under all significant conditions. The emphasis should be on concepts and should avoid any system design or implementation inferences.
6. **Why:** These provide the rationale behind any established partitioning of the mission tasks between the system components and the operators, and the reasoning for specific sequences of activities or tasks. For example, an important function of the document is to provide the rationale behind the definition of the level of technical expertise required of the system operators. This will provide a basis for the definition of a set of system requirements and designs with a consistent level of complexity and sophistication.

Generally, a ConOps includes all (or, some portion of) the following information sets:

1. **Introduction**
   A. **Document identification** - title and identification number.
   B. **Document overview** - an overview of the ConOps document.
      1. To communicate user needs and the proposed system expectations.
      2. To communicate the system developer's understanding of the user needs and how the system will meet those needs.
   C. **System overview** - a high-level overview of the proposed system in text and graphic.
   D. **Development effort** - a brief description of the scope of effort required.
2. **Applicable references and documentation** - what sources are mentioned. This section lists the document identification number, title, revision, and date of all documentation referenced in the ConOps.
3. **Current system/situation** - what is the problem to be solved, and the system or situation as it currently exists.
   A. **Background, objectives, and scope** - include as necessary all background, mission, objectives, and scope of the current system.
   B. **Operational constraints** - include descriptions on the operational characteristics of the existing system. This could include limits on fulfillment, hours of operation, hard/software limitations, and resource limitations.
   C. **Description of the current system or situation** - provide a thorough description of the current system, including but not limited to: operational characteristics, major system components, component interconnections, external system interfaces, current system functions, diagrams illustrating inputs, outputs, data flows. Include

a description of user classes and other people who interact with the system.

D. **User profiles** - describe who the users are and how they interact with the current system, and what happens when they do. Also discuss how the users interact with each other when using the system.

4. **Reasoning/justification for change** - why change is needed. Describe the issues, problems, gaps, shortcomings of the current system or situation that motivate development of a new system or modification of an existing system.
   A. **Reason for changes** - include the reasons for developing the proposed system, including the new, modified, or discovered user needs, goals, objectives; and dependencies or limitations of the current system.
   B. **Description of the desired changes** - include a summary of the new or modified capabilities, functions, processes, interfaces, and other changes needed to respond to the justifications previously identified.
   C. **Change priorities** - prioritize or rank the proposed changes. Specify what features are essential, what features are desirable, and what features are optional.
   D. **Changes considered but not included** - include significant changes or features that were considered but not included in the proposed system.
   E. **Assumptions and constraints** - describe assumptions or constraints applicable to the changes and new system features.
5. **New system description (concepts for new/ modified system)** - functional architecture and concepts for the proposed system; what is the proposed system that results from the desired changes specified in the fourth section of the ConOps.
   A. **Objectives and scope** - provide an overview of the new or modified system, including the mission, objectives, and scope. Focus on what about the vision or objectives is new that necessitated a new or modified system.
   B. **Operational policies and constraints** - describe the operational policies and constraints that apply to the proposed system.
   C. **Description of the proposed system** - provide a thorough description of the proposed system. The system description must be simple and clear enough that all intended readers can fully understand it. A high-level graphical overview of the system is strongly recommended.
   D. **Modes of operation** - describe the proposed system's various modes of operation. Examples of modes of operation (e.g., HSS operational processes) include: emergency/incident; maintenance and operations; planning; discretionary; strategic.
   E. **User involvement and interaction** - identify the users and the way they interact within the system.
   F. **Support environment** - describe the support and maintenance concepts, including the operating environment for the proposed system.
6. **Operational scenarios** - describe scenarios from different user's viewpoints. One or more operational scenarios that illustrate the role of the new or modified system, its interaction with users, its interface to other systems, and all states or modes identified.
7. **Summary of impacts** - Describes and summarizes the operational impacts of the proposed system from the users' perspective. This information is provided to allow all stakeholders to prepare the changes that will be brought about by the new system. Impacts include operation, organizational, and impacts during development.
8. **Analysis of proposed system** - summarize the benefits, limitations, advantages, disadvantages, and alternatives (and trade-offs) considered for the proposed system. In the context of a ConOps document, alternatives are operational alternatives and not design alternatives, except to the extent that design alternatives may be limited by the operational capabilities desired in the new system.
9. **Notes, glossary, supplemental material**

A ConOps may also include:

1. **Resource requirement estimate** - A rough order of magnitude resource estimate.
2. **Market cost estimate** - A rough order of magnitude cost estimate (ROM; market only).
3. **A schedule estimate** (with expected critical path).
4. A **project sequence of activities and events** - the sequence of tasks. (project plan concept).

Simplistically, ConOps answers the basic questions (for a new or existing system):

1. Who - who are the team involved in the system.
2. Why - what does the organization lack, and what will the system provide.
3. Where - what are the physical locations of the system.
4. When - what is the time-sequence of activities that will be performed by the system.

5. How - what resources are needed to design, build, and operate the system.

A ConOps is used to:

1. Provide a vision to guide to the development and operation of the system. A common vision about what is being built and operated.
2. Provide reasoning (justification) for and nature of changes.
3. Identify system stakeholders.
4. Assure a common communications reference.
5. Formulate and document a high-level system definition.
6. Foundation all lower-level sub-system descriptions.
7. Define all major user groups and activities.
8. Identify the environment in which the system will function.

5.3.1 System reasoning/justification

In order to provide a complete set of reasoning for a new system, the following questions should be answered:

1. Why is the new system need?
2. What is being changed?
3. What new functions do you get?
4. How does it change the environment?
5. What changes are needed to support the new system?
6. What are the most important changes (priorities)?
7. What changes are requested, but not included?
8. What assumptions and constraints are there in the system to be built?

5.3.2 Operational scenarios

A scenario is a step-by-step description of how the proposed system should operate and interact with its users and its external interfaces under a given set of circumstances. Scenarios are written in natural (layman's) language and should be non-technical as much as possible.

Scenarios should be structured so that each describes a specific operational sequence that illustrates the role of the system and its interactions with users and other systems. It may be necessary to develop several variations of each scenario, including one for normal operation, one for exception handling, one for degraded mode operation, etc. Each scenario will describe an operational event from the different user perspectives. Scenarios help the readers of a conOps document understand how all the pieces interact to provide operational capabilities.

Generally, a scenarios includes all of the following:

1. A description of the starting situation.
2. A description of the normal flow of events.
3. A description of what can go wrong.
4. Information about other concurrent activities.
5. A description of the state when the scenario finishes.

5.3.3 Operating environments

The various environments in which a system will be deployed, operate and be maintained include, but are not necessarily limited to:

1. Physical.
   A. Natural.
   B. Induced.
   C. Self-induced.
   D. Threat.
   E. Cooperative.
2. Social.
3. Technological.
4. State jurisdictional.
5. Market economic.

5.4 Types of OpsCon

OpsCon can be classified according to a system's life-cycle:

1. **Operations concept** - describes the way the system works from the operator's perspective.
2. **Production concept** - describes the way the system will be manufactured.
3. **Deployment concept** - describes the way the system will be delivered and installed.
4. **Support concept** - describes the desired support infrastructure and manpower considerations for maintaining the system after it is deployed. This includes specifying equipment, procedures, facilities and operator training requirements.
5. **Disposal concept** - describes the way the system will be removed from operation and retired.

OpsCon can also be classified according to who is composing and using the document:

1. User OpsCon - Written by users and operators, or by the developer in collaboration with the users and operators. Usually written prior to the commencement of development activity, but can be prepared at any point in the system life cycle. Defines the user's and operator's expectations for the system's operational capabilities.
2. System OpsCon - Written by developer personnel during or after the design activity defining how the system is to be used. Defines the developer's perception of how the system will be used.

3. Alternative OpsCon - Written during the concept exploration phase for each of the major alternative systems examined.
4. Remedial OpsCon - Written to redirect a Program that displays a lack of understanding of the overall system concept. It would typically be written at some point during the design phase.
5. Operations OpsCon - Written toward the end of the development Program to be maintained during the operations and support phase. It is written from the user and operator perspective and provides a representation of the system operations and capabilities as delivered.

6 [Engineering] Requirements

A.k.a., Requirements engineering.

A requirement is what the system must do to address the operative directive and satisfy the user of the solution (system). Requirements are the descriptions of the system services and constraints that are generated during the requirements engineering process. Requirements engineering (RE) is the process of defining, documenting, and maintaining requirements in the engineering design process. All requirements are statements using some measure that can be objectively tested. Requirements define what a system should (must, shall, etc.) do and define constraints on its life-cycle (e.g., development, implementation, operation, disposal, etc.). Simply, a requirement is a condition or capability to which a system must conform in operation and/or development. Requirements are that which is necessary for a system to function as intended and designed.

> **CLARIFICATION:** *Requirements are different than goals, and other user directive statements, such that requirements can be objectively tested, whereas goal statements may not necessarily be stated in such a way that they can be objectively tested.*

Requirements range from high-level abstract statements of a service or of a system constraint, to detailed mathematical functional specification.

> **IMPORTANT:** *At a societal engineering level, [human] requirements define the societal direction, objectives define the societal orientation (values), and methods define the societal approach.*

Requirements are often expressed as "shall" statements. Requirements are level dependent; for example, there are system requirements (top-level) and subsystem, or component (bottom-level) requirements.

Fundamental problems arise when requirements are not properly stated. Ambiguous requirements may be interpreted in different ways (by developers and users).

> **NOTE:** *In the market, requirements serve as the basis of all contracts, including all bids for contracts.*

Requirements should be (i.e., attributes of requirements are):

1. **Complete** - describe everything required. Each requirement describes one result that must be achieved by the product. A requirement should not be redundant. The requirement should not describe the means of obtaining the result. Are all functions and conditions required included?
2. **Consistent** - no conflicts and no contradictions. Individual requirements are not in conflict with

other requirements. Are there any requirements conflicts or contradictions?
3. **Necessary** - Absolute requirements that are to be verified are identified by "must" or "shall". Goals or intended functionality are indicated by "will".
4. **Correct** - Each requirement is an accurate description of a feature or process of the product.
5. **Unambiguous (clear)** - The statement of each requirement denotes only one interpretation. Can the requirement be fully understood?
6. **Realistic (feasible)** - Can the requirement be implemented given available knowledge, resources, persons, and technology? Can a real world solution be built and tested to prove that the requirement is satisfied?
7. **Verifiable (testable)** - Each requirement is stated in concrete terms and measurable quantities. A process should exist to validate that the product (when developed) will satisfy the set of requirements. Can the requirement be checked? Is the requirement realistically testable?
8. **Modifiable** - The structure and style of the requirements are such that any necessary changes to the requirements can be made easily, completely, and consistently.
9. **Traceable** - The origin of each requirement is clear and can be tracked in future development activities. Is the origin of the requirement clearly stated?

6.1 Engineering design requirements

The identification of a projected system requires a set of descriptive engineering design requirements (a.k.a., engineering requirements). Requirements integrate needs and objectives into a set of instructions (i.e., requirements) that the design has in some priority (i.e., requirements are structured). Requirements describe:

1. What functions the system is supposed to provide (what the system does)?
2. What characteristics the system is supposed to have (what the system['s quality] is)?
3. What goals the system is supposed to meet or to enable users to meet (what use is the system)?

A requirement is another type of [directional] input (imperative) into a project, more specific than and informed from needs, objectives, and goals. In order to coordinate action and access among a team, it is imperative to describe what the system is supposed to do. All requirements have rationales that logically relate requirement to a prior imperative, and must consider:

1. What is an aspect of this requirement that could be a source of confusion?
2. How is the potential confusion addressed in the requirement?
3. What is the evidence that informs the resolution of the confusion?
4. What other requirements might have some effect on the interpretation and implementation of the requirement, and thus should be referenced in the rationale?

A requirement is an engineering input composed of a statement of a need or objective, or of a condition or capacity, that a system or product must possess to satisfy a prior need or objective. Therein, a requirement is a property that a system or product must have to provide usability (or functionality) to a user. Requirements are the start of tasks (i.e., the instantiation of tasks), and the first phase of real world issue resolution. Every requirement inherently asks, "How will a successful (or complete) implementation of this requirement (a specific description of some thing in the real world) be verified?"

A requirement is visualized in a table with [at least] two related columns:

1. The system must have x.
2. Because of y.

A system requirement list explains why a system, product or service is needed, puts the system in context, and describes what the finished system will be like and/or what it will do. During engineering, the answers to how questions fall into the realm of design, the next sequential phase after initial requirements are developed. Thus, requirements specifications should not include design solutions (except for interface requirements, which often include embedded design).

One or more requirements forms a 'requirements set/list'. A requirements list is *formalized* (integrated into a unified model of the system). The requirements list is structured hierarchically.

During engineering, requirements are the basic source for communication among end-users, InterSystem Teams, and intelligent systems.

> *Note: A [problem] domain-model is an abstraction that defines the structure and behavior of the problem domain.*

In the real world, there are several potential problem domains. There are community and market concepts involved in the life cycle modeling of requirements:

1. In community, there are two types of concepts involved in life cycle requirements modeling:
   A. **Human [conceptual] objectives**, which represent the problem domain, and are emphasized in the requirements model.
   - **Engineering concepts**, which are emphasized in the design model(s).

2. In the market, there is one additional type of concepts involved in life cycle requirements modeling:
   A. **Business concepts** related to customers' objectives, which represent the problem domain, and are emphasized in the requirements model, though expressed in the design.

Requirements provide a tool for evaluating the final results of the project by examining whether each requirement has been met. Every rule and functional relationship provides a test point. Note that requirements tend to change through the course of a project, with the result that the final output, as delivered, may not adhere to the initial version of the requirements.

A 'requirement':

1. *Should specify* the expected behavior and/or form, through a detailed analysis of that which is required for creation of the new status/ state [iteration] and/or new product. Generally, requirements are statements of *what* a system should do, rather than *how* it should do it (which is present in the design).
2. Is defined as, *What is needed?* A requirement is, *a well-defined need*.
3. Is *an objective that must be met*. Requirements define *necessary objectives*.
4. Contains criteria for completion, or is *testable*.
   A. *Performance* is the degree to which requirements are met.
5. As a 'list' *includes*, descriptions of system properties, specifications for how the system should work, and constraints placed upon the development and designed operating process.

The attributes of good requirements include the following:

1. **Achievable** - A requirement must be achievable. It must reflect need or objective for which a solution is technically achievable at costs considered affordable.
2. **Verifiable** - The expected performance and functional utility must be expressed in a manner that allows verification to be objective, preferably quantitative; that is, not defined by words such as excessive, sufficient, resistant, etc.
3. **Unambiguous** - A requirement must be unambiguous. It must have but one possible meaning.
4. **Complete** - It must be complete and contain all mission profiles, operational and maintenance concepts, utilization environments and constraints. All information necessary to understand the customer's need must be there.
5. **Causative** - It must be expressed in terms of need, not solution; that is, it should address the "why" and "what" of the need, not how to do it.
6. **Consistent** - It must be consistent with other requirements. Conflicts must be resolved up front.
7. **Appropriate** - It must be appropriate for the level of system hierarchy. It should not be too detailed that it constrains solutions for the current level of design. For example, detailed requirements relating to components would not normally be in a system-level specification.

Project requirements (or just, requirements) are conditions and capabilities that must be met, or tasks that must be completed, for the project to be complete. Requirements require identifying, defining, organizing, documenting, and refining. A 'Requirements Specification' (a.k.a., Requirements Definition Document) documents requirements as a specification. Requirement become technical specifications (composed of material and/ or information properties that feed back upon us, influencing our experience. Any engineer/programmer can build one from this document.

Every requirement must be testable. To know when a project is complete, every requirement must have been tested as complete. If a requirement is not testable, then how will a complete (successful) implementation of the requirement be verified. The requirement must answer, "How do "you" the requirement has been completely implemented and works as expected?"

In application, a given system's requirements will have the following set of control characteristics:

1. **Defined -** Define a model of the system to be built; not [a model of] the system [itself].
   A. Define some mixture of functionality, behavior, performance, and systems constraints (non-functional requirements).
   B. Defines known constraints.
2. **Organized** - Organized by functionality and logical layout.
3. **Tested** - Every statement is verifiable, with level and nature of test as attributes.
4. **Assigned** - Assigned to InterSystem Teams.
5. **Opened** - Viewable to everyone.

In concern to a community-type societal system, what the engineered system does is directly perceived and independently experienced by its users – either human users or other systems. When a user performs some action, the societal system responds in a particular way; when an internal system or user submits a request of a certain form, it gets a particular response. Due to the nature of societal systems being composed

of the users themselves, the users must agree on actions they can perform and responses they should expect from the societal system. This common understanding is captured in the requirements.

> **INTERSYSTEM TEAM:** *Requirements engineering is the "sub-discipline" of systems engineering that encompasses all project activities associated with understanding a system or product's necessary capabilities and attributes, including both requirements development and coordination.*

Once the system is in operation, a new societal requirements is either:

1. A specified design change in the status or state of the societal system;
2. Or, a newly designed [habitat] service or product.

Here, requirements include only real requirements to the system (service-product), and exclude requirements to the project or any other ancillary information.

> **APHORISM:** *It is from requirements that engineering can proceed. Because it is not possible to have an acceptable system even with the best solution space if this is based on an incorrect problem space formulation.*

The concept of system existential categories, which correspond to the following requirement types:

1. Functional requirements (Do): Requirements that define what the system must do. In other words, what it accepts and what it delivers (i.e., expected transformation). Examples: The system shall provide food; The system shall transmit 4 signals; The system shall convert sea water into drinkable water.
2. Performance requirements (Being): Requirements that define how well the system must operate, which includes performance related to functions the system performs or characteristics of the system on their own, such as -illities. Examples: The system shall move at a speed higher than 30 km/h; The system shall have a reliability better than 0.80.
3. Resource requirements (Have): Requirements that define what the system can use to transform what it accepts in what it delivers. Examples: The system shall consume less than 300 W; The system shall have a mass of less than 30 kg.
4. Interaction requirements (Interact): Requirements that define where the system must operate, which includes any type of operation during its life-cycle.

> **Examples:** *The system shall withstand shock levels higher than 300 g; The system shall operate in vacuum (to reflect operation); The system shall operate in clean room class 'X' (to reflect Assembly, Integration, and Test activities).*

Each level of the requirements hierarchy represents a fully operable system as they are options that build upon previous need levels. The amount of levels is unlimited and free of preconceptions, being therefore up to each project to define theirs.

Herein, the threshold-based concepts are:

1. Base Threshold: the minimum level of service (value) that must be provided so that the system is acceptable.
2. Goal Threshold: desired value provided by the system.
3. Want Threshold: great-to-have, but considered difficult to achieve.

Functional ("do") requirements include:

1. The system shall service the spectral range of human need.
   A. The system shall provide water services to ...
   B. The system shall provide energy services to ...
   C. The system shall provide building services to ...
   D. The system shall provide medical services to ...
   E. The system shall provide production and material cycling services to ...
   F. The system shall provide to ...

Performance ("being") requirements:

1. The service system shall have an efficiency better than 98%.
2. The service system shall have a reliability higher than 0.90.

Resource ("have") requirements:

1. The system shall fit within a circular boundary.
2. The system shall have a human carrying capacity

> **NOTE:** *Carrying capacity is a limit that varies with technology.*

Interaction ("have") requirements:

1. The system shall fulfill its performance requirements in the manner specified by the modularity (or other) standard (protocol).
2. The system shall fulfill its performance requirements within the carrying capacity of the larger ecosystem

The International Council of Systems Engineering (INCOSE) [2011] proposes an independent classification of requirements that targets any complex system and that includes:

1. Functional requirements.
2. Performance requirements.
3. Non-functional requirements.
4. Architectural constraints.

Hull et al. (2005) make a similar contribution in the field of software systems and define

1. Functional requirements.
2. Performance requirements.
3. Quality factor requirements.
4. Environment requirements.
5. Interface requirements.
6. Constraint requirements.

Requirements exist on design attributes, on the existence of objects and characteristics, on the relationships, and on functions. Their proposition confirms a designer perspective when eliciting requirements: How the system has to be designed.

Function requirements, which indicate what the system must do:

1. Performance requirements, which define how well the functions of the system must perform;
2. Resource requirements, which define the resources that are available to create and maintain the functions and performance of the system (explicitly referring only to money, people, and time);
3. Design constraints, which define restrictions on the solution;
4. Condition constraints, which define restrictions on the use of the system.

Providing a domain-independent classification:

1. Input/output requirements.
2. Technology requirements.
3. Performance requirements.
4. Cost requirements.
5. Trade-off requirements.
6. System test requirements.

Medical industry a matrix classification of 5 domains as categories of requirements:

1. Process.
2. Performance.
3. Safety.
4. Cost.
5. Documentation.

Seven stages of the system life-cycle as categories of requirements:

1. Design.
2. Manufacturing.
3. Distribution.
4. Installation/assembly/integration.
5. Operation.
6. Maintenance.
7. Recycle.

6.1.1 Requirements breakdown structure (RBS)

The RBS is different than the WBS. The RBS is grouped logically, and the WBS is grouped into physical work packages for the configuration of items that need to be developed. The information in the RBS flows into the WBS.

Requirements functional flow block diagram (flows between functions are seen, not just the hierarchical relationships between functions) - provides information on the sequencing (parallel and series) of functions.

Common requirement framework characteristics include:

1. A mission statement with 5-7 key concepts. Each of those concepts if detailed (fleshed out) at the next level leading to 5-7 goal statements, each of which contains 5-7 concepts, fleshed out at the next level as objectives, each of which has 5-7 concepts, and the process continues until we reach the “leaves” of the tree.
2. Numbering requirements - The usage of a number system allows each level to be associated with the levels above and below.

6.1.2 Requirement categorization approach

The NASA approach to categorizing project requirements:

1. **Technical requirements** (functional requirements, performance requirements, and interface requirements).
2. **Operational requirements** (mission, configuration, and command and telemetry).
3. **Reliability requirements** (environment, fault tolerance, verification, and process and workmanship).
4. **Safety requirements**.
5. **Specialty requirements** (maintainability, producibility, etc.).

The European Space Agency approach:

- Functional Requirements, Mission Requirements, Interface Requirements, Environmental Requirements, Operational Requirements, Human Factor Requirements, (Integrated) Logistics Support Requirements, Physical Requirements, Product Assurance Requirements, Configuration

Requirements, Design Requirements, and Verification Requirements

In engineering, the classification of requirements should effectively describe the configuration of the output, the resulting system specification. The classification of requirements facilitates the design of the system, which implies influencing the design and selection of a solution. However, at the user level, the required imperative (goal of service) should rather specify what the system is intended to do.

6.2 Requirements as objectives

NOTE: *The way in which requirements are categorized can impact system affordability.*

Once a need has been recognised and identified, then resources are allocated to the development of a design for its fulfillment, and the 'engineering/design task' is initialized ("born").

The objectives of requirements are:

1. Completely define a system by means of defining all elements necessary to complete what the system is intended to achieve, shall fit within the proposed categories.
2. Identify requirements that are not applicable to the system to be developed, but to ancillary elements, such as supporting systems (or market contractual agreements).
3. Identify constraints that do not support the satisfaction of user needs, but that limit the solution space, thus facilitating the definition of boundaries for the solution and eliminating any influence on a specific solution.

A system is completely defined by specifying:

1. What systems do,
2. How they are (how well they do),
3. What they use,
4. Where they live.

All sub-systems are elements that form requirements that define:

1. What the system has to do?
2. In what context the system has to do it?
3. How well the system has to do it?
4. Which resources the system can use to do it?

Examples of requirements as objectives include:

1. Adaptability needs:
   A. Can you upgrade and modify it?
   B. Sub-conceptions:
      1. Flexibility, modularity, scalability, etc.
2. Operational effectiveness (readiness) needs.
   A. With what does it operate, how does it operate?
3. Efficiency needs:
   A. Is it intuitive and does it operate well?
   B. Sub-conceptions:
      1. Use of resources, process efficiency
4. Availability needs :
   A. How often does it fail?
   B. Sub-conceptions:
      1. Reliability, maintainability, supportability, etc.
      2. Durability = repairability and maintainability

6.3 Requirements as metrics

Metrics are a means of identifying whether an individual atomic requirements statement or an entire requirements set (requirements document as a whole) has been met and/or is in the progress of being met. Requirements are identified with standardized names and a method of both subjective and objective measuring.

There are three primary categories of metrics in terms of requirements:

1. **Requirements traceability (Traceability metrics)** - Is the set of requirement(s) internally traceable, with clear associations, and no conflict between individual requirements?
2. **Requirement consistency (Consistency metrics)** - Is the set of requirement(s) internally consistent, with no contradictions, no duplication between requirements?
3. **Requirements falsibility (Falsibility metrics)** - How adequately can this requirement be tested? Is it clear what test(s) are needed to confirm the requirement is met? Is it clear what should be considered a failure of a test of this requirement?
4. **Requirements visualizability (Verifiability metrics)** - How adequately can this requirement be visualized in object form? Is it clear what objects and relationships are needed to understand the requirement? Is it clear what is not a visualization of this requirement?

6.4 Requirements list

A.k.a., Requirements specification.

A requirements specification should include:

1. Definition of the function or entity.
2. Description of inputs and where they come from.
3. Description of outputs and where they go.
4. Information about the information needed for the