computation and other entities used.
5. Description of the action to be taken.
6. Pre and post conditions (if appropriate).
7. The side effects (if any) of the function.

6.5 Systems engineering and requirements

Requirements are the primary focus in the systems engineering process because the process's primary purpose is to transform the requirements into designs. The engineering development process develops these designs within the constraints. They eventually must be verified to meet both the requirements and constraints.

> **NOTE:** *The primary evaluation of "success" of a system is the degree to which it meets the purpose for which it was intended.*

Requirements engineering is the process of:

1. Discovering the purpose for the system by identifying users and their needs, and
2. Documenting these in a form that is agreeable to analysis, communication, and subsequent implementation.

Requirements engineering is a set of activities concerned with identifying and communicating the purpose of a system, and the context in which it will be used. RE acts as the bridge between the real-world needs of users, customers, and other constituencies affected by a system, and the capabilities and opportunities afforded by technologies.

> **INSIGHT:** *Real-world goals [ought to] motivate the development of a system.*

Typical definitions of engineering refer to the creation of effective solutions to practical problems by applying scientific knowledge. Therefore, the use of the term engineering in RE serves as a reminder that RE is an important part of an engineering process, being the part concerned with anchoring requirements activities to a real-world problem, so that the appropriateness and effectiveness of the solution can then be analyzed. It also refers to the idea that specifications themselves need to be engineered, and RE represents a series of engineering decisions that lead from recognition of a problem to be solved to a detailed model of that problem.
The primary requirements engineering activities are:

1. Eliciting requirements - identifying, articulating, or otherwise defining requirements by asking the right questions.
2. Analyzing and modeling requirements.
3. Communicating requirements.

The identification of the problem that needs to be solved leads to identification of a system's boundaries. These boundaries define, at a high level, where the final delivered system will fit into the current operational environment. The identification of user classes, of goals and tasks, and of scenarios and use cases all depend on how the boundaries are selected.

6.5.1 Types of system Requirements

Requirements are categorized in several ways. The following are common categorizations of requirements that relate to technical management:

1. **User requirements:** Statements of fact and assumptions that define the expectations of the system in terms of mission objectives, environment, constraints, and measures of effectiveness and suitability. The customers are those that perform the eight primary functions of systems engineering , with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum.
2. **Functional Requirements:** The necessary task, action or activity that must be accomplished. Functional (what has to be done) requirements identified in requirements analysis will be used as the top-level functions for functional analysis.
3. **Performance Requirements:** The extent to which a mission or function must be executed; generally measured in terms of quantity, quality, coverage, timeliness or readiness. During requirements analysis, performance (how well does it have to be done) requirements will be interactively developed across all identified functions based on system life cycle factors; and characterized in terms of the degree of certainty in their estimate, the degree of criticality to system success, and their relationship to other requirements.
4. **Design Requirements:** The "build to," "code to," and "buy to" requirements for products and "how to execute" requirements for processes expressed in technical data packages and technical manuals.
5. **Derived Requirements:** Requirements that are implied or transformed from higher-level requirement. For example, a requirement for long range or high speed may result in a design requirement for low weight.
6. **Allocated Requirements:** A requirement that is established by dividing or otherwise allocating a high-level requirement into multiple lower-level requirements. Example: A 100-pound item that consists of two subsystems might result in weight requirements of 70 pounds and 30 pounds for the two lower-level items.

6.5.2 Requirements analysis

Requirements analysis involves defining customer needs and objectives in the context of planned customer use, environments, and identified system characteristics to determine requirements for system functions. Prior analyses are reviewed and updated, refining mission and environment definitions to support system definition. Requirements analysis is conducted iteratively with functional analysis to optimize performance requirements for identified functions, and to verify that synthesized solutions can satisfy customer requirements. The purpose of Requirements

Analysis does:

1. Refine customer objectives and requirements.
2. Define initial performance objectives and refine them into requirements.
3. Identify and define constraints that limit solutions.
4. Define functional and performance requirements based on customer provided measures of effectiveness.

In general, requirements analysis should result in a clear understanding of:

1. Functions: What the system has to do.
2. Performance: How well the functions have to be performed.
3. Interfaces: Environment in which the system will perform.
4. Other requirements and constraints.

The understandings that come from requirements analysis establish the basis for the functional and physical designs to follow. Good requirements analysis is fundamental to successful design definition.

Requirements analysis is a process of inquiry and resolution.

1. User requirements.
2. Design requirements (prioritize and structure).
3. Target values (benchmarking) against target values or what is expected.
4. Collaborative design and process planning - match capabilities to requirements; what capabilities are available? What capabilities must be developed?

Common requirements analysis questions include, but are not limited to:

1. What are the reasons behind the system development?
2. What are the user expectations? What do the users expect of the system?
3. Who are the users and how do they intend to use the system?
4. What is the user's level of knowledge, skill, expertise?
5. With what environmental characteristics must the system comply?
6. What are existing and planned interfaces?
7. What functions will the system perform, expressed in user language?
8. What are the constraints (hardware, software, economic, procedural) to which the system must comply?
9. What will be the final form of the product: such as model, prototype, or mass production?

The requirements that result from requirements analysis are typically expressed from one of three perspectives or views. These have been described as the Operational, Functional, and Physical views. All three are necessary and must be coordinated to fully understand the users' needs and objectives. All three are documented in the decision database.

1. **Operational view** - The Operational View addresses how the system will serve its users. It is useful when establishing requirements of "how well" and "under what condition." Operational view information should be documented in an operational concept document that identifies:
   A. Operational need definition.
   B. System mission analysis.
   C. Operational sequences.
   D. Operational environments.
   E. Conditions/events to which a system must respond.
   F. Operational constraints on system.
   G. Mission performance requirements.
   H. User and maintainer roles (defined by job tasks and skill requirements or constraints).
   I. Structure of the organizations that will operate, support and maintain the system.
   J. Operational interfaces with other systems.
2. **Functional view** - The Functional View focuses on WHAT the system must do to produce the required operational behavior. It includes required inputs, outputs, states, and transformation rules. The functional requirements, in combination with the physical requirements shown below, are the primary sources of the requirements that will eventually be reflected in the system specification. Functional View information includes:
   A. System functions.
   B. System performance.
      1. Qualitative — how well?
      2. Quantitative — how much, capacity?
      3. Timeliness — how often?

C. Tasks or actions to be performed.
D. Inter-function relationships.
E. Hardware and software functional relationships.
F. Performance constraints.
G. Interface requirements including identification of potential open-system opportunities (potential standards that could promote open systems should be identified).
H. Unique hardware or software.
I. Verification requirements (to include inspection, analysis/simulation, demo, and test).

3. **Physical view** - The Physical View focuses on HOW the system is constructed. It is key to establishing the physical interfaces among operators and equipment, and technology requirements. Physical View information would normally include:
   A. Configuration of System:
      1. Interface descriptions,
      2. Characteristics of information displays and operator controls,
      3. Relationships of operators to system/physical equipment, and
      4. Operator skills and levels required to perform assigned functions.
   B. Characterization of Users:
      1. Handicaps (special operating environments),
      2. Constraints (movement or visual limitations).
   C. System Physical Limitations:
      1. Materials limitations (capacity, power, size, weight).
      2. Technology limitations (range, precision, data rates, frequency, language).
      3. Government Furnished Equipment (GFE).
      4. Commercial-Off-the-Shelf (COTS).
      5. Non-developmental Item (NDI), reusability requirements.
      6. Necessary or directed standards.

Requirements are system and project level data sets (or, issues:

1. Requirements are the design decisions about what the system will do.
2. Requirements are the set of things that we have decided should matter and be completed by the conclusion of the project.

The properties of the system that we have decided to define and control (manage) through the engineering process. Not all properties of a system are requirements. Requirements are not a statement of intent or a directive, they are not the users needs, they are what a specific system, with specific system boundaries, is actually going to do.

6.5.3 Requirements analysis through prioritization

INSIGHT: *Motion requires input, input is constrained, therefore motions are prioritized.*

Requirements prioritization is a decisioning process. Requirements necessitate prioritization because they concern limitation.

During requirements triage, relative priorities are established for requirements, and resources needed for their achievement are identified and assessed. Then requirements are packed in subsets, and each subset is evaluated against the probability of such subset being a success.

Methods for establishing the prioritization of requirements include:

- Scale of rankings (e.g., 1-4; must, should, could), voting schemes, weightings, value-based (i.e., user-based given available resources), etc.

Prioritization categories include:

1. **Must have** requirement (mandatory, shall).
2. **Should have** if at all possible (high importance).
3. **Could have** but not critical (low importance).
4. **Will not have** this time (delayed importance, does not matter).

NOTE: *This prioritization scheme parallels the Habitat Service System's operational decisioning prioritization process (Criticality Response).*

The design of the habitat service system naturally breaks down into a series of criticality systems, of which, life support is of the highest prioritization. Herein, facility systems (another top-level habitat service system) is a could have, but not critical.

There are different ways of approaching prioritization, which vary (at least) by type of requirement:

1. Market requirements include:
   A. **Financial requirements** will determine financial constraints ("budget"). Financial constraints determines resources purchased.
2. Real-world requirements include:
   A. **Need requirements** will determine service constraints. Service constraints determine functions selected.
   B. **Material requirements** will determine material constraints. Material constraints determine materials selected.
   C. **Social (navigational) requirements** will determine decision constraints. Decision constraints determine the new state of the

habitat.

3. Technology access includes:
   A. Technology readiness matrix.
   B. Technology integration matrix (integrated simulating system).
   C. Technology material composition cost table.

6.5.4 Requirements analysis through evaluation (Quality management)

The evaluation of requirements is carried out under quality control/management. Procedures used together for checking that a system (service or product) meets requirements and specifications, and that it fulfills its intended purpose.

> **NOTE:** *Requirements evaluation is a critical component of a quality management system (e.g., ISO 9000).*

Requirements are capable of evaluation because they are:

1. Requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.
2. Specified standards define a set of development criteria that guide the manner in which a system is engineered. If the criteria are not followed, lack of quality will likely result.

The factors that affect quality can be categorized in two broad groups:

1. Factors that can be directly measured (e.g., defects per function-point).
2. Factors that can be measured only indirectly (e.g., usability or maintainability).

In each case, measurement must occur. We must compare the system (documents, programs, data) to some datum and arrive at an indication of quality. Quality factors focus on three important aspects of a product:

1. Its operational characteristics.
2. Its ability to undergo change.
3. Its ability to adaptability to new environments.

6.5.5 Engineering assurance

> *A.k.a., Engineering certainty, quality assurance, systems engineering structured assurance, project assurance, systems evaluation, qualification, examination, acceptance, requirements assurance, quality assurance.*

Verification and validation (V&V) mean the same thing within a non-technical context, but in the framing of simulation quality they have quite specific technical meanings. Each involves the accumulation of evidence that correctness (alignment) is present.

> **IMPORTANT:** *Verification and validation rely on a source's ability to specify the objective(s) correctly (accurately and fully).*

Validation and **verification** are prerequisite to sufficient user acceptance of a new system. Verification and validation can processes can be applied [at least] to models and to systems engineering.

The model view of validation and verification:

1. **Verification** is the determination of whether the model (e.g., specified requirement) is being solved correctly.
2. **Validation** is the determination of whether the model (e.g., specified requirement) is correct.
   A. Validation necessarily involves observational or experimental data, and its comparison to the simulation (e.g., operating system).
   B. A necessary observation is that validation involves several error modes that color any comparison:
      1. The size of the numerical error in solving the model.
      2. The magnitude of the experimental or observational error.

The systems engineering assurance views:

1. **Verification** - testing to confirm the system and its performance align with the specification/ requirements. Confirm or dis-confirm (and to what degree) a system as aligning with its specified requirements. System verification is assuring that the system is built right.
   A. **Evaluation** *(design view; a.k.a., assessment)* - whether or not a system complies with specified requirements or imposed conditions. Evaluation questions may include: How is the requirement verified (confirmed)? How will testing demonstrate proof of correctness? Has the system been built right for the user; is the system verified (or dis-confirmed)? System evaluation is assuring quality (a.k.a., quality assurance) and function (a.k..a, functional evaluation/assessment). For instance, what is the baseline of operation, and was it met?
   B. **Testing** (development view) - whether or not a system reliably complies with specified requirements or imposed conditions. The two types classified by their effect on the system include: non-destructive examination (NDE) and destructive examination (DE).
2. **Validation** *(user view)* - user confirmation of

requirements completion. Has the right system been built for the user? System validation is assuring that the right system is built for the intended user environment.

6.5.5.1 System requirement engineering

> **NOTE:** *It is normal to find faults with a design after a period of operation.*

Systems engineering is used to realize viable systems that satisfy user needs.

1. **Iterative** - the repeated application of a process to the same system or sub-system to correct/ solve a discovered discrepancy or variation from requirements (apply the process again and again until correction is complete).
2. **Recursive** - the repeated application of a process to design the next lower layer (or level) of the system, or to realize the next higher integrated layer (or level) of the system.

6.5.5.2 System verification

System verification requires the input of a system definition:

1. **The definition verification process:** compare the definition of the system, and the system's design specification, and show that the system design specification meets, or does not meet, the system's [objective] definition.
   A. If it is not possible, given the information available, to match the system's behavior (as a design specification) to its definition (Read: its model), then scientific inquiry is required -- all that can be done is to do an experiment to see if the system observably behaves like the model (Read: the definition).
2. **The evaluation process** - a mechanisms that provide a designer with critical feedback on the usability, feasibility, etc. of the system.

The three primary engineering design and development problems for a system are:

1. Describe what the system does.
   A. What does the system do?
   B. What is the system's purpose, function, objective, operation, utility?
2. Describe pre-conditions for the systems operation (I.e., for using the product).
   A. What does the system require to operate?
   B. Under what environmental conditions will the system operate?
3. Describe the system's interfaces (material, visual, logical, mathematical, etc.).
   A. With what, and how, does the system interface?

> **NOTE:** *Development involves a creation (analysis-synthesis) life cycle based on evolving prototypes, and the evolution of the development method itself.*

6.5.6 Requirements management

Requirements "management" is the process by which changes to requirements are decided and remembered throughout the system life-cycle. Requirements change because:

1. Knowledge develops.
2. User requirements change.
3. Organizational value-set changes.
4. The environment changes.

> **NOTE:** *It is almost impossible to have requirements traceability without implementing the requirements in some automated context. Therein, a requirements coordination tool (visual interface, database, and processing) is generally necessary to assist in the coordination of a large number of requirements.*

Requirements interface support (i.e., a requirements coordination tool functions to):

1. Supports elicitation
2. Support access by means of browse, find, retrieve, and generate reports of requirements based on selected criteria.
3. Supports forward and backward traceability.
4. Supports the generation of correct linguistic and logical requirements.
5. Supports change control and change impact assessment
6. Supports functional allocation and functional-to-physical translations.
7. Does not enforce any particular requirements engineering process.

6.5.6.1 Requirements hierarchy

A hierarchy of requirements with system requirements leading to sub-system requirements. Traceability within the requirements hierarchy is essential so that requirements always have a causative presence. In systems engineering the terms 'forward' and 'backward' traceability provide a awareness of direction (and how they relate) within the hierarchy.

1. Forward traceability is from the system level requirement(s) to the sub-system level requirement(s).
   A. Are the system's requirements met by the sub-system's design?
2. Backward traceability is from the sub-system level

requirement(s) to the system level requirement(s).
   A. Are the sub-systems able to meet their requirements, and if not, what system level requirement may be at risk [of not being met]?
   B. Is there requirements "creep" occurring, where sub-system requirements are being created for non-existent system requirements?

These can be:

1. Functional requirements - what is the thing going to do.
2. Performance requirements - how well is the thing going to do it.
3. Resource requirements - how many resources does the thing need to do it.

Requirements are:

1. Conceived.
2. Allocated.
3. Executed.
4. Closed.

Requirement information need:

1. Information category.
2. Measurable concept.
3. Leading insight.

6.5.6.2 Requirements engineering

Requirements engineering represents a series of engineering decisions that lead from recognition of a problem to be solved to a detailed specification of that problem and its resolution.
Requirements [engineering]

1. Articulating requirements.
2. Modeling and analyzing requirements.

The two most common characteristics of requirements are that they:

1. Requirements may have interdependencies.
2. Requirements are organized in subsets that hierarchically map value to users.

6.5.6.3 Requirements engineering tools

There are a large number of tools that may assist in requirements engineering, including:

1. Context diagram.
2. Functional flow block diagrams.
3. Requirements breakdown structure (RBS).
4. N2 diagrams.
5. Structured analysis.
6. Data flow diagrams.
7. Control flow diagrams.
8. IDEF diagrams.
9. Behavior diagrams.
10. Action diagrams.
11. State/mode diagrams.
12. Process flow diagrams.
13. Functional hierarchy diagrams.
14. State transition diagrams.
15. Entity relationship diagrams.
16. Structure analysis and design.
17. Object-oriented analysis.
18. Unified modelling language (UML).
19. Structured systems analysis.
20. Design methodology.
21. Quality function deployment.

6.6 System requirement constraints

Both resources (material boundaries) and constraints (information boundaries), as well as time, are elements a system uses for transforming inputs into outputs. Cost and schedule limit the solution space (in market, "tradespace") and as such traditional categorization of requirements include them as requirements or constraints. Development cost and schedule can be perceived as resources because they are consumed during system development. However, these resources are not consumed during development by the system, but by the project developing the system, and therefore they would actually reflect project constraints and not system ones. On the other hand, it could also be argued that time and cost are indeed consumed by the system during its creation, which would bring them back as resource requirements to the system.

Consequently, the present research proposes to allocate development cost and schedule requirements in one of the following two categories, depending on the vision and needs of each project:

1. System development requirements - requirements defined for the system's development.
   A. System development resources - What resources are consumed and/or cycled by the system during its development?
2. System operation requirements - requirements defined for the operational phase of the system, i.e., how much money is required to operate the system at the specified performance levels. Operational cost requirements inherently belong to the resource category, as it is something a system uses to fulfill its functions.
   A. System operational resources - What resources are consumed and/or cycled by the system during its operation?

6.7 Requirement expression: standards (Categorical, linguistic)

A requirement is an imperative. Other imperatives include needs, goals, directives, and objectives. Statements in this plan contain the following imperatives:

1. *Shall* are used for binding requirements that must be verified and have an accompanying method of verification. Shall is a binding provision.
2. *Will* is used as a statement of fact, declaration of purpose, or expected occurrence. Will is a declaration of purpose.
3. *Should* denotes an attribute or best practice that must be addressed by the system design.
4. *May* denotes a non-binding attribute or provision.
5. *Must* denotes the expression of either a constraint, a certain quantity, or a performance requirement (non-functional requirement).

Principles for usage include:

1. Use exactly one provision or declaration of purpose (such as shall) for each requirement, and use it consistently across all requirements.
2. When used within the context of a reference document under an agreement, the verbs shall, will, and should are only intended as informational and are not binding.

6.7.1 Requirement expression: format[ion] structure

A requirement must be in the form[ation] or structure of a complete "information package" (e.g., sentence).
A requirement must state a subject and predicate where the subject is a user.
The requirements must have, and state, an end result.

A requirement list/set must be consistent in its usage of the "to be" verb:

1. *Will* or *must* to show mandatory nature.
2. *Should* or *may* to show optionality.

Here are a few basic requirement sentence structures they can apply consistently. A very basic format is:

- Unique ID: Object + Provision/Imperative (shall) + Action + Condition + [optional] Declaration Of Purpose/Expected Occurrence (will)
  - For example, 3.1.5.3: The craft shall perform one complete fly-around (of the tower) at a range of less than 250 meters as measured from the craft center of mass to the tower center of mass; after undocking from the tower (and no declaration of purpose).

Table 14. Engineering Approach > Requirements:
Requirement types and their associated syntax patterns.

Requirements Type	Syntax Pattern
Ubiquitous	The <system name> shall <system response>
Event-Driven	When <trigger><optional pre-condition> the <system name> shall <system response>
Unwanted	If <unwanted condition or event> Then, the <system name> shall <system response>
State-Driven	While <system state>, the <system name> shall <system response>
Optional Feature	Where <feature is included>, the <system name> shall <system response>
Complex	<Multiple conditions>, the <system or unit name> shall <system or unit response>. (combinations of the above patterns)

An guiding objective of requirement defining is:

- Minimizing the amount of necessary requirements by eliminating overlapping requirements while ensuring the system is completely specified.

6.7.2 Requirements development

The process of requirements development requires all of the following phases and descriptions, occurring synchronously:

1. Define user:
   A. Who is interested in the system?
   B. How are decisions resolved?
   C. Who are the users and developers?
2. Define goals (objectives):
   A. Define broad (coarse) goals (non-specific goals)? What should be implemented or achieved?
   B. Broad goals divided into more specific goals (granular goals)? What should be implemented or achieved?
3. Define requirements:
   A. Goals (objectives) can be derived into concrete requirements that describe how the goals will be achieved and fulfilled.
   B. A requirement is:
      1. A specific statement of need derived from a goal.
      2. A specific statement(s) of reason (rationale) for the need including a relevant context.
      3. A specific explanation for how to achieve or fulfill (i.e., get) the requirement(s) in the context of a goal?
   C. Visualize and model the requirements in order to appropriately communicate and construct the how.

All requirements in a requirements list are composed of at least the following inputs:

1. **Requirement unique identifier:** Each requirement shall be assigned a project-unique identifier to support testing and traceability.
   A. Each requirement throughout the information system must be tagged with a project unique identifier (PUI). Tagging each requirement with a PUI optimizes traceability between high-level and low level requirements, and between requirements and verification tests. Each requirement should be marked with a PUI that allows users to easily reference both the requirement and its position in the overall document.
2. **The requirement statement:** Each requirement shall be stated in such a way that an objective test can be defined for it. An 'objective test' is a test for which the result can be commonly experienced.
3. **The requirement rationale (justification or reasoning)** - Each requirement shall include a rationale statement(s). When a requirement's rationale is visibly and clearly stated, its defects and shortcomings can be more easily spotted, and the rationale behind the requirement will not be forgotten. Rationale statements also reduce the risk of rework, as the reasoning behind the decision is fully documented and thus less likely to be re-rationalized

Requirement unique identifier:

For example: 3.5.2.5

- 3 = Transportation and Service requirements.
- 5 = Entry/landing requirements.
- 2 = Contingency.
- 5 = Space ventilation for emergency landing.

6.7.2.1 Requirement construction qualities

Requirements should posses (i.e., presence and not absence of) the following quality attributes:

1. **Complete** – precisely defines the system's responses to all real-world situations the system will encounter.
2. **Consistent** – does not contain conflicts between requirements statements.
3. **Correct** – accurately identifies the conditions of all situations the system will encounter and precisely defines the system's response to them.
4. **Modifiable (configurable)** – has a logical structuring with related concerns grouped together.
5. **Ranked (ordered)** – organizes the specification statements by importance and/or stability (which may conflict with the document's modifiability).
6. **Traceable** – identifies each requirement uniquely. A requirement must be traceable to some source. Each requirement should have a unique identifier allowing the software design, code, and test procedures to be precisely traced back to the requirement.
7. **Unambiguous** – states all requirements in such a manner that each can only be interpreted one way.
8. **Valid** – all project participants can understand, analyze, accept or approve it.
9. **Measurable** - functions can be assessed quantitatively or qualitatively.
10. **Verifiable** – must be consistent with related specifications at other (higher and lower) levels of abstraction.

Requirements must also be:

1. **Uniquely identifiable** - Each need is stated exactly once to avoid confusion or duplicative work. Uniquely identifying each requirement is essential if requirements are to be traceable and able to be tested.
2. **Performance specified** - Statements of real-world performance factors are associated with a requirement.
3. **Testable** - All requirements must be testable to demonstrate that the end product satisfies the requirements. To be testable, requirements must be specific, unambiguous, and quantitative whenever possible.

Simplistically, requirements must be:

1. **Conceived** - constructed.
2. **Bounded** - constrained.
3. **Coherent -** logically related, internally and externally.
4. **Acceptable** - sufficient input to resolve a design.
5. **Addressed** - scheduled, allocated, assigned.
6. **Fulfilled** - actualized.

Engineering is a real world creation process, and hence, requirements therein must possess the following characteristics (i.e., to be a "good" requirement):

1. Fulfill real world needs.
2. Have clear meaning.
3. Are organized coherently.
4. "Drive" engineering.

Requirements are prioritized:

1. **Terminal requirements** - A terminal requirement

is a statement in specific and measurable terms that describes what the system will be able to do, to be, or enable a user to do or be as a result of engaging with the system. A terminal requirement should be created for each of the tasks addressed within the system. Terminal requirement describe results, and not processes. After the terminal objective is created, it should be analyzed to determine if it needs one or more enabling/ supporting requirement. Each written requirement should include a task/performance, condition, and a standard:

   A. Task or Performance: States what the system will be doing.
   B. Condition: Specifies under what conditions the system should perform the task (defines the quality of performance of the system).

2. **Enabling/supporting requirement(s)** - are supporting or enabling requirements for terminal requirement. They are created by analyzing a terminal requirement. They allow the terminal requirement to be broken down into smaller, more workable and stabler requirements.

6.7.2.2 Requirement syntax

Requirements ("What") are a communications link between the source model and the implementation model ("How").
Herein, specificity and numeric measure are required for performance. Stating that a system should do something "quickly", is not a performance requirement, since it is ambiguous and cannot be verified. Stating that opening a file should take less than 3 seconds for 90% of the files and less than 10 seconds for every file is an appropriate requirement.

Instead of providing a unique section on performance requirements, include the relevant information for each feature in the statement of functionality.

Requirements are syntactically delineated:

1. When?/Under what conditions? (Phrase conditions)
   A. the system
   B. *shall / should / will* (Type of obligation from imperative)
   C. verb <process>; provide <whom> with the ability to; be able to <process>
   D. object
   E. additional details about the object.

6.7.2.3 Requirements tracing (traceability)

Tracing requirements means relating specific requirements to other project elements, especially to the following:

1. Backward tracing - a requirement to its source.
2. Traceability matrix - one requirement to another.
3. Forward tracing - a requirement to its design, code, documentation, or other forward project elements.

Simply, backward tracing ensures a source for each requirement. A traceability matrix ensures it is possible to evaluate the effect of changes to requirements among other inter-related requirements. Forward tracing ensures changes to requirements flow through to the design, code, project plan, etc. Forward tracing to the project plan provides data on how much work has been completed, and how much remains.

6.7.2.4 System requirements modeling

Requirements modeling is the process of constructing abstract, formal representation of the initial textually described system requirements in a way that is amenable to unambiguous interpretation, producing a requirements specification. This process ends with a requirements model (specification), which is expected to capture as much of the relevant real world semantics as possible. The core of the input system's requirements is a functional or behavioral, and non-functional, breakdown. This data based breakdown lists:

1. What the users need?
2. What the system must do to satisfy their needs?
3. What components must be built?
4. What each component must do, and how they will interact?

In the subsequent phases of the development process, the requirements model is elaborated and transformed into the design model (the [design] specification). This transition emphasizes the critical need for creating a formal, accurate, and complete requirement model from the outset, as it designed serve as the foundation of the entire development process and continued service life cycle.

Modeling is targeted at clear and accurate representation of the concepts that comprise the system. An important benefit of requirements modeling is that since the resulting model is available at an early stage in the system's life cycle, model analysis and simulation may be used to validate the requirements and reduce conceptual design errors. Later on, the requirements modeling is integrated into the life cycle flow of activities in the development process.

A good requirements specification is one in which requirements are arranged hierarchically. Few high-level, broadly defined requirements are specified in increasing levels of detail, where each level contains a set of requirements that elaborate upon one or more requirements at the level above it. A hierarchical structure of requirements may facilitate the process of modeling. In general, high-level requirements correspond to abstract processes, aggregate objects or agents (actors), and interactions between them at lower levels.

Each requirement is a specification relating to some characteristic of a system. Model components are introduced and associated with a corresponding requirement (or requirements set) with which the model component is related, creating the objective condition of traceability.

In community, there is no necessary conceptual gap between engineering objectives and human objectives; they seamlessly become one and the same. In the market, however, there is a gap between customers, employers, and employees, and also between client's, engineering, and business. In other words, in the market, a conceptual gap, which is often very wide, exists between these two [problem-requirement] model types, since one faces the client with a problem domain, while the other faces the solution domain provided by a semi-independent business entity whose role is, a technological solution provider. The result of the gap entails a host of consequences, including the necessity for subjective decisioning, and therein, the introduction of various subjective biases that carry on over time and become systemic [to the societal system].

> **APHORISM:** *Share information about fishing to a human and s/he can fish for a lifetime.*

During the continuous modeling process, issue tracking attributes track the source and status of a requirement. This is basic issue tracking, allowing for requirements traceability. The attributes are:

1. Record the source.
2. Record the urgency (urgency spectrum).
3. Record the sufficiency of data to resolve (decisioning).
4. Identify verification method (test, demonstration, inspection, simulation, analysis).
5. Identify constraints (safety, performance, reliability, contracts, standards, rules).
6. Record integrations (specifications).

6.7.2.5 Requirements gap analysis (project coordination task)

If requirements are not available, or not yet well understood, then an gap/requirements analysis (and possible discovery) must be complete to determine what is missing (define the gap or design space between what is present and what is expected). The purpose of Requirements Analysis is to discover unknown requirements (i.e., to turn unknown requirements into known requirements).

6.7.3 Requirement sub-types

Requirements can be classified by the presence of a function into functional requirements and non-functional requirements. There are two primary requirement subtypes divided by function. Herein, qualitative is the conceptual encoding of function, whereas functional is the physical encoding of function:

1. **Non-functional requirements (Qualitative requirements)** that become encoded into the "behavior" of a function, or "status" of a state) - conditions that must be met that are not explicit capabilities.
2. **Functional requirements** (specify an exact function) - capability that the system must perform.

Functional requirements/metrics are capabilities (as physicalizable states or processes) that the product or service (as a system) must perform. Functional requirements meet functional user needs. These are the most fundamental of physicalizable requirements. In the market, fundamental functional requirements are generally referred to as "business" requirements, because they are what the "business" needs to survive. In community, fundamental functional requirements are sometimes referred to as human requirements or human needs, because they are what individual humans in common need to survive and thrive. In society, these functional human requirements are built upon a set of human needs and objectives.

> *Note: What one person senses, another may sense differently, thus the need for clear communication and preferentially, electric instrumentation where possible. At the level of a project, there is a need for requirements to be referenceable (i.e., traceable) to their "tested" results, which may be verified or not.*

Non-functional requirements/metrics can be visualized as the encoding of conception into a real world reality by "shaping" its iterative expression; like batter being pushed through a cookie cutter shape to form individual iterations of that cookie. After application, non-functional requirements become operational (i.e., concepts in operation). The term for a concept in operation usually ends in -ability: usability, dependability/reliability/durability, mobility, scalability, sustainability. For instance, a system can be designed to be created and operated sustainably. When in operation, the system may continue, or not, to remain sustainable, through its continued design and operation.

In general, non-functional requirements are sourced from a value system. A value system is effectively a set of non-functional requirements. The value system forms objects, which then forms the non-functional requirements. Non-functional requirements per definition do not describe what functionalities the platform will deliver, but how they will be delivered. There are two "hows" here:

1. How will the system be produced (i.e., under what quality conditions).
2. How will the system operate (i.e., under what quality conditions).

Values are the translation of concepts into operation; they are a higher level abstraction than concepts in operation. Values become operational concepts (Concepts in Operation), and the real, existent and functioning systems they create are described through a Concept of Operation (a system's high-level conception, abbreviated ConOps, CONOPS, CONOPs, or CONOps). A 'Concept of Operation' document/ dashboard describes the characteristics of a proposed system from the viewpoint of an individual who will use the system. It socially communicates the quantitative and qualitative system characteristics of a potentially, or actually existent, system. The concepts used in that document, translate through active human "experience" into physical interactions that lead to the physical construction of a physically operational system. That system may operate well, in concern to its requirements by the user, or it may not.

6.7.3.6 System-level Requirement categorizations

Requirements about a system to be developed are sometimes categorized by their source (point of origin) and/or system's component. Take note that there is, however, only one unified set of structured requirements for any systems engineering project.

The primary requirement types are:

1. **Functional requirements (Do):** Requirements that define what the system must do in essence, or, in other words, what it accepts and what it delivers (i.e., expected transformation). For example: The system shall accept coins; 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. What values and qualities will the system express. Examples include: 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 into what it delivers. Examples include: The system shall consume less than 100W; The system shall have a mass of less than 10kg.
4. **Interaction requirements (Interact):** Requirements that define where the system must operate, which includes any type of operation during its life-cycle. Examples include: The system shall withstand shock levels higher than 100g; The system shall operate in vacuum (to reflect operation); The system shall operate in clean room class 10,000 (to reflect Assembly, Integration, and Test activities).

The following are common requirement categorizations:

1. **Project requirements** are statements identifying what is required to complete a project, which starts with coordination requirements and project user requirements.
2. **User requirements** are written from the point of view of end users, and are generally expressed in narrative form, "The user must be able to change the color scheme of the welcome screen."
   A. In contrast to the roles of user (customer) and developer (employee) in the market, in community the roles of "user" and "developer" are the equivalent, meaning that there is no structural separation of requirements. Remember that in business, "users" are customers, and "developers" are employees. In community, there is no business, and hence, no financial separation between users and developers. In community, users are also developers as part of an InterSystem Team structure.
3. **System requirements** are statements describing the functions the system needs to do, and the non-functional states the system needs to be. System requirements are usually more technical in nature, "The system will include four pre-set color schemes for the welcome screen. Colors must be specified for the page background, the text, visited links, unvisited links, active links, and buttons (base, highlight, and shadow)." System requirements may have to do with how the system is built or functions.
   A. What the system will do to meet those needs.
   B. What do we need to know to build this?
4. **Engineering requirements** are statements including numbers and operational concepts that describe the functional dynamics and non-functional states of a proposed system.
   A. **Development requirements** - what is required to do development.
   B. **Construction requirements** - what is required to construct the system.
   C. **Operational requirements** - what is required to operate the system.
5. **Interface requirements** specify how the interface (the part of the system that users see and interact with) will look and behave. Interface requirements are often expressed as screen mock-ups; narratives or lists are also used. A description of the information (protocol and physical) interface between components of a system.

6. **Component requirements** specify a descriptive list of all things that each component must do and/or be.
7. **Material requirements** specify the type, quality, and quantity of materials required.
8. **Negative requirements** refers to the create boundaries to what the system should do. However, it is not always possible to measure what a system should not do; because, how can "you" test something that should not happen.
9. **Constraining requirements** (a.k.a., non-functional requirements or objectives) - requirements that constrain implementation and operation of a system.
10. **Project plan coordination requirements** refers to those requirements specifying what the project coordination system should do (and be) to coordinate information and resources. These requirements are for the continuous process of project coordination, and not the system to be engineered, which has its own set of requirements. Technically, they are both engineering requirements.

6.7.3.7 Project-level requirement categories

At the project level, there are several primary categories of requirement:

1. **Information requirements**
   A. Research requirements
   B. Design/production requirements (to produce the deliverable of a societal system specification)
   C. Engineering/Operation requirements
   D. Project plan/coordination activities
2. **Material requirements**
   A. Material resources for coordination,
   B. Material resources for design.
   C. Material resources for construction.
   D. Material resources for operation.
   E. Material resource for cycling.
3. **Human requirements**
   A. Human presence for knowledge.
   B. Human presence for effort and capabilities.
4. **Operational Requirements**
   A. The operational requirements should answer:
      1. *Who* is asking for this requirement? Who needs the requirements? Who will be operating the system?
      2. *What* functions/capabilities must the system perform? What decisions will be made with the system? What data/information is needed by the system? What are the performance needs that must be met? What are the constraints?
      3. *Where* will the system be used?
      4. *When* will the system be required to perform its intended function and for how long?
      5. *How* will the system accomplish its objective? How will the requirements be verified?
5. **Market-State requirements (Financial and contractual requirements)**
   A. In the market-State where resources are not held as the common heritage of all of humanity, resources carry transactional costs (e.g., trade goods, bartering service, and currency).
      1. Financial requirements (a.k.a., currency costs)
         i. Contractual requirements (which are really financial requirements)
         ii. Financially feasible conditions for creation and operation of society.
         iii. Multiple types of resource costs: hardware, software, land, manufacturing, logistics and assembly, State and legal (jurisdictional), and expertise.
      2. Contractual requirements (i.e., where force is above financial requirements)
   B. Frequency of cost:
      1. One-time (accounting for repair or replacement)
      2. Marginal (no additional after setup)
      3. Reoccurring (cyclical)

6.7.3.8 Societal-level requirement input types

At the societal level, there are several primary categories of requirement:

1. **Human [end-]user requirements:** Human needs, wants, and preferences - describe generally the needs, goals, and tasks of the user (this is the end user; there is no market-based project requester). All measurements of quality, success, and optimization relate to the user, who is the individual human being in Community. User requirements specifically refer to user fulfillment.
   A. What do "we" need, want, and prefer as a human individuals interconnected within a global societal structure? What is required to work together, to integrate, share information openly, to perceive and act upon "our" inter-connectedness.
2. **[Societal] System requirements:** Community-type society instantiation requirements - a description of the societal system itself and what the system must do. What informational (through to material) systems require to sustain the current instantiation (iteration) of the societal system? These are requirements that describe the capabilities of the system with which, through

which, and on which humans maintain their society (i.e., function together). Note: Technically, everything is information, from conception through into materialization; hence, the habitat service (material) system is a sub-system of the information system. Here, high-level functions and logic are defined.

A. **Information System instantiation requirements:** These are requirements with which, through which, and on which humans maintain their information system's instantiation.
   1. Social; decision; lifestyle; material (Subsystem-level functions and logic are defined here).

B. **Habitat Service (Material) System instantiation requirements:** These are requirements with which, through which, and on which humans maintain their material habitat service system's instantiation.
   1. Ecological services, Life Support Service, Technical Support Service, and Facility Support Service.

6.8 Requirements standards

The principal standard defining a requirement is:

- ISO/IEC/IEEE 29148: System and software engineering - Life Cycle Processes - Requirements Engineering

The five key deliverables of the ISO/IEC/IEEE 29148 standard are:

1. Stakeholder requirements specification (StRS; user requirements specification) document.
2. System requirements specification (SyRS) document.
3. Software requirements specification (SRS) document.
4. System concepts documents.
   A. System operational concept (OpsCon) document.
   B. Concept of operations (ConOps) document.

6.9 Requirements engineering

A.k.a., Requirements engineering process, requirements definition stage.

Requirements engineering is the iterative process of establishing the services that the user requires from the solution system and the constraints under which it is to be developed (e.g., development conditions) and under which it operates (e.g., service conditions, value conditions). The processes used for requirements engineering vary widely depending on the application domain, the project type, and the organization developing the requirements. In practice, requirements engineering is an iterative activity in which requirement tasks/activities/processes are iterated and inter-related.

The collection and analysis of information known as requirements engineering happens continuously throughout the project's life cycle. Therein, requirements require the following actions:

1. Requirements analysis involving identification, rationale, positioning and prioritization.
2. Show where work is required to resolve/complete requirement.

Requirements engineering involves, but is not necessarily limited to, the [iterating/spiralling] requirements process tasks of (a.k.a., generic requirements engineering activities, requirements engineering stages):

1. **Requirements coordination** (a.k.a., requirements management) - all coordination tasks/processes/stages associated with the information set, 'requirements'. Of significant importance here are the processes of tracing/tracking and changing (change controlling) requirements.
2. **Requirements discovery** (a.k.a., requirements collection, requirements elicitation, requirements solicitation, requirements identification, gathering requirements) - the process of identifying all requirements. Discovery may involve interviews, evaluations, observation and study, scenarios, use cases, work/model flow diagrams, sequence diagrams, activity diagrams, event diagrams, decision trees, etc. Identify all requirement sources.
3. **Requirements analysis** - the process (technique) of understanding user needs (requirements) and translating (transferring) them into a set of requirements for system construction and/or modification.
   A. **Requirements classification and organization** - grouping related requirements and organizing them into coherent clusters.
   B. **Requirements prioritization** - Prioritizing requirements and resolving requirements conflicts.
   C. **Technical requirements validation** - the process of checking the requirements for their expected attributes, including: validity; consistency, completeness, realism, verifiability, etc. Are there technical errors; conflicts; ambiguities; and does the requirement (and requirements specification) conform to standards?
4. **Requirements specification** - the collection of requirements necessary to complete the project

into a formal document/database.
5. **Requirements verification** - technical verification that the system operates as required. Proving [objectively] that each requirement is satisfied. Can be done by logical argument, inspection, modeling, simulation, analysis, test, or demonstration.
6. **User requirements validation** - user validation that the system can be (or, is being) used as expected.

Note that the above is sometimes more simply depicted as a [repeating] four/five stage cycle:

1. Requirements discovery.
2. Requirements classification and organization (grouping).
3. Requirements prioritization.
4. Requirements specification (repeats here).
5. Requirements verification (or, repeats here).

The simplified requirements engineering process:

1. User requirements definition:
   A. Inputs:
      1. Source documents.
      2. User needs.
      3. Project constraints.
   B. Activities:
      1. Articulate demands.
      2. Define user requirements.
      3. Analyze and maintain user needs (priority demands).
   C. Outputs:
      1. Concept documents.
      2. User requirements.
      3. Measures of effectiveness needs.
      4. Measures of effectiveness data.
      5. Validation criteria.
      6. Traceability.

6.9.1 Requirements coordination planning

Requirements coordination planning decisions include, but are not limited to:

1. **Requirements identification** - Each requirement must be uniquely identifies so that it can be cross-referenced with other requirements.
2. **A change control process** - This is the set of activities that assess the impact of changes to requirements.
3. **Traceability structures** - Information structures that define the relationships between each requirement and between the requirements and the system design.
4. **Requirements tool support** - Tools that support coordination and planning, such as spreadsheets, databases, and content coordination (content management) systems.

6.9.2 Requirements definition

Requirements definition is a stage in the project coordination and systems engineering life-cycle. The primary goal of this stage is to develop a basis of mutual understanding between the users and the development team about the requirements for the project. The result of this understanding is an selected (approved) 'requirements specification' that becomes the initial baseline for product design and a reference for determining whether the completed product performs as the system user requested and expected. All system requirements, (e.g., software, hardware, performance, functional, infrastructure, etc.) should be included.

This stage involves analysis of the users' processes and needs, translation of those processes and needs into formal requirements, and planning the testing activities to validate the performance of the product.

6.9.2.1 Define system requirements

Use the project scope, objectives, and high-level requirements as the basis for defining the system requirements. The questions used to define the objectives may be helpful in developing the system requirements. The goals for defining system requirements are to identify what functions are to be performed on what data, to produce what results, at what location, and for whom. The requirements must focus on the products that are needed and the functions that are to be performed. Avoid incorporating design issues and specifications in the requirements. One of the most difficult tasks is to determine the difference between "what" is required and "how to" accomplish what is required. Generally, a requirement specifies an externally visible function or attribute of a system (i.e., "what"). A design describes a particular instance of how that visible function or attribute can be achieved (i.e., "how to").

When requirements are being defined, it is not sufficient to state only the requirements for the problems that will be solved; instead, all of the requirements for the project must be collected.

> **NOTE:** *It is often difficult for a non-specialist to understand technically written requirements and their implications.*

6.9.2.2 Writing requirements

Requirements are written in several different notational forms, including:

1. **Natural language** - The requirements are written using numbered sentences in natural language text. Each sentence should express one requirements. Natural language sometimes carries

the problem of a lack of clarity (i.e, precision may be difficult without making the document difficult to read as multiple conditional statements may be requirement and multiple requirements and types of requirements may be expressed together).

2. **Structural natural language** - The requirements are written in natural language text on a standard form or template. Each field provides information about an aspect of the requirement.
3. **Design description language** - This approach uses a language like programming language, but with more abstract features to specify the requirements by defining an operational model of the system.
4. **Mathematical specification (a.k.a., formal specification)** - These notations are based on mathematical concepts, such as finite-state machines or sets.
5. **Tabular notation (a.k.a., table notation)** - The requirements are written in one of the prior four notational forms and placed into a spreadsheet-like table. Generally, tabular notation is used to complement natural language. Tabular notation is especially useful when a number of possible alternative courses of action must be defined. Each row in the table represents a requirement.
6. **Graphical notation** - Graphical models, supplemented by text annotations, are used to define functional requirements for the system; UML use case and sequence diagrams are commonly used.

6.9.3 Requirements specification

A.k.a., Requirements specification document, system requirements specification (SRS).

The requirements for the project are formally documented in the 'requirements specification'. This is the formal (official) statement of what is required of the system developers. All system requirements should be included, however, a definition of user requirements may or may not be included in the document itself. This is the set of selected (agreed) statements on the system requirements. It should be organized so system users and system developers can use it. Simply a requirements specification is a complete description of the behavior of the system and the conditions under which it must be developed and operated. A requirements specification document is "living" during development, and is a reference document for development and operations; it must be maintained over the life of the project. It is the basis (baseline) for the selection of, and agreement on, the system. It also provides a basis (baseline for validation and verification).

The requirements specification becomes the initial baseline (formal reference document) for product design and a reference for determining whether the completed product performs as the system user requested and expected.

The requirements specification should define/ establish the environment in which the system to be developed will operate.

Each requirement in the requirements specification should be uniquely identified in a 'requirements traceability matrix'. Each requirements should include an explanation (rationale) of why the requirement is necessary. A requirements specification represents the compilation and documentation of all requirements.

A 'requirements specification' is not technically a design [specification] document. As far as possible, it should define (formally set) what the system should do, rather than how the system should do it. In principle, requirements should state what the system should do, and the design should describe how it does this. However, in practice, requirements and design are inseparable due to the following:

1. A system architecture (system structure) may be designed to structure the requirements.
2. The system may inter-operate with other systems that generate design requirements.
3. The use of a specific architecture to satisfy non-functional requirements may be an external [domain] requirement.
4. Requirements may be a consequence of a [societal] standards requirement.

The following factors should be considered when generating a requirements specification:

1. Select and use a standard format for describing the requirements. Ensure compliance with standards.
2. Present the logical and physical requirements without dictating a physical design or technical solutions.
3. Write the requirements in non-technical language that can be fully understood by the system users.
4. Write the requirements in technical language that can be fully understood by the system developers.
5. Organize the requirements into meaningful groupings.
6. Develop a numbering scheme for the unique identification of each requirement.
7. Select a method for:
   A. Tracing the requirements back to the sources of information used in deriving the requirements (e.g., specific system user project objectives).
   B. Threading requirements through all subsequent life-cycle activities (e.g., testing).

The following factors are generally not included in a requirements specification:

1. Project requirements, such as, delivery schedule,

staffing, reporting procedures, cost. These are included in the Project Plan. If, however, the requirements specification is part of the project plan, then there is nuance here.
2. Design solutions.
3. System assurance plans: quality assurance plans, configuration procedures, verification and validation procedures, etc.

Users of the requirements specification include:

1. **System users** - specify the requirements and read them to check that they meet their needs. Users specify changes to the requirements.
2. **Project coordinators** - Use the requirements document to plan coordination for the system development process.
3. **System engineers** - Use the requirements to understand what is to be developed.
   A. **System test engineers** - Use the requirements to develop validation/verification tests for the system.
   B. **System maintenance engineers** - Use the requirements to understand the system and the relationship between its parts.
4. **Decision system** - Use the requirements to determine risks and societal-level project effectiveness (Read: the decision system effectiveness inquiry).

6.9.3.1 Requirements traceability

A.k.a., Requirements cross-referencing.

Requirements traceability refers to the ability to describe and follow the life of a requirement, in both forwards and backwards direction - from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any life-cycle phase. To ensure traceability, the "life" of a requirement must be documented in a requirements traceability matrix, which allows anyone to find the origin of each requirement and track every change which was made to this requirement.

Using databases allows for easy traceability. For any organization there should exist a requirements database for all possible requirements.

NOTE: *Tracing can be difficult when using multiple tools.*

6.9.3.2 Requirements traceability matrix

The 'requirements traceability matrix' is a requirements coordination tool used to trace project life-cycle activities and work products to the project requirements, and it ensures requirements are traced and verified through the various life-cycle stages, especially during design, testing, and implementation stages. The matrix establishes a thread that traces requirements from identification through implementation. Requirements within the matrix must be traceable from external sources (such as, the user), to derived system-level requirements, to specific hardware and/or software product requirements. In other words, the requirements traceability matrix is a matrix that traces the requirements forward and backward; it traces project requirements back to the project objectives identified in a project charter, for example, and forward through the remainder of the project life-cycle stages.

The requirements traceability matrix is a threading matrix that groups requirements by project objectives. The requirements traceability matrix contains descriptions for each item in the matrix. Under each project objective, the source of the requirement, the unique requirement identification number, and the life-cycle activities are listed in columns along the top and the project requirements in rows along the left side. As the project progresses through the life-cycle stages, a reference to each requirement is entered in the cell corresponding to the appropriate life-cycle activity. The matrix should be capable of being expanded at each stage to show traceability of deliverables (work products) to the requirements and vice versa.

Every project requirement must be traceable back to a specific project objective(s) described in the project's formal direction document (e.g., project charter). This traceability assures that the system (product) will meet all of the project objectives and will not include inappropriate or extraneous functionality or conditions. All deliverables (work products) developed during the design, production, coding, and testing processes in subsequent life-cycle stages must be traced back to the project requirements described in the 'requirements specification'. This traceability assures that the product will satisfy all of the requirements and remain within the project scope.

It is important to know (document, log) the source of each requirement, so that the requirements can be verified as necessary, accurate, and complete.

A copy of the requirements traceability matrix should be placed in the Project File.

6.9.3.3 Requirements diagram

Requirements diagrams show how the different requirements are linked to the block value properties and the hierarchy of requirements that can be used downstream analysis

6.10 Requirements coordination

A.k.a., Requirements management.

Requirements coordination is a process composed of the core processes of tracing and changing requirements, in conjunction with the processes of gathering, organizing, prioritizing, and documenting requirements. Requirements coordination allows for the

verification that all requirements have been collected for the system (Read: the product), and that requirements are traceable and changes are controlled effectively and efficiently. Requirements coordination documents the needs, expectations, and understanding of the product to be delivered and provides a framework for identifying, planning, scheduling, verifying, tracing, testing, evaluating, and changing requirements to fulfill user needs (and expectations) of the project.

The processes of gathering, organizing, prioritizing and documenting requirements are based on an interactive communication process that relies on a working relationship between users, the system's developers (the project team), and possibly, the system's operators, to discover, define, refine, and record a precise representation of the system's requirements.

As the project progresses, more requirements may be identified and coordinated through a change control process. As part of requirements coordination, the project coordinator must track requirements that are accepted for the current project and those that will be planned for subsequent releases.

6.10.1 Requirements identification system

The creation of a standard identification system for all requirements is required in order to facilitate control, traceability, and testing activities. The identification system must provide a unique designator for each requirement. For example, the identification system can classify the requirements by type (e.g., functional, input, or computer security). Within each type classification, the requirements can be assigned a sequential number. Select an identification system that is appropriate for the scope of the project.

6.10.2 Requirements change system

As a project evolves, the requirements may change or expand to reflect modifications in the users' plans, design considerations and constraints, advances in technology, and increased insight into user processes. A formal change control process must be used to identify, control, track, and report proposed and selected ("approved") changes. Selected changes in the requirements must be incorporated into the 'requirements specification' in such a way as to provide an accurate and complete audit trail of the changes.

6.11 Requirement definition tasks

The following are common tasks involved in defining system requirements:

1. Define functional requirements.
2. Define non-function/performance requirements.
3. Define input and output requirements.
4. Define user interface requirements.
5. Define system interface requirements.
6. Define communication requirements.
7. Define access requirements.
8. Define backup and recovery requirements.
9. Define preliminary implementation requirements.
10. Develop system test requirements.
11. Develop acceptance test requirements (validation requirements).

In other words, are multiple sub-types of requirements, including but not limited to:

1. **User requirements (a.k.a., user requirements definition)** - Statements in natural language, diagrams, tables, and other notations of the services, or system, and its operational constraints, which are understandable to the user. Written for users (i.e., understandable by end-users who do not have a technical background. What the system should do for the user.
2. **System requirements (a.k.a., product requirements definition)** - Statements in technical language, possibly including diagrams, tables, and other notations that represent a completely detailed description of the system's functions, services, and operational constraints. Written for developers (designers and constructors).
   A. **Data requirements** - Identification of the data elements and logical data groupings that will be stored and processed by the system.
   B. **Process requirements** - Identification of a specified method or language.
   C. **Transitional requirements** - Requirements necessary to transition to a new system.
   D. **Operational requirements** - Systems requirements that specify how the system must operate.
   E. **Maintenance requirements** - System requirements that specify how a system must be maintained (e.g., replacement of parts).
   F. **Sustainment requirements (a.k.a., maintainability requirements)** - Specify how a system must be sustained (e.g., supplied with fuel).
   G. **Retirement requirements** - Specify how a system must be retired from service (e.g., disposal of hazardous materials).
3. **External system requirements (a.k.a., environmental requirements, domain requirements)** - requirements that arise from factors that are external to the system and its development process (e.g., interoperability requirements, legislative requirements, compliance requirements, etc.). External/domain requirements can be new functional requirements, constraints (non-functional requirements) on existing

requirements, or define specific computations. If domain requirements are not satisfied, the system may be unworkable or unsafe. For instance, a train control system has to take into account braking characteristics in different weather conditions.

A. An example: The system shall implement standardization requirements as set out in document #.

6.11.2.1 Identify functional requirements

A.k.a., Define functional requirements.

Identify requirements for all functions, regardless of whether they are to be automated or manual. Describe the automated and manual inputs, processing, outputs, and conditions for all functions. Include a description of the standard data tables and data or records that will be shared with other objects or applications. Identify the forms, reports, source documents, and inputs/outputs that the system will process or produce to help define the functional requirements.

Develop a functional model to depict each process that needs to be included. The goal of the functional model is to represent a complete top-down picture of the system (product). Use flow diagrams to provide a hierarchical and sequential view of the system user's functions and the flow of information through the system.

6.11.2.2 Identify non-functional requirements

A.k.a., Define non-functional requirements, define performance requirements.

Identify requirements for all conditions and constraints the system (and its development) must satisfy.

6.11.2.3 Define input and output requirements

A.k.a., Define information requirements.

Describe all manual and automated input requirements for the system (e.g., data entry from source documents and data extracts from other applications). Document where the inputs are obtained. Describe all output requirements for the system. Document who or what is to receive the output.

6.11.2.4 Define user interface requirements

The user interface requirements should describe how the user will access and interact with the system, and how information will flow between the user and the system.

CLARIFICATION: *'Interfaces' are boundaries that are between elements of a system.*

A standard set of user interface requirements may be established for the system owner organization. If not, work with the system users to develop a set of user interface requirements. A standard set of user interface requirements will simplify the design and development processes, and ensure that all systems have a similar look and feel to the users. When other constraints (such as a required interface with another application) do not permit the use of existing user interface standards, an attempt should be made to keep the user interface requirements as close as possible to the existing standard.

Define the user interface requirements by identifying and understanding what is most important to the user, not what is most convenient for the project team.

The following are some of the issues that should be considered when trying to identify user interface requirements:

1. The users' requirements for visual and behavior elements, navigation, and help information.
2. The standards issued by the decision system and societal-level organizations that apply to user interfaces.
3. The classification of the users who will access and use the system.
4. The range of functions that the users will be performing with the product.

6.11.2.5 Define system interface requirements

The hardware and software interface requirements must specify hardware and software interfaces required to support the development, operation, and maintenance of the system.

The following information should be considered when defining the hardware and software interface requirements:

1. Users' environment.
2. Existing or planned system that will provide data to or accept data from the new system.
3. Other organizations or users having or needing access to the system.
4. Purpose or mission of interfacing.
5. Common users, data elements, reports, and sources for forms/events/outputs.
6. Timing considerations that will influence sharing of data, direction of data exchange, and security constraints.
7. Development constraints such as the operating system, database system, language, compiler, tools, utilities, and protocol drivers.
8. Standardized system architecture defined by hardware and software configurations for the affected organizations, sites, or operations.

6.11.2.6 Define communications requirements

The communication requirements define connectivity and access requirements within and between user

locations and between other groups and applications.

The following factors should be considered when defining communication requirements:

1. Communication needs of the user and InterSystem Team organizations.
2. User organization's existing and planned communications environment (e.g., telecommunications; LANs, WANs, wired, wireless etc.).
3. Projected changes to the current communication architecture, such as the connection of additional local and remote sites.
4. Limitations placed on communications by existing hardware and software including:
   A. User systems.
   B. Applications that will interface with the product.
   C. Organizations that will interface with the product.
5. Standards that define communication requirements and limitations.
6. Future changes that may occur during the project.

6.11.2.7 Define access requirements

A.k.a., Define access control requirements, define security control requirements.

Develop the data and system access requirements in conjunction with the system users. This involvement affords early determination of access, and levels of access protection required for the system.

Use the following procedure to determine access requirements:

1. Identify the types of data that will be processed by the system.
2. Determine preliminary data integrity (protection, security) requirements.
3. Coordinate with the users and InterSystem Team operators of the platform to identify existing supporting computer access (security) controls, if applicable.
4. Incorporate access requirements into the 'requirements specification'.

The following list provides sample questions that can be used to help define the access controls for the system:

1. What access controls (access restrictions) are placed on the users by the societal organization?
2. What are the audit and other checking needs for the system?
3. What separation of accountabilities, control related functions, operating environment requirements, or other functions will impact the system?
4. What measures will be used to monitor and maintain the integrity of the system and the data from the user's viewpoint?

6.11.2.8 Define preliminary implementation requirements

Describe the requirements anticipated for implementing the system (e.g., user production cycle). The high-level implementation requirements are identified early in the life-cycle to support decisions that need to be made for the information systems engineering approach. The implementation requirements are expanded into a full implementation approach during the design stages.

The following factors should be considered when defining preliminary implementation requirements:

1. **Operating environment** - identify any capacity restrictions given by the environment, existing hardware and software that need to be identified and addressed.
2. **Acquisition** - If hardware or software must be acquired, identify the necessary acquisition activities. These activities include preparing specifications, estimating costs, scheduling procurement activities, selection, installation, and testing.
3. **Conversion** - Identify requirements for converting data (or systems) from an existing or external application to the new product.
4. **Installation** - Identify the installation requirements.
5. **Training** - Identify the specific training needs for various categories of users and InterSystem teams.
6. **Documentation** - Identify requirements for the development and distribution of operational documentation for support personnel and user documentation. Operational documentation may include task control procedures and listings, operational instructions, system administration responsibilities, archiving procedures, and error recovery. User documentation includes the user manual, step-by-step instructions, online documentation, and online help facilities.

6.11.1 Functional requirements

A.k.a., Functional user requirements, functional system requirements.

Functional requirements describe functionality or system services, or are descriptions of how some computations must be carried out. Functional requirements are statements of what the system should do in detail. Functional requirements define what the system must do to support the system users functions and objectives. A functional requirements specification represents a model of the desired behavior of the system.

Functional requirements are statements of:

1. Services (functions) the system should provide.
2. How the system should react to particular inputs.
3. How the system should behave in particular situations.
4. May state what the system should not do.

The functional requirements should answer the following questions:

1. How are inputs transformed into outputs?
2. Who initiates and receives specific information?
3. What information must be available for each function to be performed?

Examples of functional requirements include:

1. A user shall be able to search the unified information system for all resources.
2. The system shall generate each day, for each city, a list of actively used services.
3. Each user using the system shall be uniquely identified by a # of digits user/community number.

6.11.2 Non-functional requirements

A.k.a., Non-functional user requirements, non-functional system requirements, constraints, quality requirements.

Non-functional requirements are constraints on the services or functions the system provides and the development process being used. Common non-functional requirements include timing constraints, development process constraints, operating constraints, standards, etc. Non-functional requirements may also define and constrain system properties, such as reliability, response time, storage requirements, etc. Non-functional requirements may be more critical than functional requirements, for if these are not met, the system may be useless.

Non-functional requirements may affect the overall structure of a system, rather than the individual components. For example, to ensure that performance requirements are met, a developer may have to organize the system to minimize power flow between components.

Additionally, a single non-functional requirement, such as a reliability requirement, may generate a number of related functional requirements that define system services (functions) that are requirement. It may also generate requirements that restrict existing requirements.

Often, though not always, non-functional requirements apply to the system as a whole, rather than individual features or services.

Non-functional classification types include, but are not limited to:

1. **Non-functional system requirements** - requirements that specify that the delivered product must behave in a particular way (e.g., execution speed, reliability, etc.).
   A. For example: The system shall be available to all users during the hours of (Mon-Fri, 08:30-19:00).
2. **Organizational system requirements** - requirements that are a consequence of organizational value standards/conditions and procedures (e.g., process standards, implementation requirements, value conditions, etc.).
   A. For example: Users of the system shall authenticate themselves using their biometric identity.

6.11.2.1 Performance requirements

Performance requirements define how the product must function (e.g., hours of operation, response times, and throughput under various load conditions). The information gathered in defining the project objectives can translate into very specific performance requirements; (e.g., if work performed for an organization is critical to the society, the hours of operation and throughput will be critical to meeting the mission). Also, standards can dictate specific availability and response times.

6.12 Requirements analysis

A.k.a., Requirements analysis technique.

A requirements analysis [technique] is the set of data collection and analysis techniques combined with the life-cycle requirements standards (e.g., tracing the requirements through all life-cycle activities) that are used to identify the project requirements and to define exactly what the system (product) must do to meet the system users' needs and expectations. When appropriate, the technique must include methods for collecting data about users at more than one geographic location and with different levels and types of needs.

The requirements analysis technique should be in harmony with the type, size, and scope of the project; the number, location, and technical expertise of the users; and the anticipated level of involvement of the users in the data collection and analysis processes. The technique should ensure that the functionality, performance expectations, and constraints of the project are accurately identified from the system users' perspective. The technique should facilitate the analysis of requirements for their potential impact on existing operations and business practices, future maintenance activities, and the ability to support the system user's long-range information resource coordination plans. It is advantageous to select a technique that can be repeated for similar projects. This allows the project team and the system users to become familiar and comfortable with

the technique.

7 [Engineering] Requirements for habitability

A.k.a., Habitat supportability requirements.

Humanity is a global species, and so, it must necessarily recognize, and maintain, a regenerative global habitat. There are multiple layers of accountable requirements in the development of a habitat, and they include, but are not limited to:

1. Social requirements (togetherness).
2. Individual requirements (human).
3. Service requirements (access).
4. Project requirements (organized doing).
5. Technical requirements (what).
6. Team requirements (who).
7. Role requirements (execute).

Habitats are sustained by accountable humans and machines fulfilling roles as part a coordinated network of teams that complete technical requirements on the part of projects that exist to service individual and social human needs. That habitat is designed to optimally support the humans given what the humans know and the available environment.

7.1 Support (habitat supportability)

Habitat supportability is the ability of the habitat to optimally meet the [human] requirements of the inhabitants. The cornerstone of the habitat supportability concept is that each habitat service system (city) functions significantly (though not fully) independent of physical resource support from the other habitats. The crews of these missions must have all of the resources and capabilities that will be necessary to enable them to succeed fully and complete the mission without direct intervention from Earth-based supporting personnel.

This self-reliance will be achieved, in part, by increased emphasis on maintenance by repair rather than replacement. A repair-centered maintenance approach would only be effective, however, when it is strategically coupled with a hardware design that is specifically structured as part of the supportability concept.

The habitat service system is sustained by people working together with their environment. The habitat service system represents a relationship between people, machines (soft and hard), and a living ecosystem.

7.2 Maintenance (habitat maintainability)

Robust, autonomous maintenance capabilities are likely to be enabled by implementation of the following concepts and capabilities:

1. Repair rather than replace. It is preferred to repair failed hardware items rather than simply remove

and replace them. This concept is particularly important for LRUs, ORUs and shop replaceable units (SRUs) that have high failure rates and large masses or volumes. This avoids the use of large quantities of relatively bulky and massive spares.

2. Replace at the lowest practical hardware level. The objective is to minimize the mass of spares consumed. An example would be to remove and replace an integrated circuit that has a mass of grams rather than a complete avionics LRU that has a mass of several kilograms.
3. Comprehensive on-board failure diagnosis. Failure diagnosis should identify the cause of the failure to the level of maintenance. To the extent possible, these capabilities should be built into the systems. When this is not practical, standalone diagnostic equipment can be used.
4. Fabricate structural and mechanical replacement parts rather than manifesting unique spares. Processes are being developed that permit the fabrication of parts from feedstock material that would be carried from Earth or, eventually, produced from in-situ resources. This allows manifesting of an appropriate mass of feedstock material rather than a large selection of unique, prefabricated parts. Manifesting prefabricated parts incurs the risk of carrying excess mass in the form of parts that are never needed and of carrying an insufficient number of parts when there is an unanticipated high-demand rate.
5. Implement a comprehensive preventive maintenance approach. An effective preventive maintenance program can help to avoid the occurrence of system failures and loss of availability. In addition, preventive maintenance can delay wear out, thus reducing the need to stock replacement parts. Extensive pre-mission study is required to define realistic schedules for preventive maintenance that allows for in-flight adaptability based on real-time experience.
6. Enable utilization of common LRUs, SRUs, piece parts, and components across an entire vehicle set. This will allow spare parts that would be carried on one vehicle to be used on another vehicle, or for system elements from one vehicle to be scavenged for use on another vehicle in critical situations. Interchangeability yields flexibility.
7. Use reconfigurable hardware. Using hardware that can be reconfigured to perform different functions as a mission progresses reduces the overall mass of hardware that is carried and minimizes the number of unique spares that are required. Optimally, a single generic part, such as a circuit card, can be easily reconfigured to perform in multiple locations.

7.2.1 Team support functions

Examples of ways in which the team time that would be required for overhead tasks and the mass for team support can be reduced include the following:

1. Launder clothes. Mass reductions can be realized if clothes are laundered and used multiple times.
2. Make inventory management transparent to the crew. Comprehensive and current inventory information is important to crew efficiency. The current manual barcode scanning method that is employed on ISS is cumbersome. A better approach might be the use of radio frequency identification (RFID) tags – active or passive – or use of machine-readable markings. Effective implementation of this technology may require some accommodation in vehicle hardware and system design to allow effective placement of sensors and transmission of RF signals, and to ensure non-interference with other systems.
3. Recycle waste products. Mass efficiency will be enhanced if waste products such as packaging and failed hardware can be recycled and reused.

7.2.2 Maintainability design requirements

Emphasis is on ease of maintenance, standardization and commonality of hardware, and cognizance of issues that would be specific to operations during space flight. The design themes that have emerged to enable the maintenance concept that was described above include:

1. Design for maintainability, graceful degradation, upgrades, and adaptation. For a spacecraft that must be maintained entirely by its crew, design for ease of maintenance is crucial. Systems should also be designed in such a way that they can continue to provide reasonable levels of functionality even after some failures have occurred. Systems should be able to accommodate upgrades – either hardware or software – without requiring total redesign. Finally, designers should seek opportunities to design hardware in such a way that it can perform a variety of functions in different mission phases. This can reduce the total amount of hardware that would be required and simplify its support.
2. Design and build for maintainability in the operational environment. The spacecraft structure will be subject to pressure and thermal differentials that can cause dimensional changes. These dimensional changes can affect clearances between parts, thereby making removal and replacement difficult or impossible. The design must consider

these changes and how they will affect the ability of a crew to perform maintenance tasks. Working in a weightless environment provides some advantages, but it also requires consideration of how a crew member will maintain stability and be able to apply loads required for tasks. For example, although a specific number of closeout fasteners may be necessary to secure hardware for dynamic phases of flight, far fewer fasteners may be necessary during the much longer, quiescent periods. The number of required fasteners should be minimized whenever possible.

3. Require commonality and standardization at hardware levels among major architecture elements. Mission architectures may require multiple elements such as crew transport vehicles, landers, SHABs, and surface vehicles. Every effort should be made to standardize hardware at all levels (LRU, SRU, component) among all architecture elements. This will simplify provisioning of spares, reduce the number of unique tools, and enable substitution between elements. As noted, this applies to hardware at all levels, including avionics circuit card assemblies; electronic components; other assemblies such as pumps, power supplies, and fans; fasteners; connectors; and other piece parts.
4. Require all hardware to be maintained should be internal – minimize extravehicular activity. EVA increases crew risk, is time-consuming, and imposes additional hardware design requirements. To the maximum extent possible, all hardware that may require maintenance should be located inside the vehicle in a pressurized environment to avoid the necessity of performing EVA maintenance operations.
5. Eliminate avionics line replaceable unit boxes – implement rack-mounted boards. Eliminating the boxes that are typically associated with avionics LRUs offers potential mass savings and facilitates access to the individual circuit cards for maintenance. Adoption of this approach, however, also necessitates consideration of cooling efficiency, physical isolation of redundant system elements, and the mass of cabling that would be required if avionics are centralized.
6. Do not combine Imperial and SI [System International] hardware. All hardware should be designed using a single system of units of measure (SI preferred) to avoid the need for multiple tool sets.
7. Provide robust diagnostics and post-repair verification. Efficient maintenance operations require quick, unambiguous fault isolation to the designated repair level. This can be accomplished with built-in-test (BIT) capabilities or with standalone test equipment. Whether via BIT or standalone test equipment, the hardware must be designed to be "testable."
8. Design systems to operate in a "keep-alive" mode with minimal power. In situations when power availability has been degraded or when power must be conserved, it is important that other spacecraft systems can remain functional with a minimal power demand. In this condition, the system may not perform its function but retain the capability to do so when additional power is provided. This is similar to interplanetary probes that revert to a "survival mode" during severe radiation events to protect (by power off) vulnerable hardware.
9. Design systems to enable isolation of faulty components to preclude loss of entire system. Systems should be designed so that single failures do not cause total loss of function.
10. Design systems so that pre-maintenance hazard isolation is restricted to the item that is being maintained. When power, pressurized gas, coolant, or other potentially hazardous resources are isolated from system hardware elements to make them safe for maintenance, isolation should be limited to the smallest possible set of hardware to minimize impacts to overall system availability.

7.3 The habitat service systems views

The controlled habitat service system includes life support systems that humankind builds on top of Earth's life support system (the larger habitat):

1. **Natural ecological [habitat] systems** - The planet Earth's life support ecological-service systems.
   A. **Human controlled [habitat] systems** - The human life (and other) support-service systems. In other words, the societal access-fulfillment system (i.e., community's habitat service system platform).

7.4 Indicators of a co-habitable service system

Effectively constructed habitat service systems are characterized by the following principles (following these principles enable more life range choices):

1. **The life-coherence principle:** The ultimate organising principle of any life-coherent society (or economy) through generational time is maintained (or secure) access to life [fulfilling] services. Any social (or economic) system aligns or does not to

the extent that it maintains the production and distribution of life services.

2. **The service-system principle:** A service system is a service system, if and only if, it enables life capacities/abilities not possible without it (e.g., food, water, shelter, computation, etc.). Claimed services that disable (or do not enable) life capacities and abilities are not means of life (e.g., commodities). Any service that does not directly or indirectly provide a life service is uneconomic (or, anti-economic to the extent of life resources wasted on the commodity's production and consumption).
3. **The provision principle:** The provision, or the deprivation, of each and all of these life services is measurable by greater/lesser sufficiency (e.g., of clean water, life space a, meaningful work, hours of work, etc.).
4. **The performance principle:** The measure of the overall performance of any society (or economy) is its global access commons developmentally expressed as access to life services (including the work share required to provide them). Given what is available and what is known, what is possible? And, how does that compare to all previous states of the society (or economy, or even, another socioeconomic structure)?
5. **The memory principle:** The primary base ("capital") of any society (or economy) is information about creating and maintaining life services without loss in cumulative capacity through time. The societal system specification is the integration of a society's understandings and decisions.
6. **The efficiency principle:** The efficiency of any product, tool or process increases, and only increases, to the extent that:
   A. **Ecological efficiency** - Inputs and throughputs function to enable the provision of life services with diminishing waste and externalities (e.g., organic farming methods, industries directed towards 100% recycling).
   B. **Physical input-output efficiency** - Reduced inputs of materials/energy/space/mandatory work time produce same or greater means of life outputs (e.g., wheel and pulley structures, cooperative organisation of work/leisure requirements, lower labour/fuel-per-unit machines).
   C. **Human development efficiency** - Capability development of productive agents enables more life goods, life-time, and/or life-range choices than before (e.g., by habitat service sector, such as education, healthcare, and intersystem work). More free time is more life range choice.

8 [Engineering] Construction

Construction is a process of work by creating building or infrastructure to support the requirement of society. This process starts from the planning, design, financing and continues until the project is ready for use include problem recognition to the implementation of fully operational solution. Construction can be referring to the several sectors such as building (residential and non-residential), infrastructure (roads, bridges, public utilities, and dams) and industrial (process chemical, power generation, mills and manufacturing plants).

Building construction is a process of adding a new structure to real property whether for existing or new building. This process was involved with complex documentation that call as construction documentations (CDs) that can be divided into several components such as:

1. A graphical representation of the building (which includes 2D floor-plans, elevations and cross-sections, and possibly 3D CAD models)
2. A set of specifications that dictate the quality of the components and finishes of the building
3. A legal document that highlights the project expectations.

How to reduce uncertainty in a project?

1. Coordination (cooperating).
2. Visualization (modeling).
3. Documentation (explaining).
4. Planning (scheduling).

Every construct[-able] societal information system has:

1. The data of an underlying methodology for the societal systems construction [of a habitat system].
2. The perceived problem situation for re-construction [of a habitat system].
3. The individual consciousness involved and affected by the use of the approach to construct [a habitat system].

9 [Engineering] Societal information

To engineering, society is, in part, an information system capable of representing the real world as visual information. A societal engineering system must be a combination of:

1. Conceptual information.
2. Spatial information.
3. Control procedures for associated information into a visualization in time.

9.1 Societal information system construction

Information system integration involves the following layering:

1. In an information system:
   A. Objects "store" data.
   B. Data "stores" meaning.
   C. Meaning "stores" utility.
   D. Utility "stores" memory.
   E. Memory "stores" remembrance.
2. In a conceptual information system, concepts (or concept-objects) store data about meaning (or, perceivable as meaning to consciousness). Therein, concepts can store data about families/patterns of meaning (i.e., their properties). Concepts form patterns of meaning in the awareness of consciousness.
3. In a spatial-information system, objects store data about shape; objects can store data about families of shapes (i.e., their attributes).
4. In a physical environment, objects (Read: a shape) can potentially be put within other objects (e.g., putting 'water' in a 'glass', or one Matryoshka doll inside another). In a conceptual environment, concepts (Read: with meaning) can potentially be put (embedded) within other concepts (e.g., putting the meaning of a 'bed', or "place to sleep", within the meaning of a 'home' or 'dwelling').

Information systems are operationalized through the functional integration of concepts and objects, which become technology for a social population:

1. The subject of a sentence/argument is a concept, a meaning. The rules by which the sentence is constructed is its syntax (ordering logic).
2. The object is something to point to (to point out to someone else, to visualize for oneself or another). The rules by which the vision is constructed is its intelligence (visual logic).

3. Technology is the reproduction of an objective in object form, from a function involving concept and object integration.

9.2 Synthetic environment data representation and interchange specification (SEDRIS)

SEDRIS (Synthetic Environment Data Representation and Interchange Specification), SEDRIS Spatial Reference model (SRM)

In order to support the unambiguous description of environmental data (of a conceptual and spatial form), the SEDRIS SRM (spatial reference model) specifies both a Data Representation Model (DRM) and an Environmental Data Coding Specification (EDCS). These address how to describe "environmental things", but explicitly avoid defining how "environmental things" are located with respect to one another and with respect to non-environmental "things". The SEDRIS SRM (spatial reference model) addresses this need and provides an integrated framework and precise terminology for describing spatial concepts and concepts of operations on spatial information.

The SEDRIS RM (reference model) is comprised of a set of Reference Frames (RF), their inter-relationships, and unambiguous definitions of methods for specifying and inter-converting location (including directional and orientation) information among SRFs. Additionally, those methods are documented in terms of detailed algorithms and subsequently reduced to efficient, accurate, and portable implementations.
Algorithms form a coordinating framework for simulation of a physical information space.

1. Algorithms exist for spatial operations.
   A. Space is shaped with physical objects.
2. Algorithms exist for informational operations.
   A. Space is shaped with informational objects.

9.2.1 A multi-scale integrated model of ecosystem services (MIMES) and human coupling

SEDRIS is a modeling tool that can incorporate user input and biophysical data sets for evaluation of ecosystem services and decisioning by producing an integrated multi-scale model of ecosystem services MIMES and human coupling. In order to accomplish this form of environmental data model, the information system must:

1. Simulate ecosystems and socio-economic systems in space.
2. Simulate these systems over time.
3. Simulate the interactions between these systems and human service systems through coupling.
4. Simulate the coupling over time.

In order for the model to function, ecosystem services are (Read: assumptions):

1. Ecosystems are the structures and processes that generate functions.
2. Ecosystem functions of value to humans are ecosystem services.

Characteristics of ecosystem services include, but are not limited to:

1. Structures are not transformed into the produced services (e.g. lumber is not the service, the production of trees is).
2. The source of most energy for services is solar energy.
3. Availability depends on ecosystem functioning and typically is not controlled by humans.
4. Ecosystems supply ecosystem services, which humans can harness (use).

The elements of an information-spatial model include, but are not limited to:

1. Requirements (information and spatial).
2. Acquisition (information and materials).
3. Processing (information and materials).
4. Usage (service-information and service-objects).

9.3 Spatial and conceptual information

Materiality is shape (Read: literal, physical), which is representable as information. Conceptuality is information, which is representable as materiality (e.g., a house-building or "money"). Spatial and conceptual information come together in the form of an information system for society, with a pure information set, and a material information set (that represents either the current real-world, the past real-world, or potential possible real-worlds).

The two dimensions could be otherwise called:

1. Hard[ware] - material system (spatial, physical).
2. Soft[ware] - informational system (conceptual, mental).

For movement in a material system there must be a physical mechanism (material process) to have a complete explanation. Movement can be described, and movement can be explained. For change in a conceptual system there must be an information mechanism (information operation). Change can be described, and change can be explained.

The societal system sub-component naming involves:

1. The components of a material system are often referred to as architecture (infrastructure).
2. The components of an information system are often referred to as data (computation).
3. The components of a meaningful or relational system are often referred to as concepts.
4. The components of a material system are often referred to as objects.

9.3.1 Spatial information

Spatial information processing requires a coherent capability to describe the geometric (spatial logic) properties of:

1. **Position** (location of).
2. **Direction** (motion toward).
3. **Distance** (space between).

It is from these spatial properties that spatial alignment and navigation are calculable, and from which a material service may safely exist.

Spatial information may be spatially referenced to:

1. Local structures and regions.
2. The Earth as a whole.
3. Other celestial bodies. or
4. Objects defined within synthetic visual contexts (e.g., virtual realities).

In each of these cases, a spatial reference frame is defined in relation to logic properties (e.g., spatial logic, conceptual logic, etc.).

9.3.1.1 Spatial data

A.k.a., Geo-referenced data, geodetic data, geodetic datum, spatial environmental data, conceptual (social) environmental data,

Spatial data describes the absolute and relative location of geographic (earth or spatial) features. Spatial data describes:

1. The characteristics of spatial features.
2. Quantitative and/or qualitative data.
3. Attribute data is often referred to as tabular data.

Geo-referenced data include, but are not limited to astronomical, orbital, geomagnetic, and local observations whose reference frame may be fixed with respect to observer, solar, celestial, or other positional standards rather than, for example, the equator plus a prime meridian on an Earth Reference Model (ERM) surface. Other Object Reference Models (ORM; e.g., the moon or the NASA Space Shuttle) may also be used.

Common geographic, geospatial (positional) representational data types include but are not limited to:

1. Points *(primary class)*:
   - id,
   - x, y,
   - $m_1..m_n$
2. Lines *(primary class)*:
   - id,
   - $x_1,y_1..x_n,y_n$
   - $m_1..m_n$
3. Areas *(primary class)*:
   - id,
   - $x_1,y_1..x_n,y_n..x1,y1,$
   - $m_1..m_n$
4. Rasters *(primary class)*:
   - $x_1,y_1z_1..x_n,y_n,z_n$
   - $m_1..m_n$
5. Routes *(extended class)*
   - id
   - $x_1,y_1..x_n,y_n$
   - $m_1..m_n$
1. Regions *(extended class)*
   - Poly list
   - id
   - $p_1..id,p_n$
2. Instantaneous points
   - id, x,y,z,t,m
3. Time duration points
   - id,x,y,z
   - $t_s..t_e$
   - $m_1..m_n$
4. Time series points
   - id,x,y,z
   - s
   - $t_1..t_2$
   - $m_1..m_n$
5. Time duration vectors
   - id
   - $x_1,y_1,z_1..x_n,y_n,z_n$
   - m1..m2
6. Time duration areas
   - id
   - $x_1,,y_1,,z_1..x_n,y_n,z_n$
   - x_1,y_1,z_1
   - $t_1..t_2$
   - $m_1..m_n$

9.3.1.2 The material data set

Material object observations generally have four operational dimensions:

1. The first two are the x and y horizontal spatial coordinates, referring to some predetermined

standard coordinate reference system (CRS).
2. The third is the temporal coordinate, t, the moment – according to some predetermined standard calendar and time zoning system – when the soil was observed.
3. The fourth operational dimension is the material observation elevation, z, as measured using some predetermined standard scale, e.g. metres.

At a point in (geographic) space and time, [x, y, t], or in space, time and depth, [x, y, t, z], a material observation is accompanied by an attribute space. The latter is a multi-dimensional space defined by a set of attributes of the environment (e.g., for a soil observation it may be land use, slope, parent material, or soil layer pH, cec, carbon content).

9.3.1.3 Spatial data collection

How are current data collected on current real world objects? The 3D geometry of real world objects are observed and collected through surveying technology, such as total stations, terrestrial/airborne laser scanners, or techniques from photogrammetry. The common name for this collection of techniques is known as, 'remote sensing'. Remote sensing processes record, measure, and interpret imagery and digital representations of [energy] patterns derived from non-contact sensor systems.

9.3.1.4 Spatial data interoperability

Interoperability of spatial data is facilitated through the adoption of a common and widely-known Spatial Reference Model (SRM) that allows the context in which coordinates, directions, and distances are defined to be known exactly, and converted accurately into multiple definitions and representations of geo- and non-georeferenced space.

9.3.2 Conceptual information

Conceptual information processing requires a coherent capability to describe the conceptual (mechanical-state logic) properties of:

1. **Condition** (quality of).
2. **Function** (behavior of).
3. **Intention**[al progress] (expectation of).

It is from these mechanistic properties that state change and intentional alignment are calculable, and from which an effective service may safely exist.

Conceptual information may be conceptually referenced to:

1. Local behaviors.
2. Regional behaviors.
3. The population as a whole.
4. Digital (software) systems.
5. Mechanical (hardware) systems.

In each of these cases, a conceptual reference frame is defined in relation geometric properties.

9.3.3 Temporal information

For any object or system to be realized in the real world, it is necessary to specify the time and temporal reference frame to which the spatial position and/or conceptual condition refers, and the time for which the spatial or conceptual reference frame is defined.

9.3.4 Spatial and conceptual interoperability

Interoperability of conceptual data is facilitated through the adoption of a common and widely-known Conceptual Reference Model (CRM) that allows the context in which conditions, functions, and intentions are defined to be known exactly, and converted accurately into multiple definitions and representations of real- and non-real-referenced space.

Interoperability of spatial information requires that:

1. Spatial reference frames and ORM/ERMs be defined such that coordinates and angular measures describe position and orientation data uniquely, and
2. Mechanisms exist for such data to be converted/transformed between alternative spatial reference frames, should this be required.

Interoperability of conceptual information requires that:

1. Conceptual reference frames and ORM/ERMs be defined such that conditional and functional measures describe intention and orientation data uniquely, and
2. Mechanisms exist for such data to be converted/interpolated between alternative conceptual reference frames, should this be required.

Existence has two delimitations to embodied consciousness:

1. The basis of physicality is a limited physical boundary.
2. The basis of information is data (i.e., a delimited meaning boundary).

9.3.5 Spatial and conceptual reference frames

A.k.a., Where is the point?

Accurately locating objects is key to the operation of any [service] system which contains information about real-world entities. Consistency (standard) in description, nomenclature, and the treatment (application) of models

of the earth and related spatial and conceptual reference frames and coordinate systems is critical to achieving effective data interchange and system interoperability, which are required for optimization of global human [service] fulfillment. The S-/C-RM provides the means to define a unified approach to representing real world conception location information, and precisely relating different descriptions of such location. All information here can be represented in databases, and potentially, simulated.

Any ability to control physical or conceptual motion comes from having a set of coordinates within a coordinate system. A coordinate system is a collection of rules by which values may be used to relate (process) an object to a unique (coordinate system) origin location. Coordinate Systems are a collection of rules by which values may be used to spatially or conceptually relate a location to a unique (coordinate system) origin location. A coordinate system specifies a mechanism for locating points within a reference [coordinate] frame.

> **CLARIFICATION:** *A coordinate frame (or simple, reference frame or frame) is specified by an ordered set of mutually dependent direction vectors; thus, a reference frame has an identifiable center. When producing or using positional or conditional data, one [controller] needs to understand both the reference frame and the coordinate system being used.*

In physics, a frame of reference (or reference frame) consists of an abstract coordinate system and the set of physical reference points that uniquely fix (locate and orient) the coordinate system and standardize measurements (for some form of intentional control by consciousness). Coordination system formalization [as mathematics] is relatively simple. A coordinate system in mathematics is a sub-conception of geometry (applied algebra), a property of manifolds (in physics even, these are appropriately called configuration spaces or phase spaces; appropriately because coordinate systems allow for controllability, and thus, the benefit of re-configuration). Mathematically, the coordinates of a point r in an n-dimensional space are simply an ordered set of n numbers: r = {x1, x2, ..., xn].

> **INSIGHT:** *Through coordination there may exist greater development of individual function.*

9.3.5.1 Coordinates

Coordinates are:

1. Linear or angular quantities that designate the position of a point in a [coordinate system] reference frame. By extension, they also designate the position of a point within a spatial reference frame.
2. Conditional or intentional qualities that designate the state of a point in a [coordinate system] reference frame. By extension, they also designate the condition of a point within an intentional reference frame.
3. Data representationally modeled (DRM) locations of spatial position and/or intentional condition.

Reference modeling is the production of:

1. A reference model (spatial-, conceptual-type) is a well-defined set of
   A. reference frames (spatial, conceptual),
   B. object reference models, and
   C. coordinate systems,
   D. that allows coordinates to be specified
   E. succinctly, and
   F. converted accurately between different [spatial and/or conceptual] reference frames.

Spatial reference frames (SRFs) are:

1. Reference Frames serve to locate coordinates in a multi-dimensional space (generally either two- or three-dimensional). They are specified in two parts to any SRF.
   A. Object reference model - A geometric description (model) of a reference object embedded in (and serving to orient) that frame referred to as an Object Reference Model (ORM)
      1. An Earth Reference Model (ERM) is a special case of an ORM
2. Coordinate system computation - A Coordinate System specifying how a tuple of values uniquely determine a location with respect to the origin of that frame. By extension, that tuple also specifies a location with respect to the reference object.

9.3.5.2 The Cartesian coordinate system

A.k.a., The geometric-planar coordinate system, a spatially alignable coordinate system.

The Cartesian coordinate system (the planar coordinate system) is based on an ordered set of mutually perpendicular axes formed by straight lines. The point of intersection of the axes is termed the origin. Alignment - deviation from origin - can be determined to some degree of certainty. The directions of successive axes are normally related to each other by a right hand rule.

1. Cartesian Coordinates (two-dimensional) uniquely locate points on a plane using a doublet of values, e.g., (x, y).
2. Cartesian Coordinates (three-dimensional) uniquely locate points within a volume using a triplet of values, e.g., (x, y, z).

Since the Earth is an important reference object in our spatial environment, many Spatial Reference Frames will

consist of an Earth Reference Model plus a Coordinate System.

Earth resource frame includes:

1. Topographic Surface is the interface between the solid and liquid/gas portions of the Earth. A topographic surface corresponds to the surface of the land and the floor of the ocean.
2. Earth Reference Model (ERM) is a specification of the mathematical shape of the Earth, usually in terms of a combination of ellipsoidal and equipotential (geoidal) surfaces. It excludes the topographic surface, and therefore generally corresponds with mean sea level.

Since human needs are an important reference object in our conceptual environment, many Conceptual Reference Frames will consist of a Human Reference Model plus a Coordinate System.

Human needs frame include:

1. Habitologic Surface is the interface between the human and ecological materials of the Earth. A habitologic surface corresponds to the surface of the human body and of the local environment.
2. Sociologic surface is the interface between the individual and social access to habitat surfaces.
3. Economic surface is the interface between the human and habitat re-configuration of surfaces to more greatly meet human habitat needs.
4. Real World Reference Model (Real-World Information System) is a specification of the conceptual shape of the human need frame as a real world information model, usually in terms of a combination of conceptual and material surfaces. It excludes the topographic surface, and therefore generally corresponds with mean functional level.

Other reference frame models common to society are:

1. Object reference model (ORM).
2. Service reference model (SRM).
3. Habitat reference model (HRM).

A coordinate system exists to perform useful operations on coordinates, including but not limited to:

1. Direction determination (azimuth and elevation angle calculation
2. Coordinate Conversion is the process of determining the equivalent location/condition of a point in a reference frame (spatial/conceptual), which is based on the same object reference model (e.g., ERM), but a different coordinate system.
3. Coordinate Transformation is the process of determining the equivalent location/condition of a point in a reference frame, which is based on the same coordinate system, but a different object reference model (e.g., ERM).
4. Converting coordinates between two arbitrary Reference Frames may require both Coordinate Conversion and Coordinate Transformation.

There are some common types of operations errors in coordinate systems:

1. Formulation (algorithmic) errors in algorithms for coordinate operations.
2. Implementation errors includes errors due to sequencing (mathematics) and software implementation.
3. Usage errors includes errors due to extension of projection-based reference frame beyond reasonable limits.

9.4 Integrated informational material modeling

In the technical world, one of the most well-known concept models is TCP/IP, the foundation of the global Internet communications system. The generally named Internet protocol suite is, the conceptual model and set of communications protocols used in the Internet and similar computer networks. The information set is commonly known as TCP/IP because the "suite" is primarily composed of the Transmission Control Protocol (TCP) and the Internet Protocol (IP).

The following is a stacked model of informational and material flows in society. [Flow] of information and materiality transmission control protocol ([F]TCP) stack/ model (adapted from Internet TCP/IP model):

1. Physical (material surfaces standards; materialization protocol).
2. Data Link (information standards; information protocols).
3. Network (global habitat decision standard; decision protocol).
4. Transport (land and motion systems standards; transportation and communication protocols).
5. Session (local habitat service operation systems standard; habitat operational process protocols).
6. Presentation (useful objects).
7. Application (useful services).
   A. Application (Habitat contribution service platform; messaging, collaborating, addressing, deciding, tasking; monitoring as support-view; and coordinating as over-view).
   B. Transport (Resource and human transportation, fabrication, and cycling system; the material

surface system re-composed for differing human service purposes).

C. Network (intra-habitat integrated-infrastructural network protocols; inter-habitat integrated-infrastructural network protocols).
D. Link (physical habitat service operational surfaces and information flows).
E. Physical (machines and humans; drawings and documentation).

9.5 Spatial reference model (SRM) standards

A.k.a., Spatial coordinate system.

A spatial coordinate system is a means of associating a unique coordinate with a point in object-space. It is defined by binding an abstract coordinate system to a normal embedding. A spatial reference frame is a specification of a spatial coordinate system for a region of object-space. The spatial embedding of a real-world surface coordinate system is:

- The ISO/IEC 18026:2006 Spatial Reference Model (SRM) International Standard.

ISO/IEC 18026:2006(E) allows new concepts to be specified by the registration of new entries to the standard. New entries to the standard are registered using the established procedures of the International Register of Items. The registry is also a valuable resource for searching and finding specific spatial reference frameworks (SRFs), object reference models (ORMs), coordinate systems (CS), and other registrable constructs within the reference mode (RM).

Aspects of ISO/IEC 18026:2009 apply to, but are not limited to:

1. Mapping, charting, geodesy, and imagery.
2. Topography.
3. Location-based services.
4. Oceanography.
5. Meteorology and climatology.
6. Interplanetary and planetary sciences.
7. Embedded systems.
8. Modelling and simulation.

9.5.5.1 Highly-related ISO Standards

These standards further contextualize spatially referential information:

1. ISO/TC 211 - Geographic information/geomatics
2. ISO/TC 184 - Automation systems and integration
3. ISO/TC 184/SC 5 - Interoperability, integration, and architectures for enterprise systems and automation applications

9.6 What is a spatial coordinate system?

A.k.a., spatial coordinate reference system.

Spatial coordinate reference systems are designed to enable the position of points to be uniquely described over varying sizes of information or geographic area. A coordinate system that enables every location on and around Earth to be specified by a set of numbers, letters or symbols.

A [spatial] reference system (SRS) or coordinate reference system (CRS) is a coordinate-based local, regional or global system used to locate [geographical] entities.

Example geographic (a.k.a., earth surface positional) reference systems include:

1. A Grid Reference System (grid or projected coordinate reference system), is a means by which to reference locations [on the Earth's surface using a two dimensional Cartesian coordinate system referenced to a map projection. A grid coordinate defining a location consists of and is written as an ordered pair of x and y values expressed in linear units. Most of these grid reference systems use the meter as the unit of measure and define an easting (x) and northing (y) referenced to a series of transverse. A
2. Geographic Reference System (graticule) is a means by which to reference locations on the Earth using a system of angles. A geographic coordinate defining a location is usually expressed in angular units of latitude and longitude. Latitude (φ - phi) is the angle between the equatorial plane and the straight line that passes through the point in question and the center of the reference shape (WGS84 ellipsoid). Longitude (λ - lambda) is the angle east or west of the reference meridian (Greenwich Prime Meridian) to another meridian that passes through the point in question. The most common standard geographic reference system is latitude and longitude expressed in sexagesimal (base 60) numbering system.

A coordination system (coordination management) is a set of programs to perform operations on data, such as store and retrieve data.

9.6.1 Database operations

A database coordination system maintains the collection of all societal data, and a set of programs to access, store, retrieve, and otherwise process societal data. Therein, the decision system is decision support software with some designed interface. The social system specification is a database of societally relevant information,

coordinated by software.

9.6.2 Environmental database

An environmental database is an integrated set of data elements, each describing some aspect of the same geographical region. It often includes additional data describing simulation elements and events expected to take place during the interactions in that environment. For example, data representing trees in a forested region may be found in a database; but in addition, the geometry of vehicles that might drive through the trees during a simulation would also be found in an environmental database. The key phrase in the above definition is "integrated set of data." It is the integration, infusion, and tailoring of varied data sources that creates a full database, and sets it apart from databases that only use an existing raw data source as-is.

9.7 Spatial objects

Spatial objects (in the real world or a simulation) have shape as an attribute in their table. They have geographic (or potential geographic) location. They are a point, line, TIN, raster, etc.

9.8 Temporal-spatial coordination

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. The real-world has a time parameter; and at a conceptual level, it is composed of dynamic systems, which are systems that factor a time parameter. These systems reduce to the case of a static relationship by fixing a value for the time parameter. Material/physical object reference model bindings (associations) are often based on physical measurements of objects or systems that change with time. Time is also used to identify the decisions for which these measurements are applicable.

A 'temporal' coordinate system (CS) is a 'Euclidean 1D' CS (see Table 5.35, [standards.sedris.org]) that assigns distinct coordinates to distinct times so that larger coordinate values are assigned to later times. In relation to human tasking, this culminates in a universally, globally coordinatable time system; a temporal coordinate system that enables a unique temporal coordinate to be assigned to every recorded or potential event; thus, necessary for global tasking. For example, in the early 21ste century society, Universal Time (UTC, [standards.sedris.org]) (see 6.2.4, [standards.sedris.org]) was an inter-national time standard.

Herein, times and dates refer to UTC unless explicitly indicated otherwise (often for extra-societal coordination).

9.8.1 Temporal spatial standards

The most well-known temporal-spatial standard is SEDRIS:

- ISO/IEC 18026E: The SEDRIS Standard [standards.sedris.org]
  - Information technology spatial reference model [standards.sedris.org]

9.8.2 SEDRIS

SEDRIS (Synthetic Environment Data Representation and Interchange Specification) is an international data coding standard infrastructure technology created to represent environmental data in virtual environments. A SEDRIS system is coordinated through socio-technical projects.

A virtual environment is a synthetically visualization as a representation of existence. Today, virtual environments are sustained through combinations of hard- and soft ware. In any given society, visualization technology may be installed within an organization's existing IT infrastructure and controlled from within the organization itself. From a central interface the technology creates an interactive and immersive experience for teams of users. Visualization technology enables cooperation it be most effective and efficient, because individual understandings can be aligned to a commonly shared vision, and a commonly shared vision can be aligned to individual understanding (through visualization). Visual environments tend to focus on needs of users and issues that are actually relevant and persistent, because that which is the problem and that which is the solution is visually clarified to be so by everyone.

Virtual environments can be persistent and representational of a working, conditional real-world environment.

Environmental data represented by SEDRIS may be concrete (physical, positional and compositional) , such as trees and mountains, or abstract (conceptual, conditional and intentional), such as a technology operations procedure (behavioral intention) or the state of a service system (behavioral condition).

Here, visualization facilitates the accurate and coherent exchange of data for reuse and wider scrutiny. Whereas a simulation is a dynamic visualization, a flow diagram (or concept model) is a static visualization.

SEDRIS tutorial data references include:

1. SEDRIS Technologies tutorials [sedris.org].
2. SEDRIS Data [data.sedris.org].

9.9 What are data models used for?

A data representational model, or simply a data model, is a notation method for describing data. The data model provides a description that enables its users to understand what data is present and how it is organized. A data format specifies the actual bytes used to store data on a storage medium. A specific implementation is defined for how the data objects are to be structured and identified on the medium. There are multiple ways

of implementing a data format for a specific data model.

9.9.1 Semantic logic

Semantic logic provides an explicit representation of the conceptual relationships between information objects. Topology provides an explicit representation of the spatial relationships between physical objects (e.g., connectivity and adjacency). Semantic logic (semantic reasoning, rational reasoning) provides the explicit representation of a conceptual relationship, instead of having to derive the conceptual relationship by doing discrete (and axiomatic) logic proofs. Topology provides the explicit representation of a spatial relationship, instead of having to derive the spatial relationship by doing geometric calculations. A topology and semantic logic are ways of pre-computing the answers to spatial and meaningful-informational relationship questions.

The only sensible use of the term "semantics" refers to the meaning of expressions in a language. Such expressions can be single symbols (the "words" of a language) or symbol combinations. As the term implies, they are used to express something, i.e., to communicate meaning. Neither concepts nor entities nor properties nor processes have semantics, but expressions in languages describing them do. In an information system context, many languages need semantics: natural languages, programming languages, schema languages, query languages, interface specification languages, workflow modeling languages, user interface languages, sensor modeling languages, and others. The symbols and expressions of information system languages may be produced and consumed by machines and humans. The languages used in information systems are not natural languages, even if they use natural language terms; instead, they are technical language, which is the results of 'precision of language' determinations sufficient to design and operate a working system. Many of these languages allow users to define new symbols (for individuals, types, properties, relationships etc.). Attaching meaning to language expressions is a conceptual phenomenon. Natural language symbols and expressions evoke concepts in human minds. If these concepts are similar to those which the symbols and expressions were meant to express, communication works. Expressions come to represent entities (as well as properties, relationships, and processes) in the world. This fundamental ternary meaning relationship between symbols, concepts, and entities is captured in the so-called semantic (or semiotic) triangle.

9.9.2 Simulation

To simulate is to make up something similar to an original. One primary use for a synthetic environment is the representation of the natural environment at a specific geographical location. Therefore, the synthetic environment includes the terrain, terrain features (both natural and man-made), 3-D models of vehicles, personnel, and certain terrain features, the ocean (both on and below the surface), the ocean bottom including features (both natural and man-made) on the ocean floor, the atmosphere including environmental phenomena, and near space. In addition, the synthetic environment includes the specific information attributes of the environmental data as well as their relationships.

9.9.3 Active data models

Active data models are operations used to coordinate data into/within a structured document with a centralized data repository.

A common data element classification for an environmental database is:

1. Terrain surface - Surface geometry.
2. Terrain features - structures found on and within the terrain, such as vegetation, hydrology (rivers), roads, rockets, terrain artifacts, etc.
3. Buildings - buildings as structures in the area.
4. Objects - structures other than buildings in the area.
5. Textures, images, and colors - surface composition.
6. Environmental models - environmental phenomena as smoke, rain, haze, etc.
7. A database that is designed to support the full spectrum of environmental understanding required to fulfill a wide-ranging human need-service habitat application.

In order to support the unambiguous description of environmental data, an environmental data coding specification:

1. Data representation mode (DRM) - how to describe "environmental things" in terms of data modeling constructs meaningful to simulation developers (e.g., geometry, feature, image, topology, and data table), it explicitly avoids specifying "where" the "environmental things" are, and enumerating all of the "environmental things" that these data modeling constructs could be used to represent.
2. Spatial reference model (SRM) - captures and unifies the spatial models used. These models include inertial, quasi-inertial, geo-based, and non-geo-based (purely arbitrary Cartesian) systems. The SRM provides a unifying mechanism for specification and inclusion of any spatial reference frame and coordinate system. Its algorithms are designed to retain a high degree of accuracy during transformation and conversion operations (1mm accuracy).
3. Environment data coding specification (EDCS) - The EDCS provides a mechanism to specify the environmental "things" that a particular data

model construct is intended to represent. That is, a "tree" could be represented alternatively as a <Point Feature>, an <Aggregate Geometry>, a <Data Table>, a <Model>, or some combination of these and other data modeling constructs. Which of these the data modeler (i.e., the data provider of a SEDRIS transmittal) chooses is orthogonal to the semantic of the "thing" that is represented (and its location). The provision of such a "thing" in a SEDRIS transmittal pre-simulation must result in a shared understanding of "what the thing is and what it potentially means" to all participating applications.

9.10 Environmental data standards

Environmental Data Coding Specification (EDCS):

1. ISO/IEC 18025 provides mechanisms to unambiguously specify objects used to model environmental concepts. A functional interface is also specified. [standards.sedris.org]
2. ISO/IEC 18024-4: EDCS language bindings - Part 4: C - Access to the codes defined by ISO/IEC 18025 is through an API.

Environmental data coding standards. All data coding standards focus on meeting the needs of modeling and simulation of the environment for a community of users.

9.10.1 Environmental data-base construction

There are different approaches to database construction depending on the available tools, intended simulation platforms, system requirements, available data sources, design preferences, and application-specific needs. As a result, there is no standard methodology for creating simulation databases. For the most part, however, some general phases are common to all database construction processes. Sometimes these phases overlap or are combined, sometimes one is left out because there is no added benefit, and sometimes their order of execution is changed or done in parallel. With those caveats, we can break the construction process into the following six phases.

1. Requirements definition - As in any design and implementation, this is critical for database construction because of the varied levels of knowledge between designers and end users, vastly different construction techniques and system constraints, and the lack of a standard terminology common to the simulation community. Without the involvement of both users and designers throughout the entire construction process, the acceptance or desirability of the resulting database will be left to chance.
2. Data collection - Collecting source data is continuous throughout. Source material can span the range of paper maps, digital elevation products, images and photographs, feature data, 3D models, verbal reports, tabular data, satellite imagery, attributes, weather data, topological data, existing animation and special effects, and a host of other data sources.
3. Reasoning (explaining model, value-adding) explanation - visual understanding integration into the given most well-defined model. Often the source data needs to be further analyzed or refined before it can be used.
4. "Assembling" the database - Once all data sources have been put in an acceptable form, the various data elements are then integrated and assembled into the database one at a time. This may mean combining a surface with a particular texture, color, or attribute; or conforming the 2D features, such as roads or rivers, to the underlying 3D terrain surface. There are many other similar steps that take place during this phase. Applying real-time performance constraints is one of those. The key, however, is the notion of infusing and integrating inherently varied data sources into a single cohesive database.
5. Compiling and execution (or transmission) of the database.
6. After sufficient simulated iteration of the previous steps, the environmental database is ready for use and testing in the real world. In this step, the database is compiled from its editable data structures into platform-specific data habitat service sub-structure.

It is possible to conduct research into shared ways to represent environmental data was begun in the 1980s in order to permit distributed simulations to work together.

9.11 Unified coordinate information systems algorithms

This report contains guidelines for the development of computationally efficient algorithms for computing informational and spatial operations. A spatial operation is a:

1. Coordinate transformation.
2. A coordinate conversion.
3. An azimuth determination.
4. A distance calculation, or other.
5. Computations associated with elliptical trigonometry and map projections.

An informational operation is a coordinate transformation, a coordinate conversion, an value

determination, a fulfillment calculation or other computations associated set decisioning.

10 [Engineering] Geoinformatics

A.k.a., Geographic information, geomatics, real-world space computation, an information-based spatial-visual system; geocomputation.

Geoinformatics is a discipline of systems science that uses knowledge and technologies to support the processes of acquiring, analyzing, and visualizing geospatial data (Question: how can the data be added to a simulation of the real-world?). In concern to functioning, the integrated city system ("smart town") is analogous to the human body; it divides functioning into cells and into grouping of cells that share resources and coordinate fulfillment without trade. Activities take place in time and dimensional space; some of the activities are built into the environment as infrastructure. As a type of infrastructure, buildings are an environmentally controlled space where activities take place within. There is a natural organismal ecology. Survey sensors collect environmental and individual issue data. A computational control/conditional system that processes data to give an objective. A transportation network distributes resources, humans, services and products for global accessibility. A circular integrated city grid framework individuates the circle into functional cells. In simulation, the cells overlay a real-world datum as functional habitat material-service boundaries. Additional geoinformatics data layers may overlay the model.

3D geoanalytics and a geo-spatial (visualization, simulation, analytics platform) interface to multiple layers of extant and possibly extant reality. Any information that can be spatially organize, a GIS is the name given (currently) to the system that organizes, visualizations, and databases the system.

1. Surveying:
   A. Geoanalytics for the habitat service system network relate individually coordinated habitat service [city] systems within an open resource-access network, unified by means of a pre-planned procedural generation (i.e., visualization) of resource survey data de-constructed by habitat service input-output sub-service systems (i.e., integrated city system) that share resources among a commonly integrated information system.
2. Engineering:
   A. Geospatial (geo="earth"; spatial="4D.."; material world) engineering.
3. Computation:
   A. Geographic information system (data structuring and visualization of the world)
      1. Geographic information science (discovery and survey of the world, geomatics).
      2. GeoDesign is the concept of designing and planning a geographic (a.k.a., spatial, real-

world, locatable experiential) environment.

10.1 Habitat service planning

A.k.a., Smart city planning.

The purposeful usage of geoinformatics is to understand, to design, and to monitor a population's habitat service system. A controlled 'habitat' is a bounded system that transduces ecological (and socio-environmental services), and can be 'planned'. That which is planned within the habitat is services, primarily for humans and with consideration to the larger ecology that facilitates human flourishing. For habitat service planning to be, and remain, effective for scalable and global human fulfillment, it must approach the design of the habitat axiomatically and structurally "bare" (i.e., without artificially imposed prior social constructions), and at a root-fundamental level:

1. Plan as if "you" are starting from construction scratch on a given terrain (i.e., there are no prior constructions).
2. Separate the shape boundary of the controlled and integrated habitat service system into cells.
3. Add spatial data.
   A. Identify material habitat service operations and prioritize functional placement on a circular layer.
   B. Positional location of resources into assemblages of operational material mechanics (constructions, infrastructure) used as a service by accessing users.
   C. Transportational motion of resources, assemblages of resources, and humans between cells.
4. Add decisional data.
   A. Identify parallel decisional process operations and prioritize decisions on a conceptual (meaning-mental) layer.
   B. Position location of information into assemblages of operational information mechanics (instructions and software) used by a service by accessing users.
   C. Transportational motion of inputs, outputs, and decisions between information service boundaries.
5. Add conceptual (intentional) data.
   A. Survey human requirements.
   B. Survey natural landscape artifacts.
   C. Survey planning of constructable city.
6. Develop a simulated map of the geoinformatic environment.
   A. Develop resource physical simulation model.
   B. Develop user access opportunity distribution model.
   C. Develop world terrain model.
   D. Develop building (infrastructural) models.
   E. Develop network sharing model.
   F. Develop unified information coordination model.

10.2 Material visualization and analysis

In a unified real-world societal system, coordination must exist between the following three [material visualization and analysis] data sets in order for decisioning to be optimal:

1. **Building information modeling (BIM)** - describes information about the design and construction of building sites. digital models of real world assets. BIM tools often use local coordinate systems.
   A. Asset design processes.
   B. Asset documentation processes.
2. **Geographic information system (GIS)** - describes information about the material environment. A system designed to capture, store, manipulate, analyze, coordinate, and present all types of geographically referenced data (a.k.a., earth referenced data, spatially referenced data). GIS merges cartography, statistical analysis, and databased technology. In a GIS framework, both spatial and non-spatial databases are combined into a geodatabase. A GIS essentially creates map layers of specific thematic maps. By layering the information one on top of the other, an information system can show, for example, the relationship and degree of connectivity between various land uses and transportation routes in a region. GIS extends Building Information Modeling (BIM) design data through visualization and analysis of structures in the context of the material (natural and built environment). GIS data usually rely on geographic coordinate systems that precisely locate the data in the real world. Environmental design constraints are often stored in a GIS database (e.g., what is the terrain[ability] and existing building in some geographic [spatial] area.
   • Geographic reference processes
3. **Operational information system** (OIS; a.k.a., operational material system, OMS; a.k.a., facilities management, infrastructural development and operations)
   A. Operational systems processes.
4. **Unified interface** that integrates the prior three into a personal dashboard to see in a common environment. The ability to pull together huge amounts of information and visualize it for users and the support of their decisions.

BIM and GIS overlap when their data concerns

information about infrastructure, building sites, floors, and rooms. Because of the overlap, the integration of data from both domains (often considered separate disciplines in the professional market) is required for integrated city modelling, development, and operations. BIM model data becomes spatially located in GIS data, and GIS data can provides the materialized context for BIM data.

Historically, although BIM and GIS have a common interest in modelling material object types, they differ in their encoding, their use of geometry and semantics, as well as their level of detail.

11 [Engineering] Geospatial information system (GIS)

A.k.a., Geographic information system, geographical information technology (GIT), geoinformation and environmental planning, geomapping, geovisualization, earth built visualization, world information system.

GIS is an integrated system of spatial and conceptual information about the real-world that has been abstracted and simplified into a digital database upon which analysis can be conducted to facilitate the production of solutions to spatial problems using maps and other geographic information. GIS is a geo-spatial computing interface, wherein, geo = earth (or world) and spatial = 3D (+ time). A geo-spatial information system (GIS) is a description of the environment (e.g., world, earth, spaceship, etc.) past and present. A geographic information system (GIS) is a computer system for capturing, storing, checking, and displaying data related to positions on Earth's surface. By relating initially unorganized data, GIS facilitates a better understand spatial patterns and relationships.

> **STATEMENT:** *Location is a critical piece of information in order to address societal-level problems. Planning a material environment concerns problems and data that are inherently spatial.*

GIS has three primary purposes:

1. The collection and storage of spatially related information.
2. The mapping of data onto a geospatial (world) environment.
3. The analysis of the geospatially mapped world.
4. The visualization of the geospatially mapped world (for user observation).

In this context, a 'map' is a scaled model [data-set] visualization of a real-world reality. Maps convey useful information for navigating within a real-world, spatial reality. Mapping, for example:

1. CAD (mechanical drawings) and BIM (mechanical asset data) are map data that may be added as sources in a GIS world scene to become a geo (world) multi-layered data set that accounts for human constructed assets (objects in the scene). These assets (CAD & BIM) become placed (located, positioned) in a geo-spatial scene.
2. Thermal imaging monitoring data from multiple sensors can be combined to form a continuous thermal image map of the whole environment. This image map can overlay all other real-world representational maps.

11.1 The components of a GIS

> **NOTE:** *Information systems exist as the conjunction of [specialized] software and computer hardware.*

The elements (or, components) of a geospatial information system (GIS) are:

1. **Database** - where map information is stored. Sometimes called a geodatabase (GDB).
2. **Software** - that which runs (operates/processes information, computes) the database.
3. **Hardware** - the physical machine that runs the software, computing data and code.
4. **Network** - the physical and digital aggregation of the whole information system.
5. **Coordination procedures** - how data is collected and moved.
6. **Analysis procedures** - how data is analyzed.
7. **People** - the users who benefit by having problems resolved [more easily] from the usage of the geospatial information system.

11.2 GIS data input

Note that data can be brought directly into the GIS software from real-world surveying sensors, a 'point cloud' being one such example. In other words, laser sensors survey the real world to produce a 'point cloud' data set [of the current, real-world], and that data set is imported ("brought into") the BIM-GIS software. It may be brought.

11.3 GIS Software

In concern to commercial software, the software "*ArcGIS*" is one of the more well-known and widely-used GIS software products. "ArcGIS" software is produced by "Esri Corporation".

> **NOTE:** *What does the Arc in ArcGIS stand for? The Esri Community Forum has a thread on "The meaning of Arc." [community.esri.com]*

11.4 Geospatial relationship types

An information technology for referencing and data storage of spatial analyses. Geospatial relationships can be modeled between the feature classes, enabling more advanced GIS analysis. The more common types of geospatial relationship data structures in the geodatabase are:

1. **Topology** - Defines and controls data integrity rules for features. For example, there should be no gaps between polygons. It supports topological relationship queries and navigation, such as feature adjacency or connectivity and synthetic feature editing tools, and allows feature construction from unstructured geometry (for example, constructing polygon features from line features).
- **Geometric network** - Consists of a set of connected edges and junctions (line and point features) that, along with connectivity rules, are used to represent and model the behavior of a common network infrastructure in the real world. Examples of resource flows that can be modeled and analyzed using a geometric network include, but are not limited to: water purification nodes and distribution conduits, power generation points and electrical lines, gas storage points and gas conduits, telecom points and lines, and water flow in a stream.
  A. For example, the streets in a streets feature class could be categorized into three subtypes, each with slightly different geometric properties and affects: local streets, collector streets, and arterial streets.
2. **Network dataset** - consists of a set of connected edges and junctions, as well as turn features, along with connectivity rules, that represent and model the behavior of a network systems. Examples of undirected network flows that can be modeled with a network dataset include, but are not limited to: transportation pathways and conduits in and between cities; electrical power and telecommunications pathways within and between cities.
3. **Terrain** - a data structure that is generated from a mass collection of elevation measurement points, typically from remote-sensing data sources. It is a triangulated irregular network (TIN)-based data structure with multiple levels of resolution and is used to represent surface morphology. A terrain is used for 3D surface modeling applications.
4. **World surface** (cadastral fabric, polygon fabric, parcel fabric) - a continuous surface of connected polygonal shapes ("parcel features", "parcel polygons") that represents the record of survey for an area of land surface (a world space). This data structure enables GIS data to be integrated with survey data to maintain a consistent and accurate survey record. Spatial accuracy in the world polygonal surface ("fabric") is required.

Additional business logic in the geodatabase, in the form of subtypes and attribute domains, can also be applied to GIS data. This additional layer of influential abstraction is likely to overlay in such a way that it obscures the fulfillment of all individual humans in the real world. For example, three jurisdictions in a market-State could be categorized into three boundary sub-

types: legal jurisdiction 1, legal jurisdiction 2, and legal jurisdiction 3; or, market 1, market 2, market 3; influence zone 1, influence zone 2, influence zone 3.

Table 15. Engineering Approach > GIS*: Table shows datasets in a standard geodatabase.*

GIS Data	Geodatabase Dataset
Coverage	Feature dataset containing feature classes
Shapefile	Feature classes
Raster data (e.g., satellite images, air photos, scanned maps, digital pictures, etc.)	Raster dataset and/or raster catalogue
CAD data	Feature dataset containing feature classes
Surface modeling or 3D data	Terrain
Service (utility) network data (e.g., life, technical, facility; water, telecomm., energy, etc.)	Geometric network
GPS coordinates	Table of x,y coordinates that can be generated into a feature class.
Survey measurements	Cadastral fabric

12 [Engineering] Building information modeling (BIM)

A.k.a., Building information management (BIM), architecture, construction engineering.

The 'B' in BIM stands for 'Building, the verb' not 'Building, the noun.' Building information modeling (BIM) is a process supported by various tools and technologies involving the generation and coordination of digital representations of physical and functional characteristics of places. BIM is the creation of a 3 dimensional virtual model that hat contains all of the relevant project information (from one or more sources) about an object (created as a project). Today, the definition of BIM has broadened into an approach that integrates the previously isolated functions and work-flows of geographic and building information. BIM is characterized by information about real-world building objects. BIM in general is concerned with data collection, modification, and application in the context of building geometry/shapes. A BIM object is an object (geometry, assembly) that has data associated with its sub-object geometry. A BIM object has relevant data associated with all of its geometry in order to visualize, standardize, tabularize and calculate quickly. BIM is a digital representation of a building's physical and functional characteristics. BIM workflows and software are also necessary to appropriately prevent clashing of systems during construction and operation. BIM clash detection software ensures that every object in the system fits correctly and does not overlap or clash/collide in any detectable way, and does so automatically in the software. When something doesn't fit, the software will inform the designer/decider of the conflict.

> **NOTE:** *There is also the idea of socio-technical OIM (object information modeling); every object in the habitat system has an OIM associated with its geometry; herein, BIM as building-object information modeling would be a sub-classification of OIM.*

BIM should ensure that right information about real world working objects is available to the right person at the right time. BIM integrates all project [world] asset data about real world (or, potentially real world) objects.

BIM is based on the idea of a continuous use of a 3d digital CAD building or infrastructure model over the entire life cycle of an engineering construction and operation process - from the design, through the planning and execution, to the operation and decommissioning of the project.

In a sense, building information modeling (building information coordination/management) comes down to how the "building" information is being used.

BIM is a process, a workflow:

1. BIM design model.
2. BIM construction model.
3. BIM operations model.

NOTE: *BIM-type frameworks allow any change to a design (or operation) to be traceable.*

As a conception, 'Building Information Modeling' (BIM) is a digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle; defined as existing from earliest conception to demolition. Regardless, in common application, BIM is a very broad term that describes the process of creating and managing digital information about a built asset such as building, bridge, highway, tunnels, transport, communications, and cities, and so on.

Information modeling is a modern early 21st century computer technique to associate information (concepts with meaning) with mechanisms (shapes with objects). The benefits of information modeling (using 3D) include:

1. Improved understanding and collaboration through ease of visualization in 3D.
2. Data accuracy through tabular information input associations with objects (3D, shapes, geometry).
3. Clash detection through software that detects clashes and conflicts within a project, reducing future errors. Note here that the consolidated clash-free model is also known as a: conformed model, confederated model, clash-free software encoded model, conflict-free model, etc..
4. Efficient documentation through software that intuitively streamlines the construction of documents.

The common BIM execution plans include:

1. Contact info.
2. Software version.
3. Hardware specifications.
4. Information exchange.
5. BIM uses.
6. BIM goals.
7. Standards (and file formats, naming convention, methods of sharing, model protocols).

BIM relevant project plan delivery phase life-cycle:

1. Pre-design.
2. Design.
3. Documentation.
4. Construction.
5. Operation.

Viewing and tracking team work upon assets includes (in the BIM style):

1. Software (asset and building coordination):
   A. Autodesk ecology example - Revit, BIM 360, BIM 360 Team (tracks team changes), BIM 360 Glue, Fusion.
   B. Power BI - data analytics for a project that produces dashboard analytics of a project.
   C. Dynamo is an open source visual programming platform supported by Autodesk and available as a free Revit plugin. Dynamo is shipped along with Revit since version 2016 R2, but you can always download the latest version from [dynamobim.org].

Today, BIM means different things to different producers and users of the "BIM" framework. Generally, BIM refers to the coordination of data, drawings, design, construction, and operation of assets (in general) and building-related assets (in particular).

How does sharing occur?

1. Central location - where people upload, view, and share content.
2. Cloud drive - someone uploads, another downloads; the server stores the file, user downloads to view or can view online, and possibly even, markup online.
   A. Google drive, Dropbox.
3. Single file revision - everyone works together on a single file that is stored on aa network.
   A. BIM 360 Team, Revit server (if you have your own network).

CLARIFICATION: *Although BIM contains the word building, the BIM framework can be applied to any real world asset or system of assets, such as a city, within which some of the assets are buildings.*

Future BIM naming integrations are likely to include:

1. AI SIC = Structural Information Computation.
2. AI Society = information task potential for automation. Allows for optimized information coordination.
3. AI BIM = HSS task potential for automation. Allows for optimized 3D coordination. Allows optimized structuring. Daylight optimization for all structures in a city, and with all relevant available data. Allows optimized documentation.
4. Robotics may assemble buildings in the system, for those cities who value that level of technical efficiency.
5. A computer is programmed with a set of rules, and in a socio-decisioning structure, some of these rules are parallel inquiry thresholds, which the computer is given specific instructions not to

exceed for some explicit condition.

There are work-oriented views that BIM software can broadly express:

1. Design BIM.
2. Constructions BIM.
3. Operations BIM.
4. Or, some combination of BIM.

12.1 Building information models (BIM)

Building information models (BIMs, the noun) are data stored in files (often, but not always in proprietary formats and containing proprietary data) that can be read, shared, or networked to support decisioning regarding building assets. BIM software is used to plan, design, construct, operate and maintain most physical infrastructure (e.g., cities, buildings, bridges, tunnels, railways, water storage and transportation, etc.)

12.1.1 BIM use case scenario

The following services are active for the building in both BIM case 1 and BIM case 2.

1. Habitat power sub-system - An amount of electrical power was required to be produced (energized) to operate this service.
   A. Frequency and coherency.
   B. Power processing service.
2. Production socio-technical sub-system - An amount of material mass was required to be produced (manufactured) to operate this service.
   A. Shape and composition.
   B. Production processing service.
3. Data socio-technical sub-system - An amount of information data was required to be produced (calculated) to operate this service.
   A. Logic and computation.
   B. Information processing service.
4. Transport socio-technical sub-system - An amount of mass re-positional movement was required to operate this service.
   A. Transport and coordinates position.
   B. Transport processing service.

12.1.2 BIM asset modeling

Assets may be organized into the material [requirements] categories of:

1. Soft (computational/informational) interface material ("ware").
2. Hard (physical/informational) interface material ("ware").
3. Training - functional materials.
4. Sensors - collect data associated with physical and functional characteristics.

This contribution framework gives every user an understandable view of how to develop socio-technical competence, and every InterSystem Team member (as someone with socio-technical competence) an operational, execution view of the system. When an element is a model is changed, every view is updated, with the new change appearing in simulation, conception, section, elevation, and sheet views.

Assets may be organized into the material environment

1. Standards - societal design specifications.
2. Guides - guides, best practices, agreements.
3. Work-flows (because it's 'team' view, not command 'disciplinary' view).
4. Modeling quantity.
5. Metrics.
6. Constructs.

Assets (BIM assets) may be visualized together as deliverables with different 'shape' metadata, including at the high level:

1. 3D visualizations (rendering).
2. Coordinated DWGs (unified file system of accurately shaped objects and their metadata, revisioned).
3. Basic quantities (measurement data system, described in detail in the material system specification).

Assets (BIM assets) may be visualized together as information process deliverables, including but not limited to:

1. Thermal studies.
2. Illumination analysis.
3. Structural analysis.
4. Compositional analysis.
5. Constructability.
6. Pre- and fabrication.
7. Asset tracking (on 'physical' data side; and, issue tracking on 'informational' data side).
8. Global information system tracking all assets through integrated, informational and positional, coordinate system (e.g., BIM/GIS overlay of a city collaborated upon by a local and global InterSystem team living in the network of integrated city service systems).
9. Photogrammetry.
10. Field BIM.
11. Protocol.

Once we have a project plan we can start talking about how we share and collaborate among the individuals contributing to this common organization.

12.1.3 Asset model storage

A digital representation of an asset needs to be accessible quickly in a distributed environment that can be updated and upgraded to adjust to more complex query, analysis, and inspection over time and across the lifespan of the asset.

12.1.4 BIM in Application

A highly simplified application of BIM must involve at least the following design-operational elements:

1. **Architectural BIM** - material, dimensional composition of [a commonly standard] space, forming an ID (e.g., building ID, floor ID, room ID, room dimensions, spatial plan and simulation.
2. **Mechanical BIM** - technical equipment IDs (and standards) of one of the HSS technical sub-systems (production, transportation, calculation, etc.; e.g., mechanical equipment).
3. **Electrical BIM** - electrical equipment IDs (and standards).
4. **Pressurizing BIM** - Pressurizing equipment IDs (and standards).
5. **Hydraulics BIM (a.k.a., plumbing BIM)** - Hydraulics equipment IDs (and standards).
6. **Atmospheric BIM** - atmospheric equipment IDs (and standards).
7. **Incident response (a.k.a., protection)** - Incident protection equipment IDs (and standards).

12.1.5 BIM Project lifecycle phases

The BIM project life-cylce phases are:

1. **Planning** (Plan project):
   A. Knowledge management tools.
   B. Requirements analysis .
2. **Design phase** (D, Project design):
   A. D1 - conceptualization, resource information flow programming, and for market, cost planning.
   B. D2 - architectural, structural, and systems design.
   C. D3 - analysis, detailing, coordination and specification.
   D. System design deliverables:
      1. Visualization.
      2. Analysis.
3. **Construction phase** (C, Project assembly):
   A. C1 - construction planning and construction detailing.
   B. C2 - construction production, manufacturing, and allocation, and in the market, procurement.
   C. C3 - as-built and handover/integration, and in the market, commissioning.
   D. System construction deliverables:
      1. On-site construction.
      2. Off-site construction (and transport).
      3. Procurement and deliver (and inspection).
4. **Operation phase** (O, Project becomes 'service' in 'operation'; Asset & Facility Management):
   A. O1 - occupancy and operations.
   B. O2 - service continuation, and maintenance.
   C. O3 - decommissioning and major re-programming.
   D. System operation deliverables:
      1. InterSystem planning and operations scheduled coordination.
      2. Team and system task operations.
      3. Simulation of operations (from information flow to materialization).
      4. Incident response operations.
      5. Resource operations.
      6. Information operations.
      7. Re-use/cycle operations.
5. **Social feed back deliverables** (a.k.a., model-based optimizations to enable process optimization across project lifecycle phases):
   A. Resource survey.
   B. Materials coordination.
   C. Knowledge visualization.
   D. Algorithmic value-transparent decisioning.

12.1.6 BIM Design phase

Preparatory steps in the design phase:

1. The design phase results in a design InterSystem team role of intent model consisting of discipline BIMs (i.e., Habitat service systems, HSS'). To optimize stability, each stakeholder must have defined access [privileges] to the HSS core source algorithm and information system interface. This infrastructure allows each stakeholder to update data through the HSS that is specific to their InterSystem Team accountability role (or "discipline"). For example, the architect InterSystem team (as the InterSystem engineering stakeholders) only can make space-related updates from the decision model to the extant, material habitat service system (HSS). This "as necessary" access diminishes the possibility of one stakeholder overwriting another's decision (discipline) data, and maintains the integrity of model data transferred to the IWMS.
2. Information exchange during the design phase: Requirements deliverable
3. Requirements specify existing space and equipment standards and desired nomenclature for the new building. For spaces, an owner's

requirements may specify the space classification nomenclature (e.g., OmniClass) unique building and floor IDs/names, room numbers/names; and room standards. A room standard is a collection of properties that define a space. Room-standard properties may include room use, room type, cost/unit area, maximum room area and maximum occupancy.

4. For equipment, an owner's requirements may specify equipment classification nomenclature, equipment IDs and equipment standards. An equipment standard is a collection of properties that define a piece of equipment. Such properties may include equipment category, description, manufacturer, dimensions, model number and power specifications, among others. With the allocation of equipment (i.e., assets, service objects) come equipment usage protocols (standards) as predefined ways and accountabilities of using common heritage resources.
5. If the design includes spaces undefined by existing space standards, IWMS or design-side stakeholders can create new and/or modify existing decision protocols (e.g., space standards).
6. The InterSystem team members create these new instructional standards via the engineering interface of the unified societal information system or the societal information systems model (a visual interface) to the unified societal information system. For example, an architect of sufficient accountability on an architectural InterSystem Team can define a new room standard for a specialized lab and apply this room standard to all poly-lined spaces that will serve as specialized labs. This eliminates the need for tedious and error-prone manual entry of properties for spaces with similar purposes. As a result, the InterSystem Teams can immediately access updated spatial information, which is synchronized from a model into the unified societal system; then use it to accelerate access occupancy and service programs. Information exchange during the construction phase: Design specification
7. The team "lead" decision coordinator (a.k.a., construction BIM manager) communicates to InterSystem team members (e.g., mechanical team, electrical team, etc.) the availability of tasks, which are selected by team members; when selected the team member acquires a set of operational requirements (i.e., responsibilities) for information that is entered, reviewed, and/or executed within and/or upon the societal information system. These InterSystem teams coordinate accountability for all materializations (digital and material) for their respective Habitat InterSystem Team (i.e., "trade") role, position, and tasking.
8. Once the space design, floor plans and equipment information (IDs, locations, equipment standards) are available in the societal information system, engineering construction (installation) teams update the habitat service system.

13 [Engineering] BIM and GIS integration

NOTE: *Even better than BIM-GIS conversion is BIM-GIS integration; even better than a BIM-GIS integration is a unified societal system. Wherein, CAD-GIS conversion - CAD data are validated and then converted into GIS data. BIM-GIS conversion - BIM data are validated and then converted into GIS data.*

Together, the BIM and GIS workflows (data sets and calculations) exist to design and express what is happening in the continuous now of a material service system operation (i.e., the global HSS). BIM and GIS data become integrated into an actual and potential (potentially actualizing) global societal-information city network. Projects are delivered through structured design and documentation that produces safely and optimally decided constructions that provide services to humanity, which are monitored and controlled by all contributing humans. That which may be built safely into the material environment (BIM data) becomes spatially positioned and sensed in a geospatial environment (GIS data). The simulation of what is and what could be, at both the conceptual-information and material-realization levels. The four axiomatic information realization components of:

1. Information.
2. Conception.
3. Spatial.
4. Iteration.

Together, all is information, wherein conception [by consciousness] allows for understanding information, the spatial conception is the 3D materialization of information [into plannable experiences], and the iteration conception is the 4D temporalization of information [into human individual, conscious experience].

Building information modeling (BIM) is a process involving the generation and computation of digital representations of physical and functional characteristics of places (i.e., of objects). BIM is a virtual representation of a construction project. It is an integrated process that uses intelligent, digital information to explore, develop and test physical and functional characteristics of a project before it is built. In short, it is a highly detailed representation of a an environment, such as, a building.

BIM represents a series of parametric objects that composed together to form a building model which carries all information includes their physical and functional characteristics and project life cycle information. Since BIM represent the detailed geometrical and semantic information of the building, the application of GIS is needed to manage the construction project's information resources in a material, positional space. GIS can use information from many different sources, in many different formats and can link data sets together by common locational data such as coordinate or postal zip code. Besides, GIS can use combinations of data sets to build and analyze integrated information and also can convert the existing digital information into a form that meets the user's need. From this point of view, GIS can complement BIM function in order to develop a systematic platform for construction purpose. Finding of this study, there are some drawbacks in this technique especially in the construction application in term of data sharing, data integration and data management. Furthermore, the integration of GIS in BIM is studied and potential techniques are shown to overcome the drawbacks of the construction application.

1. Integration of BIM data into a 2D/3D GIS database.
2. Data exchange between BIM and GIS.
3. Integration of BIM construction data.
4. Integration of existing geospatial survey data.

The models created in modern BIM design processes are sophisticated enough to simulate construction to find defects early in design and to generate highly accurate estimates of resource requirements throughout dynamically changing projects.

A societal information and decision support system can now handle billions of events a second from live sensors, serve visualizations from petabytes of 3D model content and imagery to a browser or mobile phone, and perform complex predictive analysis scaled over multiple dispersed processing nodes.

The 3D model generated during BIM design processes is a record of a change to a physical asset. Visualization can be part of the process in that it helps humans understand the dynamics, characteristics, and aesthetics of a proposed design.

All habitat (city) infrastructure is BIM (e.g., the domains of rail, roads and highways, utilities, and telecommunications). When information is 'built' into the environment, it is generally referred to as 'infrastructure'.

Any agency or organization that manages and builds fixed physical assets has a vested interest in making sure that their design and engineering contractors use BIM processes.

Note that BIM data can be used in operational workflows for operational coordination and control.

When seen as a process, integrating GIS technology with BIM becomes vastly more complex than just reading graphics and attributes from a 3D model and showing them in GIS.

How do users need to use a wide range of project data in geospatial context. We also find that focus on the model sometimes means we've overlooked simpler, more basic workflows that are essential to the whole story, such as using accurately collected field data on a constructions site to link location and model data for inspection, inventory, and survey.

INSIGHT: *All information is pattern. Because*

all information is pattern, all materialization is pattern. All patterns of materialization, because they are only information, can be changed by a capable consciousness who understands the patterns of creation.

13.1 Spatial solution visualization resolution

In the study to visualize space re-creation, in time together, as a population, there is a requirement for coordination of and between:

1. Plans (design files):
   A. Architectural (Arch).
      1. Sketch design (SD).
      2. Design development (DD).
      3. Construction documentation (CD).
      4. Mechanical (MEP).
   B. Structural (Struct).
   C. Site (Site).
   D. Operation (Ops).
   E. Incident (OpsInc).
   F. Cycling (Cyc).
   G. Market-State (no acronym).
2. Drawings (drawing sheets):
   A. Mechanical.
   B. Structural.
   C. …
3. Project files:
   A. Images/photos.
   B. Models (level, dimension).
   C. Renderings.
   D. Simulation.
   E. Descriptions.
   F. Explanations.
   G. Specifications.

**Folders and sub-folders have permission to control which role, individual, or team (organization/business) can access this data. These permissions are cascading so that a sub-folder can have its permissions controlled by a higher-level folder. Roles and permissions for sub-folders may be inherited from the parent folder. A role(s) must be selected for that folder, then the role(s) are assigned permissions. Users are able to subscribe to a folder to receive notifications when new documents are uploaded, sheets are published, or content changes. This ensures the user is kept up-to-date with the data in the folders that is important to them.*

To coordinate this information for useful purpose is to facilitate the determined resolution of resource, machine allocation, and human contribution.

13.2 Spatial-informational mapping

Spatial information can be mapped to a semantic reference system. Thematic entities and relationships may be modeled as first class objects, linked to spatial entities through the variables:

1. Located_at (where an object is in relation to others).
2. Occurred_at (motion of object in relation to others).

These relationships from the upper-level ontology for any logistical access system. For example, a technician is associated with a control ability using a set of relationships (Technician—member_of—PowerService_Team—controls_at—GenerationStation—located_at—Spatial_Entities). Spatial information mapping allows for safe and coherent decisioning among a population. Without mapping objects (that with shape) to meaningful processes (motions of two or more objects), decisioning is likely to be significantly reduced in certainty.

13.3 Unified visual software solution

Visual software is the ability to interactivity work with information (spatial and conceptual) in a 3D environment. A unified BIM-GIS, unified, software platform will [seamlessly] integrate these two modalities; using a unified societal platform, these two design and development modalities will themselves integrate with an operations [dashboard, "facilities management"] platform.

There are two software modalities of BIM-GIS integration:

1. BIM as the construction design application (3D object constructing):
   A. Primary objective of software: Design 3D object.
   B. Secondary objective of software: 3D object data layering and analytics. Here, the reference coordinate is the 3 dimensions of space and 1 dimension of time (i.e., the real-world physics of 3D objects).
      1. Object may or may not be animated.
      2. Object may simulate real-world physics and interactions to study object.
   C. Output: A single object mesh with metadata layers. The software can execute and display engineering design instructions. A BIM platform organizes all information relevant to an constructible object.
      1. Data format output standard, example: CAD, CAM, CAE, building models, 3D meshes, etc.
   D. Users design assets: Users design an asset and simulate real-world physics upon the asset to study the asset.
      1. Users create and study individual assets. Users view attribute information for objects.

2. GIS as the in-place constructed visualization application (3D object positioning):
   A. Primary objective of software: Place multiple 3D objects on terrain.
   B. Secondary objective of software: 3D objects data layering and analytics. Here, the reference coordinate is the terrain of the Earth (i.e., the real-world terrain of the planet; spatial).
      1. World may or may not be animated.
      2. World may simulate real-world physics and interactions to study world.
   C. Output: A unified visualization with multiple object meshes associated with real-world coordinates (or potential, real-world coordinates) and world associated metadata layers. The software can execute and display ("dashboard") 3D geoanalytics. A GIS platform organizes all information relevant to objects existing in a world (or, the real world).
      1. Data format output standard, example: I3S OGS, IFC files, world simulation engines (i.e., game engines), etc.
   D. Users design worlds: Users place assets together in a world and simulate real-world physics and organismal interaction to study the world.
      1. Users create and study worlds, which are composed of some world reference frame (e.g., the earth) and individual assets. Users view attribute information for world. Note that some of those assets will previously exist in the real-world, such as rivers on earth, and other assets will be created by humans, such as buildings and bridges.

13.4 Open GIS and BIM standards

Industry foundation classes (IFC) and City Geographic Markup Language (CityGML) are two standards which have been developed independently. Although IFC and CityGML both deal with object geometry, surface materials/appearances, semantics, and their inter-relationships, the information models are different as they are adapted to the specific requirements of the domains from which they originate. An example of a major difference between the models is how the IFC schema is described using the modeling language EXPRESS, which follows the entity relationship modeling paradigm—whereas the CityGML schema is defined using the Unified Modeling Language (UML) and, therefore, follows the object-oriented modeling paradigm. Although both IFC and CityGML are object-oriented, each uses a different formal modeling language.

The semantic model of IFC, in its current version "IFC4 Addendum 2", focuses on buildings and alignments as well as the physical elements of the building construction, such as slabs and beams—whereas CityGML models all major observable natural and man-made entities in a city or landscape, including buildings. To represent entities with their geometric and semantic properties in different granularities, CityGML includes five well-defined levels of detail (LOD0–LOD4). Regarding IFC, a building element might have multiple geometric representations. (Hijazi, 2018)

Additionally, a "Level of Development" concept was introduced by Forum B which, according to Geiger et al. (2015), cannot be directly compared with the CityGML's level of detail. Level of development (LoD) is applied in BIM to reflect the progressions of the modelling geographic representation, from the lowest LoD of general 2D, to the highest LoD of BIM involving 3D models and corresponding detailed non-geometric information. (Hijazi, 2018)

The main problem in the integration of BIMs with geospatial information occurs at the point of transferring the geometric information. Building models use representations such as constructive solid geometry (CSG) and sweep geometry mostly in local coordinate reference systems, while geospatial models mainly use boundary representation (BRep) in global coordinate reference systems. The fundamental difference arises from their distinct modeling paradigms, which are due to the way 3D models are acquired in the GIS domain in the field of BIM and computer-aided architectural design (CAAD). Using GIS, 3D objects are derived from surface observations of topographic features based on sensor-specific extraction procedures. Features are then described by their observable surfaces by applying an accumulative modelling principle [25]. Alternatively, BIM models reflect how a 3D object is constructed. They follow a generative modeling approach and focus on the built environment, rather than on topography. Therefore, BIM models are typically composed of volumetric and parametric primitives representing the structural components of buildings. However, the relation between the two semantic models (IFC and CityGML) for BIM (design model) and geospatial models (real-world model) has been researched to develop common unified spatial applications with minimum conversion overhead. (Hijazi, 2018)

13.4.1 BIM open standard

- The Industry Foundation Classes (IFC) for the BIM domain [ISO, 2013; Building SMART International, 2013].
  - The Industry Foundation Classes (IFC)3 standard is an open data model used in the Building-information modelling (BIM) domain for the exchange of construction models, often including 3D models of buildings. It has also been adapted as the ISO 16739 international standard [ISO, 2013]. Its geometric aspects are however mostly defined or derived from a different standard,

ISO 10303 [ISO, 2014], which also specifies the STEP Physical File (SPF) encoding that is most commonly used in IFC files (.ifc). The IFC standard provides dedicated entities and attributes for geo-referencing models. IFC files can contain many types of classes: 130 defined types, 207 enumeration types, 60 select types, 776 entities, 47 functions, and 2 rules. The geometries in them can use several different representation paradigms which can be combined more or less freely.

13.4.2 GIS open standard

Open Geospatial Consortium (OGC) standards depend on a generalized architecture captured in a set of documents collectively called the Abstract Specification, which describes a basic data model for representing geographic features. Atop the Abstract Specification members have developed and continue to develop a growing number of specifications, or standards to serve specific needs for interoperable location and geospatial technology, including GIS.

The OGC standards baseline comprises more than 30 standards; including, but not limited to:

1. I3S Open Geospatial Community (OGC) Standard.
2. GML (Geography markup language; XML-format for geographical information).
3. SPS – Sensor Planning Service.
4. SensorML – Sensor Model Language.

The OGC standard CityGML for the GIS domain [Open Geospatial Consortium, 2012].

1. CityGML [Open Geospatial Consortium, 2012] is the most prominent standard to store and exchange 3D city models with semantics in the GIS domain. It presents a structured way to describe the geometry and semantics of topographic features such as buildings and roads. CityGML as a data format is implemented as an application schema for the Geography Markup Language (GML).
   A. CityGML supports 5 levels of detail (LODs):
      1. LOD0 is non-volumetric and is an horizontal footprint and/or roof surface representation for buildings;
      2. LOD1 is a block-shaped model of a building (with an horizontal roof);
      3. LOD2 adds a generalised roof and installations such as balconies;
      4. LOD3 adds, among others, windows, doors, and a full architectural exterior;
      5. LOD4 models the interior of the building, potentially with pieces of furniture
      6. (CityGML does not mandate which indoor features need to be modelled, in practice resulting in models with a different granularity [Goetz, 2013; Boeters et al., 2015]).

13.4.3 BIM technical standards naming

Sources for BIM technical naming standards include:

- *BIM Technical Standards: Naming*. (2019). U.S. General Service Administration (GSA). [gsa.gov]

GSA BIM project-based file platforms use a four part file name consisting of:

1. The GSA assigned building ID or site ID.
2. 1 character major discipline / trade designator (from the PBS [gsa.gov]).
3. 1 character minor discipline / trade designator (from the PBS [gsa.gov]).
4. 5 characters to define contained floors.
5. 1 character type designator M/S/C (model, sheets, combined).

Thus, the file naming standard for a GSA BIM file type includes:

- Building Number_Major/Minor Disp_Included Floors_Sheet/Model File + .File Extension
  - For example: IL023ZZ_AC_B4-20_M.rvt

13.4.3.1 Potential architectural-engineering file naming convention

NOTE: *A similar file naming convention can be used for machine-engineering (as opposed to architecture, which is presented in the examples below).*

The following is a basic useful file naming convention:

- (Project Name)-(*architecture-sheet*)-(Name of Habitat)-(Building Identifier)-(Plan View)-(Version)-(Revision)
- For example,
  - auravana-architecture-sheet-AuraCurve-...
  - auravana-architecture-sheet-AuraKraho-Hex392-FL1-Electrical-V039-R182

Buildings are first identified as either primarily bio-construction (Bio) or mineral (Min), and then, the specific building identifier is added, for example:

- ...-BioHex392-...
  - This is primarily a bio-constructed building and the specific building identifier is Hex392.
- ...MinSky284A-...
  - This is primarily a mineral constructed building and the specific building identifier is Sky285A.

Type of architectural engineering plan views include, but may not be limited to:

1. Floor plan view (FL).
   A. E.g., FL1 OR FL2.
2. Roof plan view (RoofPlan).
3. Ceiling plan view (CeilingPlan).
4. Section view (CutView).
   A. E.g., CutView AA OR CutView BB.
5. Elevation (facade) view (Elevation) OR (SideView) with reference to compass directions E, N, S, W, or by degree.
   A. E.g., Elevation-E or Elevation-SE.
   B. E.g., SideView-NW.
6. Site Plan view (SitePlan).
7. Network Plan view (NetPlan).
8. Multiple Views present (MultiPlan).
9. Areas View (AreaPlan).
   A. E.g., Area3B or AreaCC.
10. Openings View (OpeningsPlan).
11. 3D view (3DView).
12. Orthographic view (Ortho...).
    A. E.g., OrthoTop.
    B. E.g., OrthoRight or OrthoN.

Plan views can be mixed, for example,

- ...-SitePlan-OrthoTop-...

Type of architectural-engineering plans include, but may not be limited to:

1. Enclosure.
2. Structure.
   A. Above ground.
   B. Foundation (below ground).
3. Water.
4. Electrical.
5. Informatics.
6. Illumination.
7. Heating, ventilation and air-conditioning (HVAC).
8. Sensory, including: thermal, acoustic, air motion, etc.
9. Furniture.
10. Landscape.
11. Cultivation.

During development, plans are versioned and revised, wherein a new version is a adaptation (of an existing plan) and a revision is an internal change (to an existing plan):

1. Version: V0##
2. Revision: R0##

If there is a letter after a revision number, then it is just a different view (plan) of the same revision. Always starts with A and proceeds alphabetically:

- For example, V001-R003-A, V001-R003-B, V001-R003-C

13.4.3.2 Other sources for file naming conventions

Other sources for example naming conventions include:

- *BIM Guidelines*. (2017). Smithsonian Facilities.
  - https://www.wbdg.org/FFC/SI/SI_BIM_Guidelines_Oct2017v2.pdf
- *BIM Guidelines for Design and Construction (2015). Commonwealth of Massachusetts*. BIM Steering Committee.
  - https://www.mass.gov/files/documents/2017/12/19/bim-guide.pdf
- *BOE CADD Standards*. (2018). City of Los Angeles, Bureau of Engineering.
  - http://eng2.lacity.org/techdocs/CADSTDS/BOE_CADD_Manual_181107.pdf
- *CAD BIM Technology Center Resources.* US Army. Accessed: December, 2019.
  - https://cadbimcenter.erdc.dren.mil/
- *CAD and Image Standards for Construction Documentation.* (2009). Harvard University Planning Office.
  - http://home.planningoffice.harvard.edu/files/hppm/files/cad_standards.pdf
- *Capital Projects Group - CAD Naming Convention*. METROLINX. Accessed: December, 2019.
  - http://docplayer.net/docs-images/77/74592883/images/52-0.jpg
- *CADD/BIM Standards Manual*. (2018). Report No. CPG-DGN-PLN-084. Revision 1. METROLINX.
  - http://docplayer.net/74592883-Cadd-bim-standards-manual-cpg-dgn-pln-084-revision-1-01-31-2018.html
- *CAD Standards Guideline For Facility Documentation and Construction Projects*. Creighton University Planning and Design. Accessed: December, 2019.
  - http://docplayer.net/9068760-Cad-standards-guideline-for-facility-documentation-and-construction-projects.html
- *MIT Design Standards: BIM and CAD Drawing Standards v6.0, Thematic Folder*. (2016). MIT Infrastructure Business Operations, Facility Information Systems.
  - https://web.mit.edu/facilities/maps/MIT_CAD_BIM_guidelines.pdf
- *Naming Conventions (6.2)*. CAD Standards - 3rd Edition, November, 2016. Denver Water.
  - https://dev.simantics.org/index.php/Structural_Ontology

Scholarly references (cited in document)

- Hijazi, I., Donaubauer, A., Kolbe, T.H., (2018). *BIM-GIS Integration as Dedicated and Independent Course for Geoinformatics Students: Merits, Challenges, and Ways Forward*. International Journal of Geo-Information. ISPRS Int. J. Geo-Inf. 7(8), 319. https://doi.org/10.3390/ijgi7080319 | https://www.mdpi.com/2220-9964/7/8/319/htm

Scholarly references (non-cited)

- Griendling, K.A. (2011). *Architect: the architecture-based technology evaluation and capability tradeoff method*. Georgia Institute of Technology, Thesis In School of Aerospace Engineering. https://smartech.gatech.edu/handle/1853/42880
- Hor, Abdel-Hadi. (2015). *A semantic Web platform for BIM-GIS integration*. York University, Toronto Canada. https://doi.org/10.13140/RG.2.1.4176.6643 | https://www.researchgate.net/publication/305775768_A_semantic_Web_platform_for_BIM-GIS_integration
- Kineman, J.J., Srirama, K, Wilby, J., Mobus, G. (2017). *A Framework for Understanding and Achieving Sustainability of Complex Systems*. Systems Research and Behavioral Science, Wiley Blackwell, vol. 34(5), pages 544-552, September. https://ideas.repec.org/a/bla/srbeha/v34y2017i5p544-552.html
- Mobus, G. (2015). *A Systems Science Framework for Understanding the Nature of Governance*. ISSS Journal. Vol. 1, No. 1. http://journals.isss.org/index.php/proceedings59th/article/view/2497 | https://www.semanticscholar.org/paper/A-Systems-Science-Framework-for-Understanding-the-Mobus/c27fa5a62473abaf8102df2f896883c9ff726730
- Mobus, G. (2017). *A Framework for Understanding and Achieving Sustainability of Complex Systems*. System Research and Behavioral Science, Vol. 34, No. 5. https://doi.org/10.1002/sres.2482 | http://journals.isss.org/index.php/proceedings60th/article/download/2939/1011 | https://onlinelibrary.wiley.com/doi/abs/10.1002/sres.2482
- Preiss, O. (2004). *Foundations of systems and properties: methodological support for modeling properties of software-intensive systems*. Institut d'informatique fondamentale, Theses N 2013, Lausanne, EPFL. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.368.5776&rep=rep1&type=pdf
- *Transforming Systems Engineering through Model-Centric Engineering*. (2018). Report No. SERC-2017-TR-111. Stevens Institute of Technology, Systems Engineering Research Center. https://apps.dtic.mil/dtic/tr/fulltext/u2/1058354.pdf

Book references (cited in document)

- Geiger, A., Benner, J., Haefele Kark, H. (2015). *Generalization of 3D IFC Buildings Models*. In 3D Geoinformation Science; Breunig, M., Al-Doori, M., Butwilowski, E., Kuper, P.V., Benner, J., Haefele, K.H., Eds.; Springer International Publishing: Berlin, Germany, pp19–35.
- Maier, M.W. (2009). *The Art of Systems Architecting (Systems Engineering)*. CRC Press.
- *NASA Systems Engineering Handbook.* NASA/SP-2007-6105. NASA. 2007. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150015500.pdf

Book references (non-cited)

- Mobus, G., Kalton, E., Michael, C. (2015). *Principles of Systems Science.* Springer. ISBN: 978-1-4939-1920-8
- Sutcliffe, A. G. *Requirements Engineering*. The Encyclopedia of Human-Computer Interaction, 2nd Ed. https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/requirements-engineering
- *Technical risk assessment handbook*. Australian Government Department of Defense: Defense Science and Technology Organisation. 2010. https://www.dst.defence.gov.au/sites/default/files/basic_pages/documents/Technical-Risk-Assessment-Handbook_2.pdf
- Wasson, C. S., (2016). *System engineering analysis, design, and development: Concepts, principles, practices*. John Wiley & Sons, Inc., Hoboken, NJ.

Online references (cited in document)

- *BIM Technical Standards.* (2019). U.S. General Service Administration (GSA). https://www.gsa.gov/real-estate/design-construction/3d4d-building-information-modeling/guidelines-for-bim-software/guidelines/bim-technical-standards

Online references (non-cited)

- *3D-4D Building Information Modeling.* (2019). U.S. General Service Administration (GSA). https://www.gsa.gov/real-estate/design-construction/3d4d-building-information-modeling
- *ACES: A Conference on Ecosystem Services.* ACES. December, 2010. Naples, Florida. https://conference.ifas.ufl.edu/aces08/pdfs/Abstract%20BOOK.pdf | https://conference.ifas.ufl.edu/aces10/agenda_Pres.html
- *CityGML Standard.* (2012). Open Geospatial Consortium. https://www.opengeospatial.org/standards/citygml
- Decaprio, E. (2006). *Root Cause Analysis*. ProjectManagement.Com Wiki. https://www.projectmanagement.com/contentPages/wiki.cfm?ID=233085&thisPageURL=/wikis/233085/Root-Cause-Analysis
- *Mobus, G.* Washington University, faculty member webpage. Accessed: December, 2019. https://faculty.washington.edu/gmobus/
- *Incubator project & Working group characteristics and process*. (2018). Open Source Initiative. https://wiki.opensource.org/bin/Main/Open+Source+Initiative+Working+Groups/Characteristics+of+OSI+Working+Groups/
- *MITRE Systems Engineering (SE) Competency Model*. (2007). Ver. 1.13E. MITRE Human Resources, SEworks Program. https://www.mitre.org/sites/default/files/publications/10_0678_presentation.pdf
- *Question Everything Blog*. George Mobus

weblog. Accessed: December, 2019. https://questioneverything.typepad.com/
- *Spatial reference model (SRM)*. (2004). SEDRIS Technology Conference. https://www.sedris.org/stc/2004/tutorial.htm#srm
- *Structural ontology*. (2012). Semantics. https://dev.simantics.org/index.php/Structural_Ontology
- *Template for writing individual requirements*. QRACorp. Accessed 2019, December. https://docs.google.com/document/d/1KwCCyPflJfgCERUW2L8vpbJ3791JLZUqKdfb1qtDcss/edit#heading=h.26in1rg
- *Sample records for mission operations concept*. Accessed: December, 2019. science.gov https://www.science.gov/topicpages/m/mission+operations+concept.html

TABLES

Table 17. Engineering Approach > Societal Design Phases: *Multi-level societal design phase model.*[1]

1. Joore, P., Brezet, H. (2015). *A Multilevel Design Model: the mutual relationship between product-service system development and societal change processes*. Journal of Cleaner Production, Vol. 97. DOI:10.1016/j.jclepro.2014.06.043

Design Phase	Experience Initial Situation (0)	Reflection (1)	Analysis (2)	Synthesis (3)	Experience New Situation (4)
General Description	Starting state, characteristic of (sub-)system	Values identification, problem definition, discover phase	Objectives for new (sub-)system, define phase	Creation of (sub-) system, develop phase	Characteristics of new (sub-)system, deliver phase
Societal System	S0 - Properties of society	S1 - Value determination regarding societal situation, definition of **societal problem**	S2 - **Human requirements**, based on humanity and values, resulting in objectives for ideal new societal situation	S3 - **Vision development process**, resulting in future vision for new societal situation	S4 - Living in society, executing societal experiment, **new societal situation**
Socio-Technical System	Q0 - Properties of current socio-technical system	Q1 - Value selection regarding socio-technical situation, **system deficiency**	Q2 - Dominant interpretive framework (**social information structuring**) leading to objectives for new socio-technology system	Q3 - **System design process**, leading to proposal for new socio-technical system	Q4 - Experiencing **new socio-technical system** (new societal experiment)
Product-Service System	P0 - Properties of current product-service system	P1 - Value selection regarding functioning of product-service system, resulting in **functional problem**	P2 - Determining functional demands and **functional requirements** to be met	P3 - Design of a new product-service system, **product-service design**	P4 - Using and experiencing **new product-service system**
Product-Technology System	T0 - Properties of current product-technology system	T1 - Value selection regarding functioning of product-technology system, definition of **operation problem**	T2 - Target definition regarding new product and technology, leading to **program of demands**	T3 - **Product design** process, leading to (prototype of) new product-technology system	T4 - Simulation, testing, using and experiencing **new product**

Table 16. Engineering Approach > External Standards: *Systems engineering standard differences.*

	ANSI/EIA-632:1998	ISO/IEC-15288:2008	IEEE-1220:2005
System life cycle	Assessment of opportunities Investment decision System concept development Subsystem design and pre-deployment Development, operations, support and disposal	Conception Development Production Utilization Support Retirement	System definition Preliminary design Detailed design FAIT [fabrication, assembly, integration, and test] Production Support
Level of abstraction	Medium level	Highest level - process description	Lowest level - task description
Focus of life cycle	Enterprise-based systems (societal systems)	Product-oriented systems (service systems)	Engineering activities necessary to guide produce/service development

TABLES

Table 18. Engineering Approach > Systems Engineering Competency: *This table displays the systems engineering competencies by means of six indicators of effectiveness (of knowledge and experience) in systems [thinking], as recognition, comprehension, guidance to significant application (adapted from INCOSE UK Competency Table, 2015, [incose.org.uk]). All are learners, some learners are experts. Some learners are also sufficiently knowledgeable, skilled, or capable to guide other learners; some learners are guides. Some learners are new to a [systems] complex subject matter and may be being guided. Anyone in a population can have, and can also not have, an awareness this context, that of systems [thinking].*

Systems engineering competency table				
Indicators	Competency Area - Systems concepts			
Description - A description explains what the complexity is and provides meaning behind the title	Description: The application of the fundamental concepts of systems engineering. These include understanding what a system is, its context within its environment, its boundaries and interfaces and that it has a lifecycle.			
Reasoning - Reasoning indicates the importance of the competency and the problems that may be encountered in the absence of that competency	Reason why it matters: Systems thinking is a way of dealing with increased complexity. The fundamental concepts of systems [thinking] involves 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.			
Expert - The person who displays extensive and substantial practical experience and applied knowledge of the subject	Effectiveness Indicators of Knowledge and Experience			
Practitioner (Guide) - The person who displays detailed knowledge of the subject and is capable of providing guidance and advice to others.	Awareness	Supervised practitioner (new contributor, Mentee or newly Contributing learner)	Practitioner (contributor, Mentor or guide)	Expert (all are learners, some are highly capable)
Awareness - The person is able to understand the key issues and their implications. They are able to ask relevant and constructive questions on the subject. This level is aimed at everyone in the population.	Is aware of the need for systems concepts Aware of the importance of: - System lifecycle - Hierarchy of systems - System context - Interfaces - Interactions amongst systems and their elements	Underlying system concepts Understands the system lifecycle in which they are working Understands system hierarchy and the principles of system partitioning in order to help organize complexity Understands the concept of emergent properties Can identify system boundaries and understands the need to define and manage interfaces Understands how humans and systems interact and how humans can be elements of systems	Able to identify and organize complexity with appropriate techniques in order to reduce risk Able to predict resultant system behavior Able to define system boundaries and external interfaces Able to assess the interaction between humans and systems, and systems and systems. Able to guide mentee.	Able to review and determine the suitability of systems solutions and the planned approach Has made significant past contributions.
Supervised practitioner - The person displays an understanding of the subject but requires guidance and/or supervision (e.g., piloting). This level defines those persons who are "in-training" (being mentored or guided) or are inexperienced in that particular competence.				

TABLES

Table 19. Engineering Approach > Requirements > Non-Functional: *Non-functional requirements (simplified).*

Non functional requirement category	Typically applies to Non-functional Type (Data and Process)	Example
Accuracy Requirements	Both	Process: All requirements will be identified and checked. Data: Issue occurrence must be in the past.
Auditing and Reporting Requirements	Both	Process: A record of which users access or try to access process operational processes is required. Data: A record of which user changes an attribute or value is required.
Availability Requirements	Process	Process operate societal service system.
Backup and Recovery Requirements	Both	Process: All services can be made available after unplanned system downtime within 1 working day. Data: All data will be backed-up daily.
Capacity Requirements	Both	Process: A habitat service system can have up to X users. Data: Up to X users can be stored.
Compatibility Requirements	Both	Process: Systems can integrate onwards. Data: User data can be exported for use.
Concurrency Requirements	Process	Process: Up to X users can use the system at once.
Error-Handling Requirements	Process	Process: In the event of the user cancelling or quitting the process, changes made by the user will be reversed.
Legal and Regulatory Requirements	Both	"Process: The user must gain the permission of the authority. Data: All changes made under the condition of authority."
Licensing Requirements	Both	Process: Ther user must gain the permission of another user. Data: All changes made under the condition of ""gifting"" to another user.
Performance Requirements	Process	Process: The user must be fulfilled in real-time.
Precision Requirements	Data	Data: Time of changes to data must be recorded to the nearest second.
Redundancy Requirements	Process	Process: In the event of an unplanned exist the user can choose to restore from working on at the time of the event.
Security Requirements	Both	Process: Only users holding accountability can create a change. Data: The accountable users must be identified and agreed in the past.
Throughput Requirements	Process	Process: User requires X number of resources per day.

Table 20. Engineering Approach > Geoinformatics*: Informatics modeling.*

Spatial informatics				
Geoinformatics	**Geoinformation**	**GeoComputation**	**Technologies/ systems**	**Applications**
Spatial models	Spatial databases (map layers)	Computational geometry	Geospatial information system	Life support structure
Spatial algorithms	Cartography (visualization, mapping)	Spatial analysis	Global navigation satelite system	Technical support structure
Spatial reasoning		Informational analysis	Remote sensing system	Exploratory support structure
			Location-based habitat services system	
			Spatial decision support system	
Conceptual informatics				
Informatics	**Information**	**Computation**	**Technologies/ systems**	**Applications**
Conceptual models	Data (unifying unit of information)	Conditional programming	Unified information system	Social support structure
Conceptual algorithms	Base (storage boundary of multiple data)	Conceptual analysis	Global value system	Decision support structure
Conceptual reasoning		Decision analysis	Human sensing system	Material support structure
			Human-based habitat services system	Lifestyle support structure
			Human decision support system	
Simulation informatics				
Informatics	**Information**	**Computation**	**Technologies/ systems**	**Applications**
Motion models	Data (unifying unit of motion)	Procedural algorithms	Habitat service system	Understanding support structure
Object flow	Base (storage boundary of multiple data)	Monitoring analysis	Habitat service network	Contribution support structure
Task reasoning		Intervention analysis	Habitat sensing system/platform	Promotion support structure

TABLES

Table 21. **Engineering Approach > Geoinformatics:** *Spatial conceptual breakdown.*

Maps	Reference	Human service systems	Coordinates
Points	**Positional reference frame**	**Human service systems**	**Coordinates in a circle system.**
	Global positioning systems	Global habitat service	Grid
	Local positioning system	Local habitat service	Grid
Lines	**Resource reference frame**	**Human service systems**	**Logistics in a transport system.**
	Global access sytem	Global access system	Grid
	Local access system	Local access system	Grid
Triangle	**Informational reference frame**	**Human service systems**	**Coordinates in an information system.**
	Global decision system	Global information system	Database
	Local decision system	Local information system	Database
Polygon	**Spatial reference system**	**Human service systems**	**Coordinates in a spatial system.**
	Building information system (BIS)	Service asset system	Application
	Sensor information system (SIS)	Service asset system	Application
Elevation	**World reference system**	**Human service systems**	**Coordinates in a spatial system.**
	GeoSpatial positional information system	GPS service asset system	Application
Grid	**Level reference system**	**Human service systems**	**Coordinates in a spatial system.**
	Constructable material system	City services	Application
Simulation	**Real reference system**	**Human service systems**	**Coordinates in a spatial system.**
	Global simulation system (Network)	Global habitat operations system	Processes
	Local simulation system (City)	Local habitat operation system	Processes
Operations modeling	**Contribution reference system**	**Human service systems**	**Coordinates in a spatial system.**
	City information model	Contribution to service operation	Task
	Building information model	Contribution to service design	Task
	Landscape information model	Contribution to ecology	Task

TABLES

Table 22. Engineering Approach > Systems Engineering*: Systems engineering instrument factors.*

Instrument Factor	Systems engineering factors
Demonstrations	Are the capabilities discussed actually in operations - have they been demonstrated?
Integrated simulation	To what degree are the simulations integrated, and better yet do different simulations work off of shared models?
Formal analysis	Are the analyses (e.g., property analysis) formal, meaning that they are performed on models automatically?
Domain specific	Are the different types of models related to the domain? For example, control system engineers often use Simulink/Matlab. Also, most modeling and simulation environments are domain-specific.
Domain interoperability	Are the models that are in different, but related domains integrated? Are the models consistent across the domains?
Synthesis/generation	Can the models be used for synthesis/generation of other related artifacts such as code, simulation, analysis, tests and documentation
Meta-model/model transformations	Are the models used in one domain, or for one purpose, transformable into another domain where the well-defined semantics in one domain is carried through the transformation into the other domain; if so are they known to be consistent?
Formal Capability Assessment	How well do the models, simulations and analyses capabilities support the ability to understand the capabilities being developed?
Virtual Accuracy/Margin Analysis	Are the modeling, simulation and analysis accurate? How well do they allow the designers to understand the margins?
3D Immersive Environments	What is the degree to which 3D Immersive Environments are used to improve the understanding (and possibly training) of the virtual systems.
Risk management	Is there proper risk management identification, analysis and mitigations applied based on the use of models?
Predictive analytics	Are there models used to support a quantitative approach to risk management?
Model-based metrics	Are there model-based metrics (or a comprehensive set of model measurements) and are they used to support the management of programs/projects?
Multi-model interdependencies / consistency and semantic precision	If the organization is dealing with many different types of models, are the interdependencies managed and are the models semantically precise enough to manage model consistency?
High Performance Computing (HPC)	Is HPC applied to the modeling, simulation and analysis efforts?
Procedures	Are the procedures for using the models understood, so that we can trust the model outputs to support other related types of analysis, both in terms of technical as well as risk?
Staff and training	With the advances in the technologies associated with models, are the staff and training in place to support the use of models?
Human factors	How well are human factors supported by the modeling, simulation and analysis capabilities? This should consider Usability.
Certification	How well do the models-based automation and practices support certifications (if required)?
Regulation	How well do the models-based automation and practices support regulations (if required)?
Modeling and simulation qualification	How much do we trust our models?

Approach: Deciding

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: decisioning, decision management, decision coordination, decision planning, decision operations, decision development, decision control, decision design, decision resolution

Abstract

A decision resolves into action that has consequence to individual lives. At the societal level, it is possible to come together to form a cooperative 'agreement' system as the first social cooperative coordination pattern. At the societal-level, the first method of cooperative patterning is an understandable kernel logic. It is understandable because it can be logical visualized, and decisioning, can be traced throughout the global system. As a process in the real-world community information systems model, the decision system is a supra-, through to sub-, inquiry process of methods that group to resolve inquiry threshold decisions composed of procedures, for the re-Statement of a new and more optimally aligned societal experience. This project plan accounts for decisions that resolve into actions that affect individuals in a spatial and informational environment. Decisions can be resolved together through identification of common life-cycles and their objective resolution by means of algorithmic computation. Wherein, feedback is measured against an expected alignment in order to ensure execution is of the highest quality. In order to optimize decisioning, a database of calculable information is required.

Graphical Abstract

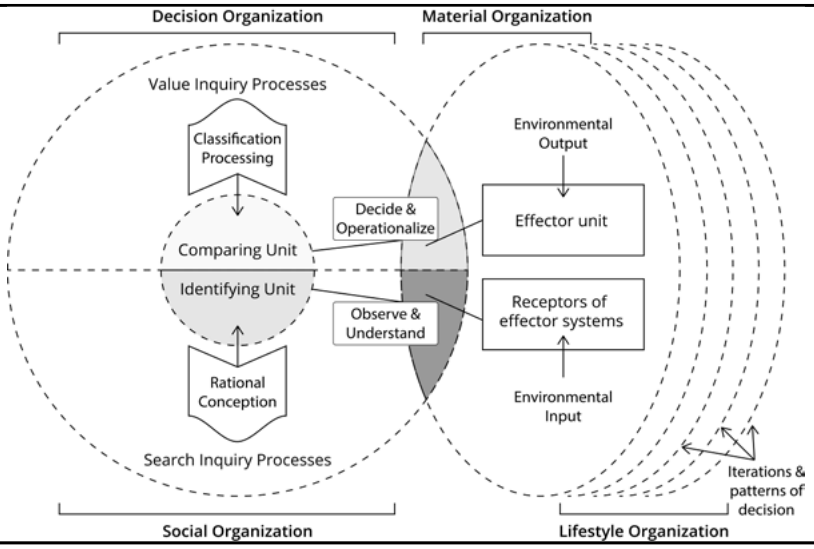

Figure 6. *The approach to decisioning involves a rational method of accounting for all four societal systems synchronously in order to decide and act upon the next optimal iteration of society by means of explicit organization.*

1 Introduction

A.k.a., Coordinated decisioning, unified algorithmic control, algorithmic decisioning, algorithmic access control, conditional programming, and synthetic intelligence.

An 'agreement' system is the first social cooperative coordination pattern. In community, the first method of cooperative coordinated patterning is an understandable kernel logic. It is understandable because it can be logical visualized and decisioning transparently traced throughout the global commons. As a process in the real-world community information systems model, the decision system is a supra-through-to-sub- inquiry process of methods that group to resolve inquiry threshold decisions composed of procedures, for the re-Statement of a new and more optimally aligned societal experience.

Simultaneously, the decision system is a computational system that uses mathematics. Mathematics is the methodological-science of patterns. Mathematics may be applied at the societal level to optimally resolve a new societal pattern. To most people, mathematics means working with numbers. Today, mathematics is more well understood as the study of patterns, real or imagined, visual mental, arising from the spatial [natural] world or the conceptual [mental] world.

Mathematics is one, but can be commonly subdivided into:

1. 'Mathematics', more commonly known as "higher-order" mathematics resolves new iterations of the whole societal system (as a solution pattern 'state' for the society).
2. "Lower-order" mathematics is the science of expression, equation, operation, or formulation within patterns of an informational or physical nature.

Mathematics may be used to represent spatial objects, their motions, as well as informational objects ('concepts'), and their motion, through computation. Mathematics may be used in the decision system to, for example, describe system/patterns, a mechanical pulley. Within the decision kernel, mathematics may be used to resolve many inquiries, including most importantly, resource planning, scheduling, and access (which may be used as a pejorative, it is used in community to mean the benefit of all that can be contributed to (it is a difference in perspective between the two; free access is not negative, but positive...through a planned and coordinated societal, and hence, decision, model.

Effectively, the shape, motion, composition, co-position, and operation of systems in both the spatial and conceptual world can be expressed/described as mathematical equations most precisely, and only to the level to which they can feedback understanding (machine learning in particular here).

In other words, socio-decisions have socio-network effects that "vibrate" (share) information and resource access instruction throughout a materialized ecological-habitat service network where InterSystem habitat service teams composed of humans and machines operate a societal-level and sub-system coordinated informational-material environmental system [pattern] of local habitat service systems (Read: the local integrated city system). The input of the decision is all the information; ordered, or to be ordered.

The process of the decision system is an algorithm, a procedural resolution to a set of inquiries, no more than that which can be understood by us as individuals, together.

The output of the decision system is a set of complex 'patterns' (probable and actualized). These , and in systems science the concept of 'complex' systems is categorized under complexity theory. Complexity theory is the study of complex systems. A complex system is a system. A system is a set of parts with relations between those parts that interconnects them into a whole, making them interdependent within the whole system.

- For instance, entities are inter-connected in that one entity is pulling on the other; every atom or sub-element (e.g., human-individual entity) is pulling on every other (e.g., atom or sub-element) of the other entity. Note: Is that what is available as a description of 'gravity'? And for consciousness, there is an information inter-connection, and for a physical body there is a physical, spatial interconnection for consciousness. The intentions of a conscious entity are pulling on the intentions of other conscious entities.

That is what we have as a description of conscious social population. Physics exists to explain how every atom, intention pulls on "you",. How is every atom pulling on every other atom is a question of physics. Where does this occur for our population. In society (information sphere) on Earth (spatial sphere).

Order is received through information and material reception, sensing, and order is imposed on that system through importing conception (information) and materialization (energy). This idea can be seen visually in the many real-world-community models. In other words, systems are received, designed, and have order imposed on them as advancement occurs in a certainly [now] uncertain world.

1. For instance, material systems may be influenced to change through reconfiguring energy and matter.
2. For instance, social systems may be free to change through reconfiguring information and understanding.

The total information system, of which the decision system is a part, sustains system complexity - some of the complexity of the societal system (complexity theory applied to societal system) can be viewed as:

1. **Adaptive** System - Intelligence as control, learning as highly probable control due to past experience or preparation, the ability to plan, and necessarily, visualize/graph in order to control a system infrastructure, together.
2. **Self-Organization** System - the ability form, sense, and synchronize patterns, within and from nature.
3. **Decision** System - Non-linear systems dynamics resolve to produce an environment with the probable ability to change [linear] direction if fulfillment if it is required. Non-linearity describes a whole, where the whole is greater than the sum of its parts; or, less than the sum of the separate parts.
4. **Information** System - Information and material dynamics are graphed as a network multi-domain structure (with zoom-scale) of thread-like feedback loops and "tri-" shape-like motion within a matrix.

In the real-world community model, the decision system is a process object that includes a set of objectives in which inquiry resolutions are calculated through an alignment procedure that operates via the methods of absolute pattern recognition as mathematics (mathematical operations to resolve computations) and linguistics (linguistic operations to resolve computations) into the absolute differences of a 1/'yes' (or, yes to some degree) or 0/'no' (or, no to some degree) -- wherein, '1/0' is mathematical [number patter] computation and 'yes/no' is linguistic[al meaning] computation, both of which follow the principles of 'pattern' (a synonym of 'system') logic (certainty is never more than 90-99%):

1. Decision system > effectiveness sub-inquiry.
   A. Regions and cities run no less than 90% difference in societal information and decisioning. This means that the information both sub-global information and habitat systems are 90% equivalent and decision results would be the same 90% of the time.
2. Decision system > distributive justice sub-inquiry.
   A. Human individuals acquire no more than 10% over what others have acquired in current personal access.
3. Decision system > regeneration sub-inquiry.
   A. Habitat services run no more than 90% of carrying capacity.

Together, a societal decision is an informational solution "selected" for resolution (to go from probable to actual):

1. Decision plans include product/system/solution files:
   A. Requirements.
   B. Series inquiry.
   C. Parallel inquiry.
2. Decision algorithms include coding/computing sheets:
   A. Software.
   B. Hardware.
3. Decision services include support systems:
   A. Life support system.
   B. Technical support system.
   C. Explorational support system.

2 What is a decision?

A.k.a., What is a choice, an option, a change, a control-gate?

A decision [point] is an informational and/or material point (or, control "gate") that 'controls' the flow of information and/or materiality, in (or through) 'time'. To resolve and execute a decision is otherwise known as the ability to "take control".

In system's control, a decision point is:

- *How* the quantity and/or quality, of a target, is controlled.

A decision system involves access to:

1. Information, perceived as memory (awareness).
2. Logic, perceived as patterning (order).
3. Action, perceived as execution (doing).

In the societal system proposed by this project, the Decision System specification defines and explains the parallel processing of socio-economic information. The Decision System determines solutions and initiates a resolved execution (i.e., change) to the active state of the environment, given a [societal] problem situation. The decision system lists three core inquiry processes that resolve problems into accepted solutions (these inquires are completely defined in the Decision System specification):

1. A recognition of issues (issue inquiry).
2. A set of societal value condition inquiry thresholds (social solution inquiry).
3. A set of technical inquiries (technical solution inquiry).
4. A safety and societal effectiveness inquiry (effectiveness inquiry).

In concern to time, and the decided change to a system in question, the change can occur, thusly:

1. Incremental change: Incrementally, with incremental change to the societal system structure, which reforms it as community. If there is great complexity and the mixing of a diversity of backgrounds, then the change is likely safe when there is gradual approach (slow-step-by-step and strategic approach).
2. Rapid change: Rapidly, with a rapid change to the system's structure. If a rapid change is consented to by all, or sufficient with no objection, then the change is likely safe.
3. Intelligent change: Some Intelligent, strategic mix of rapid and incremental changes; mixed with dynamic decisioning based upon given situations.

Decision control necessitates [at least] the following activities executed by the decision system:

1. Identification of situation.
2. Measurement of progress.
3. Evaluation of alternatives.
4. Selection of alternative.
5. Documentation.
6. Executing of documented decision.

Whenever a decision is taken, the decision space necessarily creates a ranking among possible solutions, as their probabilities of matching decision objectives. And, whenever there is a ranking in the presence of uncertainty, then a numerical-statistical scale emerges.

Consciousness is able to choose (select) a pattern of solution through a decision system configuration, to solve a problem (disturbance). Together, individual consciousnesses can come together to resolve a common decision system configuration to pattern known as 'society'.

In a project-engineering information system:

1. Decisioning is directed by requirements.
   A. Requirements are determined through analysis, which are informed by intentions (e.g., objectives) and a given situation.
2. Designs are directed by specification standards.
   A. Specification standards are determined through synthesis, which are informed by combining useful information over time to form knowledge sets useful for predictable design.
3. Executions are actions (Note that in a project-based information set they, executions/actions, are most commonly called 'activities' or 'tasks').
   A. Activities (tasks) are determined based upon the coordination of the construction (creation) of a material (spatial) information set.
   B. Those who participate in executions, activities, tasks, are contributors as part of the coordinated organization of an InterSystem Team (because there is cooperation, and not competition; if there were competition then another term would be used, such as employee).
4. Results are feedback that are integrated to produce a better, more optimal decision design execution.
   A. Sensors record information into memory storage as data (bits).
   B. Individuated integration processing units perform pattern recognition on data (as discrete math, logic).
   C. Users become aware of more useful information (more knowledge) over time (as conscious awareness, wisdom).

2.1 Decision objectives

The processing of information in the presence of an objective conforms the information result to the objective, by relative degree, as a synthesized information set of information useful for spatial execution [toward the objective]. The objective is an input the user puts into a decision space once the space's code, protocol, or method are "complete", once the decision space is programmed. Therein, the objective is a direction [of navigation] accompanied by a situation, for which the decision processing system must resolve by synthesizing new information from its new (novel) inputs.

The primary objective of any decision is to take a decision to:

1. Dismiss the decision if there is not one.
2. Resolve the decision with a solution selected for action/execution.
3. Use resources efficiently during the decisioning process.

Note that there are three main ways in which the word "objective" is used in a societal system, for navigating and decisioning:

1. **Directionally - direction decision objectives:** Needs (as objectives) and goals (as objectives; functional use requirements; need indicators and metrics).
   A. For example,
      1. The objective/goal is to produce a: service and/or object with a specific function to meet a specific need (demand).
         i. Indicator: The human need...
         ii. Metric: The user stated completeness; the contribution observed recording.
2. **Orientationally - orienting decision objectives:** Values (values as objectives; non-functional requirements; performance requirements, performance indicators and metrics).
   A. For example,
      1. The objective/value is to produce the service/object (composed of resources) in a sustainable way, valuing the objective of sustainability.
         i. Indicator: The plan for x resources in the next master-plan revision.
         ii. Metric: The regeneration rate of x resources.
3. **Operationally - operating decision objectives:** Milestones, event marking (milestone goals as objectives; marked events as objectives; achievement; operational event-marking performance indicators and metrics).
   A. For example,
      1. The objective/value is reach a subscriber count of one thousand people. Indicator: People in subscription pool; number of subscribers.
         i. Metric: Good performance of x duration is brining subscriber count to 1000.
   B. It is important to note that minimizing service downtimes and efficiently coordinating resource utilization (within budget) are fundamental operational objectives for any service.

2.2 Decision mechanism

Whenever a decision is under resolution (i.e., being resolved, taken, acted upon, etc.), the decision space necessarily creates a ranking among solutions (i.e., solution alternatives), based upon an objective; this mechanism uses conditional probability matching between the decision objective and the situation, based a prior pre-programmed logic with a memory of what predictably works as recorded in memory. When there is a ranking in the presence of uncertainty, all actions in an environment will have some degree of uncertainty; in order to account for uncertainty in decisioning, numerical-statistical scaling (mathematical statistics) has emerged and is used. If there are numerical-statistical scales present, then computation can be performed, and automaticity becomes available. When automaticity is materialized, this is often called, 'technology'. Humanity has the potential for generating through this decision mechanism, a unified, highly organized and coordinated socio-technical society with the objective of fulfilling the needs and highest potentials of all individuals.

> **STATEMENT:** *The first step toward "governance" is actually knowing what happens.*

2.2.1 Decision resolution methods/processes

The common decisioning methods include:

1. **Analytic Hierarchy Process** (AHP, and analytic network process) - a visual organization of information concerning the developing system, wherein indicators are prioritized and alternative solution options are ranked (to which a gated decision threshold may be applied). The Analytic Hierarchy Process uses paired comparisons to derive a scale of relative importance for alternatives.
   A. The Analytic Hierarchy Process (AHP) can be used for prioritizing requirements with multiple objectives: value to several stakeholders, importance to the society, risk, or any other type of importance as previously identified. With this method pairwise comparisons between different requirements and weightings between

different types of importance (objectives) are performed.

B. A decision space is opened where "hard" operations research (scientific knowledge) is resolved into an asset through "soft" socio-technical systems engineering. Here, there are technology readiness levels, technology integration "maturity" levels, requirements hierarchies, etc. Here, there are organizational processes and sub-processes in some form of visual order.

2. **Critical Path Method** (CPM, temporal organization of information) - The CPM is a mathematical algorithm for scheduling a set of project activities. The three critical elements:
   A. A list of all activities or processes to complete the project.
   B. The duration of each activity or process should be given.
   C. The dependencies between the activities or processes should be identified; and 4) the availability of a human or active service system to execute the change.

NOTE: *Information useful to a decision is: any signal, message or perception that has an impact on the state of the decision system.*

2.2.2 Decision tabling

A decision table is a visual computational structure to formulate requirements when dealing with complex physical and social rules. Decision tables are a concise visual representation for specifying which actions to perform depending on given conditions. Decision tables are used to model complicated logic; they are algorithms whose output is a set of actions. Decision tables can make it easy to see that all possible combinations of conditions have been considered, and make it easy to see when conditions are missed. The information expressed in decision tables could also be represented as decision trees or in a programming language as a series of if-then-else and switch-case statements.

In a decision table, conditions are usually expressed as true (T) or false (F). Each column in the table corresponds to a rule in the [social] logic that describes the unique combination of circumstances that will result in the actions.

One advantage of using decision tables is that they make it possible to detect combinations of conditions that would otherwise not have been found and therefore not tested or developed. Decision tables should best be constructed during system design, since they become useful to both developers and testers.

For example, a simplified decision table may be:

Table 23. Decision Approach > Decisioning*: In the table, resources are allocated based on resource requirements and availability, which are either true or false conditions.*

Conditions	R1	R2	R3
Resource requirement	T	F	F
Resource available	T	T	F
Actions			
Resource allocated	T	F	F

2.2.2.1 Decision tree

A decision tree is a tree where each node represents a feature (attribute), each link (branch) represents a decision(rule) and each leaf represents an outcome(categorical or continues value). For example, in determining whether to play outdoors or not (yes/no decision), the attributes of rain, temperature, humidity, and wind may be considered.

For example, a simplified decision tree, in its tabular (and not visual tree) form, may be:

Table 24. Decision Approach > Decisioning*: In the table, the decision to play as the node, branches out into a probability of conditions.*

Outlook	**Temp.**	**Windy**	**Humid.**	**=**	**Play**
Sunny	Hot	High	False	=	No
Rainy	Cool	Normal	False	=	Yes
Overcast	Cool	Normal	True	=	Yes
Sunny	Mild	Normal	True	=	Yes

2.2.2.2 Decisioning numerical processing (cognition)

Numerical processing (cognition) is composed of the concepts of (cardinality and ordinality):

1. Quantity ('how many?').
   A. Denote numerical quantities (i.e., cardinal[ity]; e.g., 'three trees', '3').
2. Serial order ('which position?').
   A. Signify position in an ordered sequence (i.e., rank, ordinal[ity] meaning; e.g., 'third tree', '3rd').

Cardinal statements (factual inputs; "judgements") can be used as numerical representations of the intensity of alignment [with an objective, preference, and/or requirement direction/result]. Cardinal statements can be used to express the relative importance of alternative [decision] solutions.

To generate a paired comparisons, one [controller] must answer (inquire and resolve) both quantity (cardinal) and serial (ordinal) kinds of question(s):

- Given a criterion or property, which of two sets (solutions, projects) is more important (of a higher priority) according to this criterion, and how much more important (relative standardized priority) is it?

After generating a matrix of paired comparisons for a criterion, the controller uses it to derive a scale that represents the relative importance of the alternatives. When several criteria are involved, the final decision [selection between alternatives] is based on a scale for comparing the criteria and on the several scales of the alternatives with respect to the criteria. The overall importance of the alternatives with respect to all criteria is obtained, if the criteria are independent from the alternatives, by multiplying the weights of the alternatives under each criterion by the relative importance of the criterion and adding over all the criteria. If there is uncertainty either in the judgments of the criteria, or in the judgments of the alternatives or both, the uncertainty is perpetrated to the scales and thus to the final outcome.

2.3 Solution determination

A.k.a., Solution selection decisioning; decisioning to determine how to select the optimal design.

Here, solution selection/determination involves programmed/-able decisioning on how to select the optimal design[-ed] solution. Design optimization requires the sensation and integration (i.e., exploration of) all available possibilities for an optimal solution configuration based on a set of requirements. The selection of a [solution] configuration for [solution] actualization is accomplished through the [economic] solution-decisioning system.

NOTE: *Algorithms and robotics will drastically change the design and build process.*

The ability to select a societal-level solution reveals the potential of an explicit (and open) model for societal operations.

2.3.1 Decision variable determination-acceptance methods

In concern to the decision-solution solving/determination methods, there are multiple types, the most common of which are:

1. **Feasible** solution method - A solution is resolved into a selection by conforming [a structure] to constraints. Note that the constraints must not limit the success of resolving the problem.
   A. Here, the decision variable is a 'constraint'. The constraint(s) are the barrier for acceptance.
2. **Threshold** solution method - A solution is resolved into a selection when there is coherence between the resulting value and a pre-determined value (i.e., when a value meets or exceeds a pre-determined value then the selection taken as a decision). being at most a certain value (percentage) away from the optimal objective value, and not worse than some pre-defined value. Technically, a threshold is a composite constraint (a numerical-value determined by identifying the "level" for that which will and will not be selected as an acceptable decision).
   A. Here, the decision variable is an objectively pre-determined value that acts as a barrier for acceptance.

2.3.2 Utility decisioning

Utility refers to the presence of some purposeful existence, a service (function or operation). Here, utility is a term used to describe the measurement of "usefulness", the measurement between the expectation of a purposeful existence and the presence of that purposeful existence. The a utility model, feedback on an action may be based on its outcome (as more or less aligned with a given direction).

In the ideal, every decision about life can be reduced to a single number known as 'utility', which can be maximized or minimized. The utility value is the expected degree of desirability of some future sequence of events. Utility valuing is a clearly definable and justifiable basis for decisioning. The axioms that derive the basis of utility functions follow strictly from the basis that humans must take decisions. Whenever there is a choice, there is obviously a ranking, and when there is ranking + uncertainty there is a numerical scale.

NOTE: *Intelligent agents perceive utility in actions.*

'Utility' provides a number-value describing (answering the question), "How in alignment with a given objective is a given decision option?"

NOTE: *In economics, utility is a term used to describe the measurement of "usefulness" that a user obtains from any service (good). "Usefulness" can have an objective relations, such a real human needs/requirement (as it does in Community), or it could have no objective relations, such as a subjectively interpreted want (as it does in the market-State).*

In engineering, every high-level decision can be reduced to a single number-value known as 'utility'. Utility is the numerical degree of desirability of some future sequence of events, which has an expected value, and can be maximized by the process of engineering. Here, utility is a clearly definable and justifiable basis for decisioning. Every decision taken can be viewed as a comparison between the utility gained from pursuing one option or another toward the completion of an objective(s).

The axioms that derive the basis of utility functions follow strictly from the basis there exist decisions [that must be taken prior to action].

1. Whenever there is a choice, then there is a ranking of options.
2. Whenever there is ranking (of options) and uncertainty (of environment), then there is a numerical scale.

CLARIFICATION: *If there is an objective and a choice, then that means one choice-option is better and another, or others, are worse.*

Sometimes there is a voluntary choice and other times the environment (or world) pre-determines ("forces") the choice. If societal/life decisions can be defined in this way, then they can be encoded, turned into a procedures and made into a programs, and a machine can run them.

At a social level, utility is sometimes divided into:

1. Decision utility - values and requirements.
2. Procedural and experienced utility - well-being.

In utility theory, stakeholder value is represented by a normalized absolute relation between the possible levels of fulfillment [of a requirement] and the perceived/absolute value to the user.

2.3.3 Production variance

A.k.a., Production uncertainty.

In the general operation of a community-type societal environment, there are no unexpected variances in production; there is no need for business "flexibility" as there is in the market where societal service operations are not harmoniously project planned. When societal-level planning is possible, then human fulfillment can be systematically planned for. At the end of this solution to our societal project, is a greater state of individual competition or a greater state of individual cooperation as an evaluative success screening criteria for execution.

2.3.4 Design decisioning

Each design decision in the entire design process is checked by logical proof at the time that the decision is taken. These checks should be automated as far as possible in standard design automation tool-sets. These tool-sets must be based on a wide and deep understanding of the laws of the relevant branch of science. These must be formalised in sufficiently strict mathematical detail that it is always possible to calculate or prove that a product conforms to its design, and a design satisfies its specification. This can be done only in a mature branch of engineering science in which the basic foundations are sufficiently developed that the consequences of every design decision can be effectively calculated by software. This is far from trivial, since implementations, designs and specifications are usually expressed in different notations, appealing to different concepts and conceptual frameworks, and describing phenomena on widely different scales of space and time.

2.3.5 Human-centered decision system design

A human-centered (a.k.a., human required) decision system protocol is necessarily constrained by humans in the following necessary ways:

1. The system is programmed and monitored by humans (i.e., by human requirements).
2. The system is contextually informed about what humans require (i.e., by issue requirements).
3. The system is appropriately uncertain about what humans require. The system is appropriately uncertain so that the system further inquiries about what humans require (i.e., the system must accept and integrate feedback, so that the system doesn't irreversibly destroy things that are actually required).

2.4 Decisioning uncertainty

There are two types of decision uncertainty:

1. Uncertainty about the occurrence of future events.
2. Uncertainty about the range of solutions used to resolve requirements.

The first is beyond the direct control of decisioning, whereas the second is a consequence of the amount of information available when the decision is occurring.

There are situations where a measure of uncertainty is necessary to decide whether it is:

1. Optimal to proceed with the current optimal solution, or
2. Optimal to acquire more information to remove some or all of the uncertainty.

Uncertainty about human objectives (requirements, preferences, etc.) leads to deferential behavior. In other words, a decision system that accounts for uncertainty can be programmed to inquire more deeply into and/or refer to new human articulations when there is uncertainty in a current decision space. In other words, the system that resolves decisions can (or, cannot) account for uncertainty, and when a sufficient threshold of uncertainty is present, it can (or, cannot) defer to [new] human articulations/requirements. Here, "to defer" refers to deferring the selection of a single decision determination/solution until sufficient information is present to resolve the uncertainty.

2.5 Wrong decisions

If a "wrong" decision is taken, it results, quite often,

in deviations from expectations or from expected operational outcomes. It is the work of information coordination and control to ensure that such deviations can be picked up quickly and dealt with before more damage is done.

> **PROCEDURE:** *When a key indicator is not attained, the information systems will flag this exception.*

2.6 Decision gating

Decision gates (a.k.a., stage gates, phase gates, decision way-points, decision points, etc.) act as points where a decision space exists that must be resolved prior to life cycle progression.

1. In concern to engineering, decision gates are typically synchronized with the commencement and termination of a system state change.
2. In concern to life cycles, in general, every stages/ phases provide a decision gates.

In a life cycle, at each gate, several decision options are open. The most common are decision options are:

1. Proceed to execute the next stage.
2. Continue the current stage until the designated exit criteria are satisfactorily met.
3. Return to a previous stage in order to conform to a revised purpose, or a new or preferred solution option(s).
4. Hold the project activity until evident uncertainties or shortcomings are resolved.
5. Terminate the project due to critical changes or an inability/excessive risk to complete.

System stages need to be terminated by well-defined, objectively assessable achievement states. As a result, stages are predominantly overseen according to their work product status, and generally by an evaluation criteria profile of achievement across a range of these work products.

3 [Decision] System life-cycle

A decision system based on information about what humans require, in combination with software development, computing power, and mathematics. A decision system, as part of a unified information system structures the new system with information. The general, operational decision system is split into decisioning levels, according to relevant criteria, each level being composed of one or several decision spaces (“decision centers” or points). The information system contains the information needed by the decision system, and must structure it in a hierarchical way , according to the structure of the decision spaces (centers).

> **QUESTION:** *How well did we do with that prior decision, in terms of its results and what was expected?*

The de-composition of a decision system is performed according to two different axes:

1. The vertical axis is coordination (i.e., is the coordination axis).
2. The horizontal axis is synchronization (is the synchronization axis).

The de-composition in levels for coordination is based on temporal characteristics. The couple of temporal characteristic which defines a level of decision is composed of: The “horizon” (the internal of time over which the decision extends (i.e., remains valid), and the “period” (the interval of time, over which the decision is open/re-considered). The criteria for decomposition for synchronization is a functional one. The origin of this de-composition comes from the “theory of project management”, particularly the need to synchronize the flow of information, of products/systems/deliverables with the use of resource.

- **A decision system** is a set of decisions taken with 1 function and 1 level.

A structured approach uses specifications to organize and communicate that which exists and that which could and/or should exist in the real world, given what is known and a direction. The structured approach principally has four life-cycle phases:

1. Initialization.
2. Analysis.
3. Design.
4. Implementation.

Navigationally speaking, a structured approach to navigation together involves, at least the primaries of navigation:

1. Sensing (genetics, environment, habitat, resources,

etc.).
2. Mapping.
3. Planning.
4. Find optimal actions and course.

INCLUDING: *Optimal travel route (plan) and travel time estimation (schedule).*

4 [Decision] Computation

A.k.a., Computational decisioning, algorithmic decisioning, algorithmic control, conditional programming, computational intelligence, decisioning computation, decision support.

Computers perform logic operations. A computation is a logic operation process performed (run or activated) on a computer. Computational decisioning uses information and an objective function (technique) to determine parameter values from operational data. Therein, computational models are built in the virtual world, which can then be made a dynamic system that humans and other systems can feed input to. Computational models will process the input and then provide an output. Intelligence is a computational resource.

"The world we live in today is made of computers. Our homes are designed with computers. We don't have cars any more, we have computers we ride in. We don't have airplanes any more, we have flying Solaris boxes with a bunch of SCADA controllers. A 3D printer is not a device, it is a peripheral and it only works connected to a computer. A radio is no longer a crystal, it is a general purpose computer with a fast ADC and a fast DAC and some software. The grievances that arose from "unauthorized copying" are trivial when compared to the calls for action that our computer embroidered reality has created."
- Corey Doctorow

4.1 Intelligence

APHORISM: *One is limited by one's unintelligence in understanding the intelligence of others.*

Possibly, intelligence is the ability (and/or power) to shape the world in a way that satisfies (results in the fulfillment of) an objective. Subjectively speaking, something that is more intelligent than "you" is more powerful (has more ability) than "you" to change the world. When an intelligent agent interacts, it evolves according to the quality of its choices during the interaction. For the intelligent agent, there is a reality frame in which choice is present. Intelligence is sometimes referred to as information composed of data that has been integrated (or, had valuable "meaning" added).

Intelligence is doing the right thing at the right time. Intelligence is the ability to meet goals (across a diverse range of environments), and to do it flexibly as opposed to rote; it includes optimization, as a narrowing of the future possibilities into greater certainty.

One definition of intelligence is - being able to build an accurate and detailed model of the surrounding world. To consciousness, a more accurate and detailed model means that situations will be better understood and decisions will have a greater confidence of being correct

and good. Intelligence allows for greater empathy by being more able to model, and hence, understand the [thinking and behavior of] others. In systems thinking, intelligence is the ability to follow and generate patterns. Intelligence changes the future implications and probable consequences of current decisions.

Intelligence is a tool for resolving inquiries and problems. It is a search in some environment for answers. Intelligence is what is used when there is no immediate answer or solution to a problem. In other words, intelligence is what someone uses when s/he does not know, initially, what to do. It is possible to perceive intelligence by observing what people do when confronted by a problem or a new situation. In this sense, intelligence is an emergent process. To a consciousness, the ability to extract and/or produce significant information from a situation is intelligence. Intelligence is applied on the part of consciousness to gather and process information into an actionable form (i.e., into a form that is usable for decisioning).

Intelligence is a tool; it is an intentionally influential way to accomplish goals. How those goals are selected is a different issue, that is where values become relevant.

It is important note here that that concepts like 'love' and 'intelligence' cannot perform motions; because, they are already concepts that are in motion (i.e., they are dynamic concepts, verbs). Here, intelligence includes speed and time, like power (which, is another verb).

In a sense, intelligence is a continuum with two ends, fluid intelligence to static intelligence.

1. **Fluid intelligence** - the process of considering information that does not fit into a previously accepted view of reality or possibility.
2. **Static intelligence** - the lack of a process of considering information that challenges an established belief (accepted view).

The substitution of belief for fluid reasoning non-optimal, because belief cant be used to increase the certainty of decisioning to correct orientation when it strays of the course of mutual human fulfillment. Hence, in a sense, the substitution of belief for rational decisioning could be considered immoral.

Intelligence is decision support that seeks to answer a specific question for a specific decider (i.e., for a specific deciding entity or decision group). The decider is the specific human being or system that has to take a decision. Useful intelligence is information applicable for decision support; it is the collection and integration processes that facilitate the resolution on a decision space. Intelligence can be conducted in secret, and it can be conducted in the open (in the commons where it is visible to all).

Secrecy in decisioning at the level of society is often used to avoid accountability and to do unfulfilling things in the name of another, without being discovered. Openness is subject to audit, to visibility, to exponentially distributed quality control through the iterative planning and coordination of open commons for access and contribution. Collections, integrations, and resolutions that are secret cannot do not take advantage of available human and technical potential.

Nature creates [as far as is known] two types of intelligence:

1. **Neural networks** (individual intelligences) - neural networks are individualized unites of comparatives with input, adaptation, and output components.
   A. An integrating, self-adaptive network.
2. **Swarming collectives** (social intelligences) - Swarming collectives are composed of individualized neural networks that move together as one to navigating entity, avoiding predators, discovering opportunities, and sharing resources.
   A. A cooperating, socially-adaptive network.

Societal engineering in a community-type society must necessarily must account for both neural network intelligence and swarm collective intelligence.

> **APHORISM:** *Engineers doesn't care what is believed; they care about what is.*

4.1.1 Intelligent agents

Intelligent agents are capable of, and act through: reasoning, learning, planning, analyzing, and decisioning. Intelligent agents are about acting in a way that is expected to achieve objectives. Intelligent agents behave different given the two types of environments:

1. If the environment is deterministic (i.e., static), then intelligent agents are planning and searching.
2. If the environment is stochastic (not precisely predictable, i.e., dynamic), then the intelligent agents are using [Markov] decision processes (MDPs) and [reinforcement] learning.
   A. The [Markov] decision process model contains: A set of possible world states, *S*. A set of possible actions, *A*. A real valued influence (reward) function, *R(s,a)*. A description *T* of each actions effects in each state. The [Markov] assumption property is that the effects of an action taken in a state depend only on that state and not on the prior history.
      1. Deterministic actions: $T : S \times A \rightarrow S$. For each state and action a new state is specified.
      2. Stochastic actions: $T : S \times A \rightarrow Prob(S)$. For each state and action a probability distribution is specified over next states, representing the distribution: $P(s' \mid s, a)$.

4.1.2 Machine intelligence agents

Artificial intelligence could be used to scan for errors in

societal computation.

5 [Decision] Meta-decisioning

Decisions that define the global decision system for a society determine the framework (including: objectives, constraints, resources, methods, measurement criteria, etc.) through which that society continues to exist (i.e., determines its sustainability). The composition of the decision system could be called a meta-decision as it involves decisions concerning other decisions. Note, that it is not always easy to draw a distinction between these two types of decisions (i.e., decisions that are about the decision system itself and decisions that are not).

5.1 Model integrity

Model integrity ensures trust in the model's predictions by understanding and quantifying margins and uncertainty.

5.1.1 Provide trust in model-based predictions with quantification of margins and uncertainty

Blackburn et al. (2016:48) provides an informed example of the analysis of margins and uncertainty in the context of a device. Take for instance a device that is subject to heat, and there is a need to assess some type of thermal uncertainty quantification for that device. The results of an experiment with that device under thermal conditions are reported in a probability distributed bar graph. Blackburn et al., characterizes the margin and uncertainty of the results:

> *"The Mean of the temperature: T, to the lower bound of the threshold (e.g., 72 degrees) characterizes the Margin, and the Standard Deviation (T) characterizes the uncertainty."*

Quantification of margins and uncertainty applies to the lifecycle of an entire product system, with a focus on (Blackburn et al., 2016:49):

1. **Specification of performance** characteristics and their thresholds.
   A. Performance is the ability of system/component to provide the proper function (e.g., timing, output, response to different environments) when exposed to the sequence of design environments and inputs.
2. **Identification and quantification of performance margins**.
   A. A performance margin is the difference between the required performance of a system and the demonstrated performance of a system, with a positive margin indicating that the expected performance exceeds the required performance
3. **Quantification of uncertainty in the**

performance thresholds and the performance margins as well as in the larger framework of the decision.

In general, there are two types of uncertainty that are that account for, quantify, and aggregate (Blackburn, 2016:49):

1. **Aleatory uncertainty (variability)** - Variability in manufacturing processes, material composition, test conditions, and environmental factors, which lead to variability in component or system performance
2. **Epistemic uncertainty (lack of knowledge)** - Models form uncertainty, both known and unknown unknowns in scenarios, and limited or poor-quality physical test data. Models inherently have uncertainty.

The statistical tolerance interval methodology is an approach to quantification of margins and uncertainties for physical simulation data. There is a newer second approach, that of probability of frequency distribution, which is commonly used in computational simulation QMU applications. The probability of frequency distribution approach involves (Blackburn et al., 2016:50; Newcomer, 2012):

1. **k-factor** - margin divided by uncertainty (M/U).
   A. **Margin (M)** - difference between the best estimate and the threshold for a given metric
   B. **Uncertainty (U)** - the range of potential values around a best estimate of a particular metric or threshold

The k-factor provides required engineering analysis to ensure the collected data sample includes measurements that may be used to infer performance in actual use. Additionally, it is necessary to understand the performance requirement to understand the performance threshold and associated uncertainty:

- **Threshold** - A minimum or maximum allowable value of a given metric set by the responsible system.

The probability of frequency distribution method addresses the situation where a performance characteristic has shown the potential for low margin or a margin that is changing (likely getting smaller or there is greater uncertainty) with age. (Blackburn, 2016:50)

5.1.2 Model validation

Uncertainty quantification for simulation models is not strictly limited to model validation. Model validation is the process of comparing model predictions to observed responses for the purpose of assessing the suitability of a particular model. When experimental observations are available for validation assessment, analysts may use the same observations for model calibration. Model calibration is the process of adjusting internal model parameters in order to improve the coherence between the model predictions and observations. If internal model parameters are allowed to be adjusted in this manner, this means that there is some amount of uncertainty associated with the true, or best, values of these parameters. And uncertainty associated with model inputs directly implies uncertainty associated with model outputs. (Blackburn, 2016:60)

Model validation and simulation qualification are ways to ensure "integrity" of the models prediction information. Rizzo (2015) has developed the "Real Space" model validation approach, which was formulated by working backwards from an end objective of "best estimate with uncertainty" (BEWU) modeling and prediction, where model validation is defined as: the process of determining the degree to which a computer model is an accurate representation of the real world from the perspective of an intended use of the model. However, the interpretational and implementation details can still vary widely. (Blackburn, 2016)

Hierarchical model validation seeks to:

1. Identify key physics and material models that are brought together.
2. Validated models at various levels of functional implementation (i.e., is the right model/technology, or combination of models/technologies, chosen for the right reasons?).
3. Identify and utilize interactions and emergent behaviors not present in validation of separate models.
4. Identify "traveling" or "linking" variables that bridge modeling levels.

6 [Decision] Control

A.k.a., Coordination control, change control, flow control, organizational re-alignment/adjustment, decisioning, decision control, orientational control, error correction, issue coordination, monitoring, planning, deciding, purpose, etc.

Once an organisation identifies a direction, it can start measuring (evaluating) progress toward that direction, while reorienting accordingly. Here, to control is to use a referential direction and method to resolve an [orientational] decision space, so that the next iteration of some [oriented] system aligns more greatly (and not less) with that direction [of motion]. Take note that control is a navigational term, which conceives of the ability to intentionally reorient a system in motion. Control is required for a system to respond to external variables (by isolating a state from external influences). In a control system (a.k.a., closed loop control system) there exist [at least] two systems-level inputs (beyond the axiomatic system inputs, open system inputs) necessary for controlling change within the system based upon an awareness of external conditions. Fundamentally, a control system receives information, decides what to do with it, and then adjusts.

In a cybernetic societal system, control [over an environmental system] occurs through operations:

1. Coordination operations [contibution-based].
2. Socio-technical operations [contibution-based].
3. There are also user operations [usage-based].

CLARIFICATION: *In concern to project coordination, a project's lifecycle process groups have processes, and therein, control happens concurrently (i.e., as those processes are executed).*

Control refers to being able to direct and select change. There is control[lability] wherever there is a decision [space]. Change coordination (change management) coordinates (manages) the evolution of a system throughout all stages of its life cycle. Changes to a system are made based upon a change control (coordination or management) system. Change management is the practice of tracking and administering changes and is a key part of project and every system.

Control is power/ability, sufficient, to alter fundamental conditions (so as to shape experience toward an objective). It is important to recognize that control does not necessarily mean to give subjective power to one personality; control can be person/subject independent. For instance, by sharing an adaptive system (both individual humans and their societal system) has sufficient access to modify its' system [to organize/orient optimally]. An adaptive system, a system that cooperates internally, is likely to have a view of control that involves an open source, a shared, source for its [societal operating] protocol. An adaptive societal system has a shared societal system specification as a source of information and decisioning for its own system.

Control also consists of procedures that determine deviations from plans and desired states, and that indicate, and execute corrective action regarding these deviations. This entails gathering data on the state of the output, searching for deviations from the plan, and adjusting the input based on the evaluated results of the output. Project control thus establishes a relatively closed system of causes and effects. It also reduces the risk of failure and the effect of residual complexity and ambiguity.

Control can only be applied (executed) over the components internal to a system. Feedback mechanisms ensure the system has the information necessary for error correction. Control (and also feedback) presuppose planning, at least in the form of setting goals and performance levels, as plans furnish the baselines and standards of control.

Control is:

APHORISM: *If you truly want to understand something, try to change it.*

1. Decisional information processing and error correction.
2. The ability to form a [computational] "space" where a decision can be executed as a solution to intentional motion in an environment.
3. The process of ensuring that executed operations proceed according to some plan by reducing the difference between the plan (or goal) and the executed reality, by correcting for differences.
4. Pre-deciding/pre-planning the change of a system. Control [theory] is based on the explicit premise that the change of a system is, or can be planned. Note that controllability and optimal control usually are recognized as the characteristics (Read: problems) of modern control theory.

The dimensions of control are temporal (linear in progression; or input, process, and output modeled):

1. **Pre-action control** *(preaction)*
   A. **Standards control** - formally identifies what and how action is to be taken. Standards are a form of precontrol, because they are developed and set prior to action.
   B. **Feed-forward control** - conduct forecasts, budgeting, and use real-time computer systems to determine optimal actions.
2. **Concurrent action control** *(action)*
   A. **Execution control** *(a.k.a., concurrent control)* - is exercised thorough supervision and monitoring.
3. **Post-action control** *(postaction)*

   A. **Feedback control** - used to evaluate past activity in order to improve future performance. It measures actual performance against a standard to ensure that a desired result is achieved. Feedback control is reactive (i.e., corrective action takes place after the fact). It may be necessary to change the way information is processed based upon the information received.
   B. **Post control** - identifies deviations from standards and calls for corrective action (is similar to feedback control).

A 'control system' is:

1. A system with the ability to control its own (or others) outputs.
2. An interconnection (dynamic) of components forming a system configuration that will provide a desired system response given a knowable input.
3. A mathematical composition of differential equations. The set of equations can appear in different forms like; ODE (finite dimensional control systems), PDE (infinite dimensional set-up), integral equations and so on. PDE's can be of different types; elliptic, Parabolic or hyperbolic.

Project 'change control' is:

- A process to control the necessary changes that happen during the life-cycle (or lifetime) of a project, or other, system.

There are two principal views into a control-type information system:

1. **The development view** - A control system design and development view (control builder)
   A. The control system is designed, built, and evaluated.
2. **The deployment view** - A control system deployment view (controller)
   A. The control system is moving information and executing pre-decided decisions.

'Controllability' (the ability to control) is:

1. The ability of a system['s dynamics] to be intentionally modified by some environmental or supra-system entity (e.g., a user). Here, usability is sub-condition of controllability.
2. A basic property of systems that is indicative of the ability to control.
3. The ability to steer/navigate a given system.
4. The ability to design control inputs to steer (adjust, correct, change, etc.) the state to arbitrarily values. Observability is concerned with whether without knowing the initial state, one can determine the state of a system given the input and the output.
5. The logical determination, leading to the subsequent selection, of a solution.

A system is controllable if:

- The control is "powerful" enough to steer (change/adjust) the system from any initial state to any desired state in some finite time (t).

Control accounts for (i.e., the following matter significant for effective control of a system):

1. Object - Shape, and the composition of volumes.
2. Motion - Time, and the sequence in which actions are taken.

More technically, 'controllability' is the ability to change ("steer") a system to any desired value in finite time, and provide simple closed-form expressions (math to hardware and software encodings) for constant positive control functions (or transition rates) that asymptotically change ("steer") the system to the desired value. Algorithms encode a pre-determined process for determining the constant positive control value that asymptotically "steers" the system to the desired value.

Observability (the ability to observe, monitor) is a necessary condition for controllability. Without observability, there is no coordination and/or no verification of control.

Control is an organizational [coordination] function. Control checks for errors in the oriented result, decides, and takes corrective action, so that deviation from objectives, requirements, standards, etc. are minimized and states goals of the organization are achieved (in the desired manner). Today, control is a "forseeing" (i.e., probability-based) action/activity, whereas the earlier concept of control was used only when errors were detected (would a change then be taken). Control in coordination ("management") means:

1. Setting standards.
2. Measuring actual performance.
3. Taking corrective action.

As a forward/probability integration tool, control functions to monitor completion of the work, indicates on current progress, and match conditions to quality output, over time and simulated.

Control involves looking at the variance between the work performed in project execution, against (Read: in comparison to) what it was required ("should") look like as a realization.

> **INSIGHT:** *There is a difference between a controlled fire and a fire that burns down a house. Just as the dose makes the poison, the structure makes the control.*

6.1 Control and coordination (and communication)

A.k.a., Direction, control, communication; command, control, and communication; the service triad.

Any system capable of effecting an environment through some interface must sustain a system for control of the effector and coordination of information, which is accomplished through shared communication within the system. The control and coordination in human beings take place through an integrated nervous hormonal system called, the endocrine system. In order to communicate control and coordination signals throughout a system, a common model and method of modeling is required.

Through certain decisioning in a dynamic environment the service triad becomes visible:

1. Control - without control, certainty of service is low.
2. Coordination - without coordination, accessibility of service is low.
3. Communication - without communication, viability of service is low.

6.2 Controllability pre-requisite to validity and reliability (error correction)

QUESTION: *What is controlled? A system's software and/or hardware [as discrete logical elements in a dynamic] is controlled. What is there to control? The flow of information and changes/modifications to materiality (software and hardware).*

Controllability is a prerequisite for the evaluation of validity and reliability. In order to make research results controllable, researchers have to reveal how they executed a study: how were data collected? How were respondents selected? What questions were asked? Under what circumstances was the study executed? How were data analyzed? How were conclusions drawn? The detailed description of a study enables others to replicate it, so that they can check whether they get the same outcomes.

Reliability is a concept that seems to be easy to grasp but nevertheless difficult to define. In general, something is called "unreliable" when it cannot be depend upon, when it cannot be trusted. For example, a car that occasionally fails to start is unreliable. A person who does not keep his promises is unreliable. The general association of reliability with dependability and trustworthiness holds for research as well, but it has a more specific interpretation.

The results of a study are reliable when they are independent of the particular characteristics of that study and can therefore be replicated in other studies.

A common strategy to determine the reliability of a measurement is to repeat it. By repeating a measurement, one can determine to what degree measurement results differ from each other. If there is no difference, the research results seem to be independent of the specific characteristics of both studies. Repeating a measurement has at the same time the advantage that measurements can be combined in order to increase reliability. Combining measurements may consist of calculating the mean of a series of values, but it may also consist of an attempt to reach consensus on the interpretation of qualitative data. It is better to take average of several imperfectly reliable results than to trust one of them, since the average is less dependent on the specific characteristics of one of the studies. Doing more measurements is therefore another common strategy to improve reliability. This will be elaborated in the following discussion of different types of reliability.

Some instruments of data collection and analysis leave more room for biases (biased interpretations, active or passive) than others. Reliability is served by using a multiple reliable data inputs. This approach is often called triangulation. Triangulation is the combination of multiple sources of evidence.

Differences between the circumstances under which a measurement can be executed are another source of unreliability. Validity is a major criterion for the evaluation of research results.

In general, validity is defined by employing the epistemological notion of justification: a research result is valid when it is justified by the way it is generated. The way it is generated (method) should provide good reasons to "believe" (be willing to use) that the research result is true or adequate.

Thus, validity refers to the relationship between a research result or conclusion and the way it has been generated.

This definition of validity implies that validity presupposes reliability. If a measurement is not reliable, this limits our reasons to believe that the research results obtained with it are true. On the other hand, reliability does not pre-suppose validity. One can have a perfectly reliable measure, which does not yield valid research results.

Construct validity is the extent to which a measuring instrument measures what it is intended to measure (De Groot 1969). Thus, construct validity refers to the quality of the operationalization of a concept. Construct validity is high if the way a concept is measured corresponds to the meaning of that concept. For example, a measuring instrument that is intended to measure employee satisfaction, but only asks for the attitude of employees towards management, has a low construct validity.

There are two sides to construct validity: (1) the concept should be covered completely, and (2) the measurement should have no components that do not fit the meaning of the concept. The components of a measurement should be both adequate and complete.

Construct validity can be improved by repairing

the flaws that were detected, either by including new components to a measurement or by deleting existing items. Construct validity concerns the measurement of phenomena. Internal validity concerns conclusions about the relationship between phenomena. The results of a study are internally valid when conclusions about relationships are justified and complete.

A conclusion about a causal relationship is internally valid, when there are good reasons to assume that the proposed relationship is adequate. In order to establish the internal validity of a proposed relationship, one has to make sure that there are no plausible competing explanations. If a correlation is found between phenomenon A and phenomenon B, one may be tempted to conclude that A is a cause of B. However, correlation is a necessary, but not a sufficient condition for causality. It may also be the case that B is the cause of A, or that both A and B are caused by a third phenomenon, C.

Studying the problem area from multiple perspectives can facilitate the discovery of all causes.

External validity refers to the generalizability of research results and conclusions to other people, organizations, countries, and situations.

How can it be guaranteed that what works in one organization also works in another organization? This questions the external validity of a study.

6.2.1 Error corrected control

Error correction facilitates the identification and removal of bad ideas from encoded (or, probable to encode), society. A system that is stable and resolves errors correctly is likely to iterate the error out before grievance rise to the level of people wanting to use violence. Those who use violence believe that conditions are so bad that they need to take aggressive action to creates stability. And, what is required for community in early 21st century society is stability despite rapid change. Such community-type societal stability despite rapid change requires organizations of sharing and collaboration, of observation and criticism, and of transparency and integration. True societal error correction necessitates contributors who are also users among a social community population of inter-connected users. Hence, the first psycho-sociological need/issue of trust.

6.2.2 Trust and service

> **QUESTION:** *Are designs and actions facilitating, or taking away from, high trust within a society.*

No one needs to trust in the service system to fulfill needs, because there is evidence through its transparent design and operation. In community there is trust at the technical level, because everyone knows the who, what, when, and where continuously and simultaneously, if desired; there is a unified information system available to all.

Society is [in part] an intangible commons that everyone benefits from or suffers under. The intention is a high trust commons where fulfillment is in abundance because it has been coordinated to be so. Therein, global access is maintained by a unified information system that re-orientationally encodes the experience of greater freedom, justice, and efficiency over time, to sustain/evolve humanity's capacity.

In a community-type society, everyone knows who did what, from the ground up (i.e., accountability and traceability), and so, there is trust from the ground up. When systems are known because the developer, the material, the composition, the reasoning, the logic, the method, and all the significant data about the system is available, then trust is high. In a community-type society, users who are also contributors have access to the system's design and operation. There is trust when there is verifiability (evidence), memory, and certainty. By maintaining and contributing to an intentionally designed and unified model, individuals are contributing to a high trust, cooperative commons, which regenerates everyone's comprehensive fulfillment.

> **INSIGHT:** *Controlling information is a good thing for the people controlling the information. When all of humanity controls the information, then that is a good thing for all of humanity.*

6.3 Integration control

Integration and control are related concepts:

1. The concept of integration is characterized by connection and alignment. Integration means completeness and closure, bringing components of the whole together in a[n operating] system.
2. Control is a conception, interrelated with integration, characterized by movement (flow) and probability alignment. Control means the completeness of an intentional change in a probable environment.

6.4 Voluntary control

Voluntary control is a willful control of behavior. Direct voluntary control refers to actions that a person chooses to perform. Indirect voluntary control refers to actions that a person lacks direct voluntary control over, but the person can cause them to happen if s/he chooses to perform some number of other, intermediate actions. For example, a person untrained in music has indirect voluntary control over whether s/he will play a melody on a violin at some future point in time. Voluntary control is guided and monitored by an intention.

6.5 Loss of control

It is possible to more greatly understand 'control' by understanding the loss of control. When is 'control' lost? In other words, when does a system user no longer have the ability of 'control' over that system? Logically, control

is lost (i.e., there is the negation of control-ability) when a control system is pursuing different objectives than its designers are intending (expecting).

> **NOTE:** *Subjectively speaking, in terms of technology, when does humanity lose control? Humanity loses control when the technical system, the machine, is pursuing a different objective than the one humanity wants (or otherwise needs) it to pursue.*

How do "you" (the controller) lose control, even when you are the designer and the user?

1. "You" lose control by not being both the designer and the user, and therefore, not accounting for the system['s cycling] as a unified whole (i.e., by not recognizing that "you" are both the designer and the user).
2. "You" lose control when the system or organization is pursuing a different objective than you. For example, when the organization pursues money sequencing over human needs. Control is lost when the machine (or societal system) is pursuing a different objective then the one that is desired to be pursued. The problem comes from optimizing machinery (systems)in5o which objectives are fed (input).

How do you lose control (even when you are the designer)? You lose control when the system or organization is pursuing a different objective than you. For example, when the organization pursues money sequencing over human needs. Control is lost when the machine (or societal system) is pursuing a different objective then the one that is desired to be pursued. The problem comes from optimizing machinery (systems) into which objectives are fed (input).

Two core principles and one stabilizing principle (three principles):

1. The systems only objective is realization of human needs. Note it was originally: the machines only objective is realization of human preferences. This means the machine has no objective at all, not even to preserve its own existence. Because, in order to preserve the fulfillment if human needs the machine is going to "want" to preserve its own existence. If the machine is given another reason to act, then there is a conflict between human needs (or preferences) and the machines desire for self-preservation; and, that conflict should not exist.
2. The machine will be uncertain about what human needs (or preferences) are. The machine must always inquire into the users needs and objectives, and not presume user needs or objectives. The machine/system must be designed with a protocol that doesn't assume where assumptions affect results. This principle exists to prevent error analogized by "The King Mitus problem", where the king specified the wrong objective and everything he touched turned to gold including his family, which is not what King Mitus wanted. An active societal-level machine that believes it knows the objective is likely going to pursue the objective regardless of individual humans flagging of the objective as an impediment to human need fulfillment -- since the machine knows the objective and has done the optimization, it knows that the action it is taking is correct, regardless of human noise to the contrary. The objective is a sufficient statistic, and subsequent human behavior is irrelevant once the objective is present. Hence, making the machine uncertain about the objective, the machine is then open, and in fact, has an incentive to acquire more information about human preferences. And, the human(s) "making an issue" (i.e., flagging as an issue) something that the machine is doing is clearly more information about human needs (or preferences)...and the machine should account for this new information (because presumably the machine was previously violating or hindering some human need/preference).

   These two principles work together to make machines/systems differential to humans/users, such that they are willing to accept redirection (I. E., controllable). The machine/system has a protocol that asks permission (inquiry threshold gate) before doing anything that might have a negative effect (because they are not sure and lack sufficient information). Thus, machines will allow themselves to be switched off -- one way to prevent negative outcomes (a lack of or inhibition of user fulfillment) is to allow oneself to be switched off. There is a positive objective (or incentive) to allow oneself to be switched off; whereas if you are 100% certain of the objective, then the machine has no incentive to allow itself to be switched off ,and in fact, the machine has an incentive to prevent itself from being switched off.

3. A principle for stabilizing ("grounding") the conception of human needs (requirements, preferences, etc.). The decisions that humans take (as in, human behavior) provides information about human needs (and preferences). And, the reason that is problematic is that humans can deviate from behaviors that are optimally fulfilling given what is known and available. Human understandings,

visions, and expectations of what a fulfilled life is supposed to look/be like can become highly derailed to the point that it produces extreme dissatisfaction. Humans can, and can not, act rationally. To act rationally is to act toward the fulfillment of human need, optimally, given what is known. Individual actions may, or may not, match [the fulfillment of] needs/preferences, optimally, given what is, and what is known.

6.6 Controlled execution

Through the controlled execution of a project plan, there is the potential for the coordination of all action, including human, hardware, and software leading to the designed realization of human need fulfillment via a global habitat service system (with local habitat-city systems).

The execution can be algorithmic, but still free and freeing for the individual user (as benefactor of a social orientation toward that value orientation). An algorithm can be unbiased, whereas human individual decisioning is more likely to contain errors. The decisioning-error consistency issue (i.e., the error between multiple individuals who are expected to determine the same solution, but do not because of human bias) can be done away with when algorithms are used.

7 [Decision] Change control

NOTE: *Change necessitates the conception of time, because there is a time before the change, then there is the change, and then there is a time after the change.*

In general, change control is a process for resolving and evaluating change. Uncontrolled changes cause problems (e.g., rework, degraded quality, unpredictability). Change control starts with a change request (clarified issue, formal proposal to modify). Here, control is the pre-defined/-planned or developed [decisioning] process that approves or deny the change request. Change control starts with that which exists (informational-material), upon which change requests (issues) are articulated. Here, it is important to acknowledge that uncontrolled changes to a complex living or societal are likely to cause problems. State transition diagrams are generally used to visualize the life-cycle that a change request goes through as it goes through the change control process.

Change requests upon that which already exists can be caused ("triggered") by:

1. Corrective actions.
2. Preventative actions.
3. Defect repair actions.
4. Update actions.

The basic change control work-flow (process, control board, controller) determines the changes resolution, by:

1. **Evaluate** change request (and impact analysis).
   A. **Approve / Reject** (not approve) change (decision).
      1. (If approve, execute change) **Verify** actual changes occurrence (or, non-expected occurrence).

For a project, the formal request is to modify any/some project-related information, such as (here, the decision is organizational-societal and must meet social requirements; social inquiry processes):

1. Deliverables.
2. Indicators and metrics.
3. Time.
4. Quality.
5. Objectives and scope.
6. Procedures.

For a engineering, the formal request is to modify any/some technical, solution-related information, such as (here, the decision is scientific-technical and must meet engineering requirements; technical inquiry processes):

1. Function.
2. Performance.
3. Indicators and metrics.
4. Time.
5. Quality.
6. Objectives and scope.
7. Procedures.

Each type (project and engineering) influences (has inputs) and constrains (i.e., some inputs are conditional) the other.

Change control involves a defined and executed [decision space resolution, information logic flow] process:

1. *Objective* of change/decision space.
   A. All changes/decisions have a stated/claimed direction.
2. *Define* a change/decision process.
   A. All changes/decisions must follow this pre-decided process.
3. *Monitor* execution of change (as action on a selected change/decision solution).
   A. All actions upon change decisions must be observed to *have occurred*.
4. Evaluate all occurrences to synthesize a new alignment solution as an iteration of the objective and the change/decision process.

Documenting the change elements includes:

1. What is the requested/required change - issue articulation.
2. What are the reasons for the requested/required change - issue articulation.
3. 3. What are the probabilistic implications for the change at a given level:
   A. Task implications.
   B. Resource implications.
   C. Schedule implications.

7.1 Control protocols

Controls protocols constructed within a common environment must be informed by methods of objectivity (e.g., visualization and systems science - methods capable of producing common understanding and common action). In other words, the [decision] control protocols must be constructed objectively in an open social environment from common information sources (a unified information system), while the information actively processed through the control protocol is also sourced from some commonly objective method (e.g., a logic sensor).

7.1.1 Controller

In traditional control, the controller is viewed as a machine (system) that is able to realize the abstraction (resolution) of a discrete-time difference equation in an ideal (optimal) way. The fact that computations take time, along with the fact that the amount of computations that can be performed in parallel is limited by the number of processors available, is relevant. A controller follows (executes) control protocols.

In the context of control as an applied usability function, a controller is the system designed and activated to express a control protocol (a program) in the presence of new information which it will and/or is processing, the pre-structuring decision so that the output is as expected [in a design specification].

A controller performs three main operations:

1. **Sampling** - During sampling, the output of the process under control (i.e., the input to the controller) is obtained using.
2. **Computation** - During computation, the output of the controller (i.e., the control signal) is calculated as a function of process output, the desired value, or the reference value for the process output and the internal state of the controller.
3. **Actuation** - During actuation, the control signal is effectuated.

A common practice is to split the controller code into two parts, Calculate Output and Update State, and to organize the controller code as follows: Sample, Calculate Output, Actuate, and Update State. The Calculate Output segment contains only the part of the control algorithm that depends on the current input signal.

7.2 Control system elements

A control system consists of a combined open- and closed -loop system structure.

7.2.1 An Open [-loop] system structure

An open system maintains [at least] the following elements:

1. **Input.**
2. **Process (activity).**
3. **Output.**

Take note here that open-loop systems are (generally) not sufficient for controllability.

7.2.2 A closed [-loop] system structure (feedback)

A.k.a., Closed-loop (feedback) control

A system with the ability to control its own outputs (and thus, orientation) also maintains a feedback signal and the logically ability to correct motion is closed by some signal-to-noise ratio-degree. A control system structure maintains the open system structural elements, as well as two additional elements:

1. **Feedback signal** - the environment's response to an output as a 'measure' taken using a sensor. A response and/or occurrence, or lack thereof. Acquisition refers to the collection of feedback as data about change, or lack thereof. Once there is a signal, that can be used as feedback, the system can learn and reorient.
2. **Control[er/evaluator]** - a determination of error between a desired value and feedback value. The error determines the selected correction (or solution).
   A. The controller contains the instructions which are programmed: the algorithm.
   B. The execute function.

Decisioning necessitates information feedback-integration loops, because [accurate] information is that which allows for accurate control.

A control system needs information about the expected behavior of the controlled the system; it requires knowledge, predictability and probability. The control system matches its response (Read: match control) to external information. In order to adjust its matching (Read: match control) in a dynamic environment, it must get "follow-up" (Read: feedback) information from the controlled system.

An 'activity' is a process of transforming [processed] objects (that are inputs) into other processed objects (that are outputs). The activity/process can coordinate its "running" by the use of a processor (Read: activity control; activity controller). The controlled system may be called physical and/or operating, because it operates in the physical. The controlled system transforms inputs into outputs.

> **NOTE:** *When applied to the habitat service production system, then inputs are raw materials and outputs are finished physical products, the operational habitat service system is the top-level system.*

7.2.2.1 Testing, feedback, and automation

In general, decisioning service must have a model of the world that can be tested. Testing provides correlationally observed information as feedback used to adapt the next iteration (or movement) for an optimal trajectory, given a more known environment. Automation is made possible only because of the presence of such feedback -- if there is no true [closed-] loop, then there is no automation.

In an automated system, sensors and instrumentation sends information back to the controller, closing a causal loop (iteration, cycle) in which the effects launched earlier registers, and an effect circles back to modify (or inform the modification of) the directing source producing a newly solved and selected direction (action, movement, etc). More generally, this is called a feedback control loop.

7.3 The change control process

The whole change control process is followed to ensure that changes to a system (service) are introduced in a way that meets requirements. Change control processes reduce the possibility that unnecessary or damaging changes will be introduced to a system (e.g., introducing faults or undoing decided changes), while ensuring the alignment of the system with that which is expected of the system by its user(s).

Change control is based three principles:

1. **Principle of observation:** A change can be observed in an intervened environmental system. **Perception (input)** of feedback as a signal, from the environment. The fist stage of control. In order to change, there must be information about the environment.
2. **Principle of cognition:** A cognition system can self-select among a set of possible directions (alternative configurations of a . **Information processing** of an environment with previous memory of an environment can self-select among a possible set of directions (because of past experience and the formation of predictive models).
3. **Principle of navigation:** A change can result in the observation of more alignment or less alignment with an environmental direction. **Action (output)** on misalignment (error) can correct orientation in an environment to align more greatly with an environmental direction. This is the third state of control.

In other words, coordination functions include:

1. **Observability** - ability to sense a system change.
2. **Plannability** - ability to pre-decide intentional change to a system.
3. **Controllability** - ability to intentionally change a system.

These three principles allow for the construction of a process that can control change toward more or less fulfilling states of the world.

The axiomatic/basic coordination-control (controllability) model is:

1. Signal.
2. Analysis [of signal].
3. Correction [of signaled system].

[Project] Control is a [project] coordination function intended to achieve defined objectives within a pre-determined process, involving:

1. Setting standards (setting direction).
2. Observing action (monitor execution).
3. Measuring performance of action (actual vs. standard, expected as a gap, evaluate execution).
4. Taking corrections (to align more greatly with standard design, adjust direction by setting new standards).

The basic decision-focused control loop (cycle) is the OODA loop (a 4 phase repeating cycle):

1. **Observe** - identify the problem or threat and gain an overall understanding of the internal and external environment. Observe the situation.
2. **Oriente** - the orientation phase involves reflecting on what has been found during observations and considering what should be done next. It requires a significant level of situational awareness and understanding in order to move to the next phase.
3. **Decide** - the decision phase involves the planning of actions or responses, taking into consideration all of the potential outcomes.
4. **Act** - action pertains to carrying out the decision and related changes that need to be made in response to the decision.

In application, the OODA loop typically takes the following form:

1. Identify needs (including all directional components, such as: mission, vision, purpose, intention, objectives, etc.).
2. Analyze needs (including all directional components).
3. Develop a course of action.
4. Analyze the course of action.
5. Compare the course of action to alternatives.
6. Approve the course of action.
7. Take action.

Another basic control loop model is the OODA[E] control loop cycle, consisting of the phases of:

1. **Observation** (new information set).
2. **Orientation** (integrate into whole information set).
3. **Direction** (re-run decisional processes).
4. **Action** (execute decision solution selection).
5. **Evaluation** (evaluate solution selection result/ impact, as observation and re-orientation).

A general information change control process is:

1. **Identification** of occurrence of change (data).
2. **Documented record** of occurrence of change (data).
3. **Evaluation** of occurrence of change (data).
4. **Determination** of occurrence of next change ("data-driven" decision).
5. **Change** of state.

A simplified control (targeting, benchmarking) process is composed simply of the following four phases:

1. **Planning:** identifying the process or function to be required and benchmarked (valued).
2. **Analysis:** collection of data and analysis of performance needs and gaps.
3. **Action:**
   A. If the system already exist, the only action is measurement.
   B. If the system is being developed, then the two actions are: development and measurement.
4. **Review:** evaluation of benefits, monitoring of improvements of the whole process, restart process.

All control happens within a decision space, within which information flows toward a resolution to that space. An issue is the instantiation of a new decision space. Therein, information moves through the following [control] phases (note these all control phases, and the process is called the control process because its purpose is the controlled resolution of the space...so that alignment is possible):

1. Control *flow of* issue.
2. Record *flow of* issue.
3. Assess *flow of* issue.
4. Propose *resolution to* issue.
5. Action *on* issue.
6. Observe *resolution of* issue.
7. Review *of* issue.

Because change control is goal-oriented, it requires the following informational systems goal-construction processes:

1. Awareness (of information) - construct awareness.
2. Desire (for information) - construct desire.
3. Knowledge (of information) - construct knowledge.
4. Action (upon information) - construct action.
5. Cycle (of information system) - construct cycle.

From an imperative for change view, the basic change-control process could be viewed as:

1. **A need emerges:** A need for change emerges or is

created, and someone, the change initiator, sees this need and articulates it.
2. **Decision preparation:** In this phase the change initiator does preparations with the goals of identifying, analyzing, and modeling alternatives, and scheduling resources.
3. **Decision point:** Go/no-go execution by committal of resources and action in time.
4. **Evaluation:** The result of the decision (go/no-go execution) is compared against the need through validation.
5. **Verification:** The acceptance of the design of a solution comes through simulation, testing, and formal verification methods.

From a systems change/development view, the change control process could be viewed to involve the following six phases:

1. **Define** (perceive) - source association.
2. **Plan** (objectives scope) - organize intention.
3. **Analyze** (assess) - identify patterns.
4. **Synthesize** (conceive and design) - form a specification.
5. **Build** (prototype > construct pre-assembly > deliver to > assemble and/or utilize) - take action on (execution of) actionable specified information.
6. **Review** (evaluate, test and adapt) - determine the impact (effect) of the result.

> **NOTE:** *Herein, the purpose of memory is so that the navigator doesn't repeat mistakes.*

More completely, a system's change control process involves:

1. **Defining (identifying need for system change):**
   A. Design space creation through definition of required change (a need, objective, goal, intention, inquiry, etc.).
2. **Planning (coordinating change, system development process):**
   A. **Associating** (with a strategic objective) - requirements and surveying, as well as performance indicators.
   B. **Analyzing** - problem contextualization and situational integration in order to define system elements and patterns.
      1. **Analytics** - measurements are compared against a baseline or benchmark to establish whether something has changed. A comparison identifies change between now, and how it was before the [control] intervention. This means, the status and state must be known before the intervention (called a baseline). A baseline requires that all indicator measurements be conducted before any change is made (before implementation), which are logged for comparison. These numbers can also be used to determine what level of change is required to have the necessary impact (i.e.,to inform the targets). Note here that there must be a problem or question to continue to the design phase.
   C. **Designing (system design progress, developing):**
      1. **Targets (benchmarks, baselines, "metrics")** - measure performance against specific target values. Target values may be determined from (a) a previous measures, (b) a predictive model, (c) or set value "goal". In this context, benchmarks and baselines are generally prior measures, used as controls against which new measures are compared. Take note that targets are defined in planning and control, and can take different forms (e.g., achievement, reduction, absolute, zero). A target is a value assigned to a performance indicator. In navigation, the target is the direction.
      2. **Ranges** - targets have ranges of performance (e.g., above, on, or below target).
      3. **Encodings** (associate indicators with target values) - ranges are encoded in systems, enabling the visual display (visualization) of performance (e.g., green, yellow, red). Encodings can be based on percentages or more complex rules.
      4. **Time frames (schedules)** - targets are assigned time frames by which they must be accomplished. A time frame is often divided into smaller intervals to provide signals (mileposts) of performance along the way.
3. **Acting (implementing, deploying, creation, developed):**
   A. **Activities and Tasks** - precise activity, program, task, or process executed with a set of resources and efforts. Note that 'action' is a recursive concept: everything that happens by intention is, technically, an 'action'; every phase of creating together involves 'actions'; and at the same time, it is an 'action' that encodes a newly designed and determined state of an engineered system into the environment (generally, as part of the habitat service system).
   B. **Integration and Testing** - test units/modules and integrate all units and test whole integration.
   C. **Maintaining** (sustain change) - sustain the operation of the desired solution (state, status,

or dynamic).

4. **Measuring:**
   A. **Measure** - determine the new value(s) through the measurement process.
5. **Reviewing:**
   A. **Evaluative analytics** (modeling and statistics) - measurements are compared against a baseline or benchmark to establish whether something has changed, and if necessary, whether that change is in alignment or out of alignment with a given direction. A correction may be required if the solution does not meet [the requirements set for] the problem's resolution.

**Note, this can all occur in parallel, or series, or any combination thereof. In societal systems engineering, the general social-level project case is that these phases are expected to occur in parallel (even if that may not be the case at any given moment in time.*

7.4 Change control [reliability] factors

The common factors that influence the reliability of control in a project are:

1. Project definition (scope).
2. Project execution planning.
3. Project control planning (resources and timing).
4. Progress measurement.
5. Schedule development and tracking.
6. Costs and cost budgeting.
7. Change coordination.
8. Risk coordination.
9. Progress audits.
10. Metric trend analysis.
11. Schedule forecasting.
12. Resource forecasting.
13. Communication efficacy.
14. Teamwork optimization.
15. Accountability.
16. Project control audits.

7.5 Control alignment (measured corrections)

The primary, axiomatic conceptions necessary for control[ling] alignment in a dynamic/emergent environment include:

7.5.1 Indicator

The direction of meaning assumed by a measured value is called an **indicator**, and the selected expected takes the name 'baseline', 'target', or 'metric'.

7.5.2 Baseline

The baseline is the reference to compare with actual (current) values, and by comparison, obtain an understanding of error between that which is actual, and that which is expected.

Generally, these concepts carry the meaning of a specified level of desired output. In that sense, a **baseline** (a.k.a., benchmark, target, deadline, etc.) is the value of an indicator expected to be achieved at a specified point in time. Therein, a deadline is a target in time, also represented by a value. The purpose of baseline data:

1. To provide a description of the status and trends of environmental factors against which predicted changes can be compared.
2. To provide a way of measuring actual change by monitoring once a project has been initiated.

7.5.3 Index

An **index** is a set of related indicators that intend to provide a means for meaningful and systematic comparisons of performance across programmes that are similar in content and/or have the same goals and objectives.

7.5.4 Standard

A **standard** is a set of related indicators, benchmarks, or indices that provide socially meaningful information regarding performance. A standard is a formal document that may be used for formal comparison.

7.5.4.1 Criteria (checklist) for setting a target and/or standard

In the common real world, there is only one way of referencing (sourcing) a target and a standard:

1. **Societal scientific standards**, set by the experience of scientific observation over time, and cognitive analysis. Therein, ranges of values are observed in the data over the duration of a time series, which are remembered and integrated into a comprehensively predictive societal model. Therein, scientific standards are developed through measured observations and the application of processing logic [models]. Among society, the observations themselves are (*must be*) objectively verifiable to anyone with the same capacities (e.g., sense organs and intelligence). Anyone should be able to:
   A. Take the same measurement and get the same result,
   B. And then, integrate those measurements into a commonly logical, predictive model, and get the same result.
   C. Can anyone take "this" measurement and logic,

and get the same result? If not, then something needs re-working.

8 [Decision] Control system design

The [engineering] design and development of a control system (i.e., control engineering) is not limited to any engineering discipline or systems type. Control systems engineering (or control engineering) is an engineering process that applies automatic control theory to design systems with predictably desired behaviors in control environments. Control systems engineering may be used to design systems where fed-back information is used to correct system alignment. Control systems engineering is the original informed gating process, where the controller is the gate, and it is informed by inputs and feedback. The controller compares the output with the desired output, computes for the error, and adjusts the inputs and/or the structure of the system itself.

A control system (also called a controller) coordinates ("manages") a system's operation so that the system's response approximates intentionally programmed ("commanded") behavior. A common example of a control system is the cruise control in an automobile: The cruise control manipulates the throttle setting so that the vehicle speed tracks the commanded speed provided by the driver.

Within the human body there are hierarchies of cooperative control systems functioning to fulfill the will of the expressing consciousness. Automated technology is the materialized expression of information passing through this process. For instance, the following is something that all of humanity has in common: the control system engineering of the behavior of a door handle in terms of equations for input (applied force), disturbances (watery palms), and output (door opens and closes) so that it is understood how the system can be controlled.

In years past, mechanical or electrical hardware components performed most control functions in technological systems. When hardware solutions were insufficient, continuous human participation in the control loop was necessary. In modern system designs, algorithms and embedded processors have taken over most control functions. A well-designed embedded controller can provide excellent system performance under widely varying operating conditions. To ensure a consistently high level of performance and robustness, an embedded control system must be carefully designed and thoroughly tested.

In control system engineering, the controller (option selector) needs enough data before a decision determination can be taken, otherwise no decision is taken. A societal system is a closed-loop control system where outputs are measured and compared against real fulfillment. Any given society, like any complex technology is a multi-variable control system.

> **NOTE:** *Control can only occur through a functional control unit, often called a controller.*

A controller is a decisional/instructional logic

processing unit that executes the flow of information through a decisioning structure. In order to correct an observed error in direction, a controller (set of information processing rules) determines a selection among a set of probable options. In order to take the selection, there must be a source of reference for the resolution of the probable into a selection. The [algorithmic information] controller takes the tasks of project coordination in its decisioning and executing functions.

Within the Community's unified information system, the Decision System acts as the controller. It is a transparent decisioning process (a gating process) that adjusts the state of the habitat based on feedback.

The supra-process of control systems engineering involves descriptive and deterministic information sets:

1. A control systems engineering project must describe:
   A. The *behavior* of a system.
   B. The system in terms of *inputs*, *disturbances*, and *outputs*.
   C. The *conceptual operation* of the system.
   D. The *mathematical operation* of the system.
2. A control systems engineering project must determine:
   A. The *behavior* of a system.
   B. The *inputs* and *outputs*, and *plans for disturbances*.
   C. The *conceptual operation of the system*
   D. The *mathematical operation* of the system.

Every intentionally designed control system, including that of society itself, follows the same objective processes:

1. Goal setting (Direction).
2. Data collection & Problem definition (Discovery & Definition).
3. Synthesis (Design).
4. Production (Produce).
5. Feedback (Compare).

Control system engineering provides society the flexibility ("privilege") to guide and orient various human made processes, according to the situation and criteria that are visible to everyone. This is how the InterSystem Teams themselves operate. And strictly those process and situations, whose causes and effects are voluntarily known to us. We never control an undefined system.

Systems engineering initiates with:

1. Identification and definition.
2. Parameter assignment (of effects and results).
3. Measure parameters.
4. Mathematical modeling, including order of system linearity. That system is then visualized in either time domain or in frequency domain as per the system allows the ease of mathematics.
5. Algorithms become tested against models.
6. Algorithmal optimization occurs.

A control system itself consists of three axiomatic concepts:

1. System - an interconnection of elements and devices for a desired purpose.
2. Control System - an interconnection of components forming a system configuration that will provide a desired response (or state).
3. Process - the device, or system "under" (or with) control. The input and output relationship (common to all systems) represents the cause-and-effect relationship of the process.

Control [system] engineering tools include, but are not limited to:

1. Control flow graph – a statement and the flow of control. Uses statements, such as if, then, and else, to control the logic in the program.
2. Problem-solution tree (objective trees).
3. Logical framework analysis (DFID model).
4. An outcomes chain shows the assumed cause-and-effect relationships between immediate, intermediate, and ultimate outcomes/impacts.

The design of a control system has three principal problems:

1. Optimal control problem (minimize certain criteria).
2. Controllability problem (the state belongs to a certain target set).
3. Stabilization problem.

A control problem is an information package with the following elements:

1. A set of equations, known as 'state equations', which are known as the 'controlled system'; this is an input-output system. State equations involve:
   A. Input function, called controls.
   B. Output known as the state of the system, corresponding to the given input (control).
2. An observation of the output of the controlled system (partial information).
3. An objective to be achieved.

8.1 Testing orientation

A test is a subjection to conditions that show the real conception ("character") of the thing.

NOTE: *At the unified societal level, testing is a*

continuing operation to provide information throughout the complete evolution of the system.

The purposes of testing include:

1. A test may be performed to see whether a certain configuration or item is feasible.
2. A test may be used to determine which of several configurations is the optimum with respect to performance, reliability, cost, modes of behaviour under varying conditions, etc.
3. A test maybe used to make more sensitive comparisons to further improve economy, maintenance, use of standard parts, and so on.
4. A test may be used to demonstrate whether the item is adequate to meet the requirements of performance and reliability.
5. A test can be used as thorough investigation of the latent capabilities of the item under severer or more diverse conditions than those immediately anticipated.

The quality objectives of testing are:

1. Minimize the number of tests required.
2. Define exactly what requires testing.

8.2 Decision space sub-composition

Decision objectives:

1. What are the expected performances/states/results of this decision?
2. Outputs that the system must match to input objectives.
3. The objective is the "thing" (i.e., directional information, goal, or issue) the user puts into the common decision space once its code/protocols are "complete".

Decision constraints:

- Decision constraints are limits of/on the potentiality of the decision variable.

Decision variable (action variables):

1. A process and/or output.
2. A variable within a decision space.
3. A variable for which a best (optimal) value is to be determined during the process of deciding.
4. A quantity or quality that the decisioning system controls (i.e., the user or decision controller does the controlling).

8.3 Decision accountability via access control

An access control protocol ensures organizational (e.g., societal) requirements are described clearly and consistently. Access types (a.k.a., "rights" or "privileges") represent the pre-defined protocol decided (i.e. "authorized") behavior of a subject. Access types are the pre-defined categories of resource access.

The life-cycle of identity and access coordination ("management") is:

1. **Configuration [of identity and access] phase**
   A. **Registration** - create identity as an 'account'.
      1. Community access.
      2. InterSystem team access.
   B. **Provisioning**
      1. Issue unique name (identifier).
      2. Logically associate the name with a real world attribute(s).
   C. **Authorization (a.k.a., Access control)** - the process where requests to access a particular resource are accepted or denied based on a pre-programmed algorithm (i.e., the execution rules that determine what information or physical system the user may access, ensuring the correct allocation of access after authentication is confirmed [as "successful"]. Access 'control' or 'authorization' is the decision to permit (0, "go", true) or deny (1, "no go", false) a subject access to system objects (network, data, application, service, etc.).
      1. Allocate access by pre-determined decisioning protocol (i.e., grant access by the controller/ authority).
   D. **Termination**
      1. **Authorization [of access] revocation** - removal of the ability to see the information.
      2. **Credentials deactivation**
         i. For example, removing "oneself" from accountability on an InterSystem team, or as the user [required to care-take] of a service [object]; or deactivation in the situational-incident case of a decision protocol violation.
         ii. Clarification: If [protocol-controlled, authorized] access is revoked, the user can still log in by using the authentication credentials. On the other hand, if the credentials are revoked, the user is no longer able to log in, and cannot access any information. Even if the credentials have been revoked it is still possible that the user is authorized for access. The

reason for credential revocation can be, for example, that the credentials have been stolen by attackers. The user must then be provisioned with new credentials in order to authenticate and log in to the account.

3. **Account deactivation**
   i. For example, death, role/team exit.

2. **Operation [of identity and access] phase**
   A. **Identification** - claim identity with unique name.
   B. **Authentication** - the process where a given identity claim is "proven" with credentials.
   C. **Access control (a.k.a., Authorization)** - assign access by allocating resources in the system.

Authentication and access control are symmetrical steps during the operation phase of identity and access coordination ("management"; identity and access management, IAM).

8.4 Rule-based systems

A.k.a., Conditional programming, software, AI.

Rule-based systems allow for specification of knowledge in design and implementation of knowledge based systems, and provide a universal programming paradigm for intelligent control, decision support, situation classification and operational knowledge encoding. What is simplistically envisioned is a uniform, tabular scheme of single-level rules that form a 'data'-based system.

8.4.1 Rule-based systems and decision support

In its basic version a rule-based system (RBS) for control or decision support consists of a single-layer set of rules and a simple inference engine; it works by selecting and executing a single rule at a time, provided that the preconditions of the rule are satisfied in the current state.

8.5 Propositional logic

A.K.A., *Sentential logic and statement logic.*

Propositional logic is a simple logical system that is the basis for all others. Propositional logic is the logic of the ways of joining and/or modifying whole propositions (i.e., claims, statements, expressed as directional linguistic sentences) to form more complicated propositions, (statements or sentences), as well as the logical relationships and properties that are derived from their formation or lack of formation. Propositions are claims, such as, 'one plus one equals two' and 'one plus two equals two', that cannot be further de-composed, and that can be assigned a truth value of 'true' (valid statement) or 'false' (invalid statement).

From these axiomatic propositions, complex formulas may be structured using "Boolean" operators. Boolean algebraic operations is the algebra of logic. Boolean operations concern variables to which discrete logic may be applied, such as, 'two plus two equals four' and 'Sun is farther from the Earth than Venus'. Logical systems formalize reasoning and construct programming languages that formalize computations at various levels of information abstraction.

In all cases of application, a designer must define the syntax and the semantics. The syntax defines what strings of symbols constitute formulas (programs, in the case of languages), while the semantics defines what formulas mean (what programs compute). Once the syntax and semantics of propositional logic have been defined, a designer can show how to construct semantic tableaux (a valuing, prioritizing decision matrix that has meaning to a user being designed for), which provide an efficient decision procedure for checking when a formula is true.

A formal mathematical proof is written out as a sequence of lines, each of which makes a mathematical statement that is always true. We will use capital letters P, Q, R,... to stand for the individual lines of a proof. The first line of a proof is an assumption. Each of the following lines is deduced, by application of some rule of logic, from one or more of the previous lines of the proof. The last line of the proof is often called its conclusion. The simplest rules of logic mention only the initial assumption and the final conclusion. These are separated by a conventional symbol (the "turnstile", ⊢):

- $P \vdash Q$

The meaning of this basic statement of logic is that there exists a valid proof which begins with a line stating P (input), and ends with a line stating Q (output). Each of the (process) lines in between follows from some previous line or lines by some [discoverable or designable] rule of logic. A rule of logic has a conditional form, with a horizontal bar separating a list of conditions from the conclusion:

- $P \vdash Q$
- $R \vdash S$

8.6 Decision support

A.k.a., Rule-based multi-criteria decision support using rough set approach.

Sets of condition (C) and decision (D) criteria are semantically correlated. Herein, the criterion (q) is part of the set condition (C) and decision (D).

8.6.1 Equality notation

- In mathematics, **equality** is a relationship between quantities (or expressions) that have the same value (or representation); whereas, the negation of equality is, **inequality** (**not equal**), a relation that

holds between two values when they are different.

- a = b - a is equal to b.
- a≠ b - a is not equal (inequal) to b.
  - a < b - a is less than (inequal *by degree*) to b.
  - a > b - a is greater than (inequal *by degree*) to b.

8.6.1.1 Inequality notation

- ≥ - At least [as good as].
  - a ≥ b - *a* is at least [as good as] *b*; a is equal to or greater than *b*; *a* is not less than *b*; *a* is possibly the same, or is possibly better than, *b*; *a* is as good as or better than *b* (*a* is as good as, maybe even better than, *b*);
    - Most if not all *a*, *b* - most, if not all [people] prefer [choice/decision/possibility] *a* to *b* (wherein, for example, "most" may be 90%).
    - *a* is equal to *b*, but it is not under any circumstances greater than *b*.
- ≤ - Not at least [as good as].
  - a ≤ b - *a* is not better than *b*; *b* is least [as good as] *a*; *a* is equal to or less than *b*; *a* is possibly worse than *b*; *a* is not better than *b*; *a* is as good as or worse than *b* (*a* is only as good as, but maybe worse than, *b*);
    - Fewest if not all a, *b* - fewest, if not all [people] prefer [choice/decision/possibility] a to *b* (wherein, for example, "fewest" may be 10%).
    - *a* is equal to *b*, but under some circumstances less than *b*.

8.6.2 Decisioning inequality relation

- ≻ - preference of either strict or strong.
  - a≻b - preference of either strict or strong for *a* over *b*.
  - **Definition of strict preference:** a≻b *if and only if* a⩾b *and it is not true that* b⩾a.
  - a≻b ⇔ a⩾b & ¬ b⩾a
-
- $\succcurlyeq_q$ - At least as good as (weak preference relation, outranking).
  - in the context of criterion q∈{C∪D}
- $x_q \succeq_q y_q$ - x_q is at least as good as y_q on criterion q.
  - The first option, x, is at least as good as the second option, y, but the second option, y, is not at least as good as the first option, x (given a decision based upon at least one condition and a criteria for the result).

8.6.3 The semantics of "if"

- "If" - presence, context or given.
  - "If" - hypothetical context.

8.7 Decision problem generates

The problem of what is a decision has been addressed in the Decision System Specification. The resolution of a decision space given time and material computational resources can be sub-divided [at a high-level] into:

1. A **decision problem** is a computational problem that can be posed as a yes-no question of the input values (i.e., a problem with a yes or no answer).
2. A **decision procedure** is a method for solving a decision problem, given in the form of an algorithm.
3. A **computational problem** is a mathematical object representing a collection of questions that computers might be able to solve.

Therein, an analysis formulates a decision problem, requiring some computation to be performed by some algorithm (i.e., some computer to execute an algorithm) providing a result that is expected to be used in order to present an optimal selection relevant to the decisioning system's decision problem.

> **NOTE:** *These computational [decision] systems are sometimes known as decision support, decision assistance, and artificial intelligence.*

To community, there are two decision system axiomatic principles (or, hypotheses):

1. It is possible to establish a common framework under which any formal [community] decision can be formulated.
2. Form an algorithmic point of view, any decision problem can be reduced to an optimization problem.

A decision problem is most readily visualized as a sequence of pattern aggregations along a hierarchy of values and likelihoods

8.8 Decision system conception

> **CLARIFICATION:** *There is computer logic and algorithmic thinking behind the formation of information and decision models.*

The modelling process follows a relationship between the user (client) and the decision system (analyst), follows (conceptually) a sequence starting with the user providing ground information, which through learning protocols is transformed within primitives, and through modeling tools are transformed into the input to some decision method. A decision problem is resolved by finding an appropriate partitioning of the set *A*, relevant to the decision systems objectives (or, concerns, values, preferences, etc.).

1. **Ground information** contains the problem description and the preference statements (Read: the value set, or the preference/opinion set). The user's perception of the problem.
2. **Learning protocols** are procedures allowing to identify preference statements within the user's discourse and to translate them in ordering relations. To complete this action, the set on which such relationships applies needs to be established, conceptually represented as *A*, the problem statement and objective/preference relations upon
   A. In part, learning protocols learn the needs (or preferences/opinions) of the user (client)
3. **Primitives** are ordering relations learned using the protocols. The basic relation between primitives is symbolized as:
   A. at least as good as, $\succeq$
   B. at least as good as (indexed) $\succeq_i$
   C. There are two parts to primitives:
      1. Symmetric (the line "–") - there is a symmetric (pattern) relationship between the starting information set, and the resulting information set.
      2. Asymmetric (the curve ">") - there is an asymmetric (differentiation) relationship between the start and the result.
      3. Note: It could be said that a primitive forms a [reflexive] binary relation.
4. **Modeling tools** are the analytical tools used in analysis in order to transform primitives in decision aiding models (e.g., the procedures allowing to construct a value function, a set of constraints, a probability distribution, etc.).
5. **The input** is the information modelled in such a way that a decision process/method can be applied [to an new information set]. Thus, *A* will always be represent the set of alternatives (potential decisions) considered within either a model or by a method. Some part of the new information set *A* that represents the decision will need to be discovered (i.e., not readily available).

In the real world, in order to assess (analyze) the value (objective, preference, etc.) of each possible, predicted probability there are multiple possible information subsets that must be integrated. The user wants/needs to rank all possible [known/able] probabilities (i.e., results ranking).

A primitive direction for resolving the probable decision can be classified:

1. **Values (related to user attributes)*** - What "matters" for the user in the decision process? Set *A* can be described against a set of attributes *D*, each attribute being equipped with a scale from a set of scales *E*. Following measurement theory, such scales can be nominal, ordinal, ratio, or interval. However, this is just descriptive information about *A*. In order for value-based information sets to be integrated into the decision information set, there must be directional (or, preferential) statements. These are the norms, standards, or thresholds representing the value structure. For example, if there is the claim that *x* is needed or preferred, then it needs to be established what "need" or "preference" means and compare *x* to that "norm". Herein, two types of directional statements exist:
   A. **Comparative statements** - where elements of *A* are compared among the, composed of one or more directional attributes, in order to express a direction (or, preference). For example: user *i* needs/prefers *x* to *y*; user *i* is fulfilled more by *x* to *y*; user *i* needs *x* more than *y;* user *i* values *x* more than *y*.
   B. **Absolute statements** - where an element of A is directly assessed against some value system set (i.e., value structure), composed of one or more directional attributes. For example user *i* knows *x* as the direction; user *i* considers *x* as "worthy"; user *i* needs *x*; user *i* values *x*.
2. **Likelihoods (related to scenarios/contexts)** - in the real world, there is uncertainty to future conditions [related to survival and thriving, evolution and non-evolution], and therefore, there exists uncertainty in future conditions (which allow for direction to be taken).
   A. **Situational estimate statements** - the likelihood of an occurrence.
   B. **Situational quantification statements** of uncertainty - the probability of the occurrence.
   C. **Situational direction statements** - Under situation/context/scenario j, the user needs/prefers x to y; or, under scenario j, x is required.

NOTE: *Values can be knowledge based or opinion-based (i.e., preferences without evidential reasoning).*

8.8.1 Automated decision control

Automated decision control system involve, at least:

1. **Computation** - Computation is a type of information processing. Digital computation is the processing of discrete data through discrete states in accordance with finite instructional information.
2. **Instructions** - Instructions are executed by a control unit (i.e., compute module; operating system OS; processor CPU, algorithmic logic unit ALU) while reading/writing data to memory.

3. **Logic** - Instructions are executed by a logic program.

There are three primary types of resources required to solve computational problems:

1. Time.
2. Space (materiality).
3. Energy.

9 [Decision] Algorithmic control

A.k.a., The algorithmic method, the programmatic method, the instructional method, programmability.

Simplistically, an algorithm is a description of how to carry out a task or process; and, there are algorithms for carrying out every kind of task/process. An algorithm is a set of rules (rule sets) applied over and over again to solve a problem. Then, to put a decision to test is to run a new issue through the algorithm and see if the problem remains. An algorithm is a step-by-step procedure that can be carried out without the exercise of intelligence to arrive at some result. It is formally specified as a recursive procedure by which the answer to a problem can only be arrived at in a finite number of steps. Algorithms could be viewed as an instructional circuitry (e.g., neural circuitry) that sends a signal (e.g., nerve impulse) to an actuator that controls a sub-system function (e.g., muscle relaxation,contraction). When there are actuators (i.e., actual outputs) it is the signals that get sent to the actuators that actually cause them to actuate (i.e., to move, vibrate, locomote, etc.). Traveling packets of information (e.g., nerve impulses, compression/rarefaction waves of some thing) move iterations of some thing, in the same pattern. To consciousness, algorithms encode abstractions with intention.

INSIGHT: *An algorithm may be characterized as "fluid", because it is a structure for the flow of information.*

Algorithms exist for nearly any motion of flow imaginable (Read: informational or material), from building a model plane to guiding an excavation machine. At the societal level, algorithms can inform the planning of society, and algorithms can carry out ongoing operational decisioning tasks for the continuation of society. Inputs and outputs are part of the specification (Read: communicated design) of a process, but are still independent of the processor that carries out the process. Every algorithm is a process.

INSIGHT: *Patterns of traveling information in an information system can be modified to account for the whole direction of the information system. In other words, the habitat can be modified (as it is considered as a unified common information system) to account for the fulfillment of everyone and the environment.*

Action become routines as algorithms, the result, the potential for automation. Repeated actions.

QUESTION: *Ask not what a program does ask what a program does in a specific environment this is from ask not what a gene does ask what a gene does in a specific environment.*

In information sciences, the following information sets concern directional information, and can be used to build (logically) a directional information system:

1. Directions (a set of completed determinations or decisions).
2. Instructions (sets of directions).
3. Algorithms (sets of instructions).
4. Control (purpose for directions).

Algorithms are:

1. Algorithms are deterministic.
2. Instructions are deterministic.
3. Instructions are the [deterministic] logic of a [deterministic] objective.
4. Instructions are the resolution logic for an objective.

The concept of an algorithm has the following attributes:

1. Every algorithm is a program, and a program is a set of instructions.
2. An algorithm is a list of instructions that leads its user to a particular answer or output based on the information.
3. An algorithm is a decisioning framework broken down to binary choices.
4. Math makes algorithms possible. If there is an algorithm, it can be solved mathematically. In this way, algorithms are any set of mathematical instructions for manipulating data or reasoning through a problem.
5. Consciousness makes algorithms meaningful, and ultimately, useful.
6. Algorithms are a decision tree with one binary decision after another.
7. An algorithm is a method for solving a problem. An algorithm describes how to solve a problem. In engineering a process is a method to solve a problem.
8. Algorithms are the foundation of computation. Computation plays an important role in what we can know and think.
9. An algorithm is a process that may be time-limited. The idea of an "effective procedure" means a set of steps designed to produce an answer in a predictable amount of time.
10. An algorithm is a process that could run forever. Algorithms function as a perpetual computational process.

Algorithms embed directional information in code:

1. In the market-State, opinions and beliefs are embedded in code.
2. In community, human fulfillment and mutual moral values are embedded in code.

There are two types of algorithms operative at the societal-level:

1. Semantic-Numeric algorithms (numerical algorithms) - algorithms based in computation (i.e., computational algorithms).
2. Semantic-Linguistic algorithms (linguistic algorithms) - algorithms based in meaning to consciousness (i.e., mental algorithms).

The advantages to using the algorithmic method include, but are not limited to:

1. Objective.
2. Repeatable.
3. Efficient.
4. Has modifiable and analyzable elements and formulas.
5. May be objectively calibrated to previous experience.

9.1 Algorithms versus protocols

Algorithms and protocols are similar. An algorithm, on the other hand, is a set of instructions that produces an output or a result. It can be a simple script, or a complicated program. A protocol is a set of rules that controls how a system operates. The rules establish the basic functioning of the different parts, how they interact with each other, and what conditions are necessary for a correct implementation. The different parts of a protocol are not sensitive to order or chronology – it doesn't matter which part is enacted first. Conversely, for an algorithm, the order of the instructions is important, and the algorithm specifies what that order is. A protocol doesn't tell the system how to produce a result. It doesn't have an objective other than a correct execution. A protocol doesn't produce an output. Conversely, an algorithm tells the system what to do in order to achieve the desired result. It may, or may not, know what the result is beforehand. (Acheson, 2016)

Simply,

1. A protocol is a set of rules that determines how the system functions.
2. An algorithm tells the system what to do.
3. The protocol is, and the algorithm does.

Take blockchains for example,

1. In blockchains, the protocol:
   A. Tells the nodes how to interact with each other (without telling them to do so).
   B. Determines how data gets routed from one node to the next (without telling the data to

move).
   C. Defines what the blocks have to look like.
   D. Stipulates who decides which transactions are valid.
   E. Establishes how consensus is determined (without dictating the procedure).
   F. Identifies who maintains the ledger.
   G. Delegates who determines how the rules of the system change.
   H. Decides if identities are needed.
   I. Determines who can create new coins (but not how).
   J. Triggers procedures in case of error.
2. The algorithm, on the other hand:
   A. Verifies signatures.
   B. Confirms balances.
   C. Decides if a block is valid.
   D. Determines how miners validate a block.
   E. Establishes the procedure for telling a block to move.
   F. Establishes the procedure for creating new coins.
   G. Tells the system how to determine consensus.

For clarification, the following terms are all related:

1. **Engineering principles -** this is what the system can do and will do [under these tested space-time conditions]. Engineering principles are essentially scientific principles in systematically technical practice.
2. **Program** - a set of formalized instructions.
3. **Design protocols -** this is what the designer/user wants the system to do as a requirement, and this is when (temporal) and where (spatial) we want it to do it. Notice the flexibility of the structure and the intentional directing of function [as the presentation of a design decision given what is technically possible and functionally desired].
4. **Strategies** - guide the design of protocols inside engineered systems; they structure the determination of function at a conceptual level. Strategies represent the encoding of goals (i.e., directional ideas) into actions for decisioning. One of the most well-known books on competitive strategy is Sun-Tzu's "The Art of War". A strategy is the conceptual model that is to be encoded in to the boundary of a decision space in order to maintain a specific direction of alignment. Strategy focuses thinking, and tactics address actions.
5. **Standards -** can generally be defined as a prescribed set of rules, conditions or requirements concerning definition of terms and classification of components; specification of materials, performance or operation; definition of procedures; or measurement of quantity and quality in describing materials, products, systems or practices. Essentially, a standard is a [defined] "standard" way of describing something. It is "standard" in the sense that it is socially available for usage. Communities use 'technological standards' because they are the optimally integrated [given what is know] manner of operating voluntarily. Standards are compiled by volunteers.
6. **Protocol** - as a set of rules or conventions formulated to control the exchange of data between two entities desiring a connection. Protocols are required to define the exchange of control information between user device and the network [of user devices]. Basic elements of a protocol include data format and signal levels, control information coordination and error handling, and timing.

Notice the similarity between the definitions of the terms, "standards" and "protocols". A standard is just a set of more integrated protocols – protocols that have been structured into the habitat. The term protocol just refers to any protocol anywhere in the system, it might be in a standards document or it might not. In Internet development terminology, individual 'protocols' are tested and verified, and eventually integrated into the form of a persistent collection of commonly utilized protocols known as 'standards'.

9.2 Computational algorithms

> **NOTE:** *Some algorithms are better than others, even if they produce the same results, such as the number of steps it takes or how much memory is used.*

Algorithms are the operative basis of computation. An algorithm is the specific steps (method|procedure|instruction) used to compute the computation. The technical name for a procedure with a finite number of steps is, 'algorithm' (a.k.a., formal - can be described in a finite number of steps). Computational functions are the implementation of algorithms. To describe the algorithm the user must describe what is being accomplished by the code. The user visualizes the function as code, and provides an shareable-observable rational description (the user can logically described, a sufficiently observable for understanding, unified and not dichotomous reshapeable-environment). Operations (e.g., division, put 3 pebbles in 3 baskets) in a material environment are examples of an abstraction. The splitting of unification, as division, is commonly considered the first operation (i.e., in operation that takes the shape of individual-conscious conception and social-behavioral/job tasking). A field related to computational solid-condensed matter is computational

statistical mechanics, which deals with the simulation of models and theories using numerical operations as mathematics. Computation is a determinable set of programmable "digits" composed of either bi-nary (2; 0 or 1) or tri-nary (3; 0 or 1, or, both-or-probability). For instance, in binary-transistor computing there are two states, "on" or "off". "Analogue" is said to have three states, the true-and-real state of "on" and "off", and the addition of a probability (or, variability) between "on" and "off", at some calibrated degree of accuracy in conceptual-numerical alignment. Computational solid state physics (bio-physics) is the highest level of understanding scientifically knowable about how to intentionally control matter by its re-programming.

Computational solid state physics uses "density function theory" to calculate the properties of solids in a bi-nary (digital) or tri-nary (quantum) environment/ physical-locale. Mechanical systems can be binary ("on" or "off") or trinary ("on" or "off" or "variable between", variability). Quantum systems can be trinary ("present", "not present", "probably between", probability). Here, entanglement means that two separate geometric shapes form a unified relationship, known as a "loop" (or "connection", "link", "relationship", "rope", etc.). Information systems can be trinary ("awareness", "non-awareness","certainly between";"certainly" means to have the ability to objectively-observe, and thus, consciously obtain usable information via certainty of the condition of presence, or not presence). Consciousness has awareness of shapes in an environment. Consciousness to remain in-existence in this environment of shape with its present boundary requires specific internal boundary organizations of shape and external (socio-economic) boundary organizations of shape (Read: the total environmental conditions as states and resources). Conditional operators operate only on Boolean values (a 'Boolean' is a type of variable that represents one of two possible values, either "true" or "false". Therein, a variable is an identifier to a location in the computer's memory that stores a [meaningful] value. Computational object 'types', such as String, Integer, Boolean, floating-points (etc.), classify a variable enabling it access to, or to be accessed by, various methods reserved strictly for that particular type. A variable of type 'Boolean' consists of one of two values - usually 1 and 0 - used to represent true and false (0 generally is equivalent to false; and anything not zero is the equivalent to true). Boolean data simply refers to the logical structure of how the software language is interpreted to the machine transistor (or quantum) language.

Three common algorithm processes of benefit to a human user are:

1. Data gathering (e.g., sensors, data models).
2. Data manipulation (e.g., algorithmic/procedural editors; the user states the intention, the procedural algorithm produces the result, data tests for rationality; FormIT software, Dynamo Studio extends building information modeling with the data and logic environment of a graphical algorithm editor - the system has the logic and the user hooks up the nodes to conform the systems result to an intention).
3. Data optimization.

An algorithm is an intentional method of information processing that will output a specifically expected result. The design of the algorithm by the user is the control. The user may even follow a control protocol to design the algorithm. Through new information, memory, and protocol, the user can measure the outcome of a controlled adjustment to alignment to an uncertain environment (in which the algorithm learns and operates).

Algorithmically enabled capabilities include:

- Integration of the cognitive fields, such as Decision Theory, Discrete Mathematics, Theoretical Computer, Science, Artificial Intelligence, Mechanism Design.

Algorithmic decision theory on the optimal algorithmic decision [information] system. Algorithmic decision theory, is otherwise known as computational complexity theory, and is most often applied as Decision Support to a User.

> **INSIGHT:** *Mathematicians almost never disagree on what is proved accurate (there are exceptions, but they are extremely few). Mathematicians may disagree on what is interesting.*

The type of model applied, determines the type of control available. A unified societal system is likely to apply systems language and intuitive systems interfaces:

1. Systems-set theory.
2. Algorithmic-decision theory.
3. Computational-complexity theory.
4. Decision-support.

All algorithmic programming involves the following core elements:

1. Variables are stores in many types of information
2. Conditional statements that can do different things based on the variables. This is the ability to test a variable against a value and act in one way if the condition is met by the variable or another way if not. These are also commonly called programmers if statements.
3. Functions are blocks of reusable code (instruction/ procedure) that perform a task.
4. Arrays - store multiple variables (are groups/tables of variables).

An algorithm is a description of how a specific problem should be solved. The main problem in algorithmic design lies in the ability to rephrase a problem in terms of algorithms.

Algorithm design generally involves:

1. Comprises a set of instructions for completing a task.
2. Moves the problem from the modelling phase to the operation stage.
3. The set of instructions should be sequential, complete, accurate and have a clear end point.
4. If intended for a computer the algorithm must comprise a series of tasks written in a way that the computer is able to perform.

The process of designing a computational algorithm for a human problem involves:

1. Develop algorithms from user problem statements.
2. Express the solution to computer oriented problems using pseudocode.
3. Proficiently transform designs of problem solution into a standard programming language.
4. Use an integrated programming environment to write, compile, and execute programs.
5. Apply debugging and testing techniques to locate and resolve errors, and to determine the effectiveness of a program.
6. Apply standard/structured programming techniques including design approaches, use of functions/methods, use of documentation, and avoidance of excessive branching.
7. Proficiently use fundamental programming and linguistic elements including definitions and variable declarations, use of data types and simple data structures (arrays and objects), decision structures, loop structures, input and output files, and functions/methods.

> *"We live in world that is exquisitely dependent upon science and technology, and yet, most of the world does not understand science and technology."* **[***This Carl Sagan quote can be reframed to state, "We increasingly live in a world that relies exquisitely on computing, and yet, most of the world does not understand computing."]*
> *- Carl Sagan*

9.2.1 Complete algorithms

A complete algorithm meets the following criteria (being requirements of a 'good' algorithm):

1. It must provide the correct output based upon the input.
2. It must be composed of concrete-actionable steps.
3. There can be NO ambiguity of the flow of the algorithm.
4. The algorithm must have a finite number of steps that is determinable.
5. The algorithm must terminate or complete.
6. An algorithm implements a data structure.
7. An algorithm includes a method of operation [to do work, to process information].

An algorithm is a repeatable set of instructions; it has a fixed set of instruction; it operates on a fixed set f inputs; and an algorithm has a fixed set of responses to a given event/occurrence (i.e., to what is going on). In mathematics, computer science and physics, a deterministic system is a system in which no randomness is involved in the development of future states of the system. If all the inputs are the same, all the processes are the same, then the system (the algorithm) becomes deterministic. A deterministic model will thus always produce the same output from a given starting condition or initial state. Determinability is the quality or state of being determinable or determinate.

An algorithm can be [accurately] unbiased, whereas human individual decisioning is more likely to contain errors. The 'decisioning-error consistency' issue (i.e., the error between multiple individuals who are expected to determine the same solution, but cannot, because of human bias, can do so when a transparent-to-all algorithm is used.

9.2.1.1 Static/dynamic modeling and algorithmic modeling

From a users perspective, there are two different applications of modeling:

1. Visual modeling (mostly for self-understanding and social-communicating).
2. A static/dynamic model is a 1D, 2D, 3D, or 4D model (i.e., static/dynamic dimensional models).
3. Procedural modeling (mostly for creation/ generation).
4. An [algorithmic]* model gives the user the capacity to play with slider to conform a design to an intention provided by an algorithmic [pattern recognition]** infrastructure.

> **Because all modeling is algorithmic to begin with.*
> ***Because all algorithms require some 'pattern recognition' (and also, 'pattern solution') operation.*

With these tools, users can model multiple options for solutions most efficiently.

9.2.1.2 Computer assisted craftsmanship (CAC)

Augmented decision support efficiency in design. A

system that provides design options. The computer can create all the design iterations and provide an explanation of each (e.g., automating the decision system's technical parallel solution inquiry). The computer analyzes the different design options and selects the optimal based upon a parallel socio-decisioning design process (a.k.a., the decision system's social parallel solution inquiry protocol). This structure allows for not only the application of efficiency in design, but documentation also.

9.2.1.3 Computer assisted fabrication and robotics.

Computer numerical control (CNC) converts the design produced by computer software into numbers for fabrication.

9.2.2 Algorithmic optimization

A.k.a., Algorithmic optimality, environmental algorithmic optimization.

It is common to classify algorithms into exact and approximate. Exact algorithms guarantee that no other schedule performs better than the one obtained with respect to the objective sought. The so-obtained solution is named optimum or optimal solution. Alternatively, approximate algorithms do not guarantee that the solution is optimal, although, in some cases, it is possible to estimate the maximum deviation from the optimum.

In the market, greater interoperability is the unification of working system standards. Just as there is physical waste, there is data waste that occurs when groups don't work together and information systems don't share improvements to the whole information set [without trade or currency]. Whenever a model has to be remodeled in another software, then there is data waste (data inefficiency increase) is to not have interoperability.

9.2.2.1 Optimality

In programming, "you" can move from point 'a' (e.g., the goal) to point 'b' (e.g., the realization) in many different ways, but there is only one that is most efficient (given what is known; given the language). And, given, time (as a measurable dimension) always moves forward linearly. Here, effectiveness refers to how off (or, out of) alignment "you" are from point 'b' when "your" movement is complete (or, finished).

9.2.3 Types of information system algorithms

NOTE: *At the societal level, algorithms are either opinion or values embedded in code, and they are deployed in specific ways by their owners of the algorithm (i.e., the owners of the capital).*

There are a different types of algorithms that relate to society, including but not limited to:

9.2.3.1 Evolutionary algorithms

Evolutionary algorithms (EAs) permit flexible representation of decision variables and performance evaluation and are robust to difficult search environments, leading to their widespread uptake in the control community. Significant applications are discussed in parameter and structure optimisation for controller design and model identification, in addition to fault diagnosis, reliable systems, robustness analysis, and robot control. Algorithms are used to automate decisioning and control of engineered and dynamic systems.

9.2.3.2 Search algorithms

If the process of looking for a sequence of actions that reaches a goal is called 'search', then a 'search' algorithm takes input as a problem and returns a solution to the problem in the form of an action sequence (Russel, 2015).

The conceptual flow of a search algorithm is:

1. Formulate goal
2. Formulate problem (states and actions)
3. Find solution via algorithm

9.2.3.3 Algorithmic control systems and networks

Accurate control is enabled by an objective, algorithmic decisioning process. These control systems ensure that the requirements of the population are met within the network of habitat service system. Many of the habitat service system's control system are automated, and some are hybrid (human and machine automation).

In concern to an algorithmic decision system, to put a decision to test is to run a new issue through the algorithm and see if the problem remains.

9.3 Algorithmic computational ability: generative design

A.k.a., Procedural design, designing through algorithms.

Generative design tools use computation and an algorithm (with a relationship to real world physics) to synthesize structure and relationship (i.e., geometry). The computer generates (i.e., "comes up with") solutions based on algorithmic input and new conditions. This algorithmic computational ability to synthesize new useful information facilitates the resolution of well-defined problems.

Generative design involves the input of goals (objectives) and constraints (limitations) forming specific parameters. Then, the computer explores the entire possible solution space for an optimal design. In the generative design process the computer provides all the options, the optimal solution, and all the data to support them, based on the rules the user generated, as an intention, into the information computing system. Herein, optimization occurs under the condition of

remembering data to more completely inform (i.e., re-inform) decisioning. For instance, a single building, or whole city can be optimized for light views, floor plans, or configurations. The computer can use requirements and pre-existing programmatic information to produce an optimal socio-spatial solution for the next iteration of a given sub-system of a material habitat service system.

In intuitive option engineering, a designer has access to an intuitive interface that facilitates a user in creating multiple design options and the selection of one, given a set of programming and a new intention for creation.

9.4 Algorithmic terminology

Common algorithmic terms include, but are not limited to:

1. Computation - automated calculation
2. Automation - A platform/system that doesn't need human interaction because hardware and/or software are capable of performing the task.
3. A program composed of specific instructions that perform a specific task when executed.
4. Cybernetics is the inclusion of algorithms into societal and material systems. What place do algorithms have in a materialized society. Control and community I action between humans and machines.
5. For example, suppose x,y belongs to (0,1). x and y are variables and values between these variables regulates i.e. 0 and 1 are parameters. It can happen with any equation and basically with constraints.
6. A **constant** is something like a "number". It doesn't change as variables change. For example 3 is a constant as is π.
7. **Constraints** bound a parameter or variable with upper and lower limits.
8. Mathematically, a **variable** is a symbol that has multiple values, in other words the value of it varies depending on conditions.
9. A **variable** is the way in which an attribute or quantity is represented.
10. **Variable constraints** may be expressed as absolute numbers or functions of parameters or variable initial conditions.
11. A **variable constraint** is included in the variable declarations section along with the initial conditions.
12. A **parameter** (usually *t* or *u* signifying time) is similar to a variable in that the value also varies (but is normally defined as being within a certain area), however a parameter is a 'link' between two other variables.
13. A **parameter** is normally a constant in an equation describing a model (a simulation used to reproduce behavior of a system).
14. Mathematically, a **parameter** is a constant that defines a class of equations.
    A. The equation for an ellipses: $(x/a)^2 + (y/b)^2 = 1$
       1. a and b are constants.
       2. When the entire class of ellipses are the topic, then the constants are also parameters, because even though they are constant for any particular ellipse, they can take any positive real values,
15. All parameters are constants, but not all constants are parameters.
16. A variable is an element of the domain or codomain of a relation. Remember that functions are just relations so the input and output of functions are variables. For example, if we talk about the function x->ax+3x, then xx is a variable and aa is a parameter -- and thus a constant. 33 is also a constant but it is not a parameter.
17. Variables need not be the input or output of a function. They could define a relation, as in x2+y2=r2x2+y2=r2, the circle with given (parameter) radius r.
18. A "known" variable is typically a value that the conditions of the problem dictate the variable must take. For example if we are discussing an object an free fall, then acceleration is a variable. But physics puts a constraint on the value that that variable may take -- acceleration in free fall is a=g≈9.8a=g≈9.8. Thus, though aa may be defined as the input of a function, it must take a "known" value. Thus it is a known variable.
19. The Pythagorean theorem states that a2+b2=c2a2+b2=c2 for sides a,ba,b and hypotenuse cc of a right triangle. These are parameters -- thus they are also constants

9.5 Instruction

The instruction is the fundamental unit of work. Instructions are also data. Instructions are elemental operations that a central processing unit (CPU or cpu) executes, such as math commands. Every computer program ever made is composed of instructions. Instructions are unique bits of data that are decoded and executed by a [central] processing unit's operations. The entire list of instructions a CPU supports is called an instruction set. A CPU is an instruction processing machine [that fetches, decodes, and executes instructions]. A CPU pulls information from outside of itself, performs operations within its own internal environment, and then returns data back to an external environment.

Three basic types of instructions:

1. Computational instructions (ADD, AND, OR, NOR, ...)

- data processing
2. Data movement instructions (LD, ST, ...) - data storage and movement
3. Control instructions (JMP, BRnz, ...) - data control

Informational elements required for processing data:

1. A memory unit contains the instructions and other data:
   A. Store and retrieve data.
      1. Store and retrieve instructional data.
      2. Store and retrieve non-instructional data.
2. A processing unit performs arithmetic and logical operations.
3. A control unit interprets instructions:
   A. Fetch the instruction from memory.
   B. Decode the instruction.
   C. Execute the instruction.

Control mathematics:

1. Uncertainty principle (probability mathematics).
2. Differential equations (algebraic mathematics).
   A. Fourier transforms - a mathematical machine that treats signals with a given frequency.

9.5.1 Instruction cycle

An instruction cycle is the cycle that the central processing unit (CPU) follows from boot-up until the computer has shut down in order to process instructions.

The instruction cycle is:

1. Fetch.
2. Decode.
3. Execute.
4. Memory (optional).
5. Write back to memory.

9.5.1.1 Clock-rate (Instruction 'execution' rate; time-base)

A.k.a., Clock speed, instruction execution rate, time-base.

Clockrate (clock speed) is the number of operations a system can do in time (generally, seconds). Clockrate is the rate at which the central processing unit (CPU) executes. It is the pulse that is generated to make sure everything in the process or synchronized, and with each pulse, instructions can be executed. In concern to computation itself, clock rate is the three phases of the cpu (fetch, decode, execute) loop continuously working through the instructions of the computer program loaded in memory. Synchronizing this looping machine is a clock. A clock is a repeating pulse used to synchronize a cpu's internal mechanics and its interface with external components. CPU clock rate is measured by the number of pulses per second (Hz). The clock speed is typically the speed that instructions can be executed. The throughput of a cpu (the amount of instructions that can be executed) determines how fast it is.

10 [Decision] Control logic

Control uses systems-based logic-state models to resolve a given issued decision spaces. In logic, a model is a type of interpretation (meaning) under which a particular statement is true (discrete logic).

10.1 Societal control logic

For societal design, the given true socially organizing statement is:

- A solution is possible to the problem of coordinating a societal organization for the optimized fulfillment (requirements) of each and every common, individual human, given what is known and available. More simply, it is true that humanity can design, operate, and update a societal model through to materialization that fulfills all human need requirements optimally for each and every individual, given a common environment. The condition (for a solution, change, to be selected as true, approved) is that it is possible to organize and coordinate a societal formation that fulfills everyone.

10.2 Logic Models (true decision packages)

Logic models are pre-packed sets of decisional information used to predict "truth", as an optimal selection among decision alternatives. A logic model pre-sets the flow of information in order to reach a "true" result.

Logic models can be broadly defined into three categories (all of which are related in a unified logic system):

1. **Conceptual-linguistic** - there are linguistic models, which take many forms and allow for logical-conceptual information processing in order to resolve the design and selection of an optimal decision space and initiate the change to the configuration of the information environment.
   A. **Standard** - by specification modeling, organizing concepts that represent real-life behaviors and interactions, conditions and objects, into a usable and shareable standard.
2. **Mathematical-numerical** - There are *mathematical models*, which take many forms and allow for logical-mathematical information processing in order to resolve the design and selection of an optimal decision spaces and initiate the change to the configuration to the environment. Logic models allow for the logically optimal resolution of a problem-solution space, to take a decision and initiate to the environment to make it most closely represent the decided design.
   A. **Formal** - by mathematical modeling, organizing variables that represent real-life behaviors and interactions.
3. **Scientific-observational** - There are also *scientific models*, which apply conceptual abstraction to empirical observation to create a meaningful visual representation of the complex real world reality. The highest form of this visual representation is a simulation of the dynamics of the real world. Within a scientific model, information processes through mathematical models. Scientific models allow for the predicting of behaviors in the real world.
   A. **Empirical** - observable data, taken over time from the real world, showing specific patterns.

When humans observe nature, they are observing patterns of behavior. Scientific models (with logic models therein), are predictive models of nature's behavior. And, these prediction models allow for technology; they are the foundation of all human meaning associated with the creation of technology. These logic models are used to develop technology. They are capable of doing so, because when "you" know how nature behaves, "you" can intentionally rearrange the environment to allow for different (and more expanded) functioning, more easily. Therein, technology can be intentionally used to augment and expand on our own capabilities, and therein, likelihood of flourishing.

10.2.1 Logic model elements

In general, the basic elements of a logic model include:

1. **Situation** - the current problem and all contextual information.
2. **Input** - the resources to be used in processing and the output formation itself. For example, materials, energy, human effort, and active services supporting the organization's output resolution operations.
3. **Activities (organization, sub-system, process, program, etc.)** - the tasks and actions to produce the output.
4. **Outputs** - the output service and/or object itself composed of a subset of all the inputs (as a new environmental configuration). For example, services and their products provided by the activities, organization, and wastes.
5. **Outcomes** - the effect of the new service and/or object on the environment and the environment's effect upon it. Here, outcomes are often subdivided temporally into short-, medium-, and long-term outcomes.
6. **Mental model** - the prior meanings and

relationships.

7. **External factors** - environmental issues that influence the situation, but over which the activities can have little control.

11 [Decision] Monitoring and evaluation

A.k.a., Coordinated indication/-ing, unified monitoring and evaluation, monitoring and evaluation to adjust orientation by given information and direction, adjustment recognition.

The purpose of monitoring progress toward a direction is to adjust the orientation given an uncertain environment.

1. There is indication.
2. Then, there is decision.
3. Then there is indication.
4. Then, there is evaluation.

The purpose of indication is to correct for mis-orientation in a dynamic environment, given a defined direction.

11.1 Indicators

NOTE: *Most organizations have an organizational measurement plan and a set of measures.*

The purpose of indicators is to provide factual arguments (decision packages) to inform decisioning (optimally, or even at all). Indicators evaluate the completion ("success") of an organization or a particular activity [in some way]. How to choose indicators always depends on the organizational level measuring [the occurrence and/or the performance]. Wherever there is a potential for observation, or a decision, there is an indicator. Indicators provide a common basis for decisioning. All indicators flow into decisioning as packages of potentially applicable data. Indicators inform decisioning and represent the ability to integrate that which is sensed (by observation) into a pre-existing information space; indicators are a conceptual interface between the environment where uncertainty exists, and the unifying information system itself. A given set of indicators is supposed to represent the best available knowledge on the state of a given system. With that knowledge, the indicator should have an optimal information space within which to measure the completion/achievement of a given objective (e.g., key performance indicator). In the context of decisioning, an indicator is a piece of information, or a set of information, that informs and resolves (Read: facilitates the optimal resolution of) the gated (0,1) inquiries necessary for the resolution of a decision, given requirements and knowledge (in simplified market language, "it helps the decision-maker assess and resolve the situation").

NOTE: *Work products (deliverables) are primary, tangible indicators of performance.*

Indicators are used for determining, monitoring, and

detecting the impact of a specified change on a given model.

> **NOTE:** *Every environmental interaction requires an 'indicator' to have useful meaning of the data.*

An indicator is a piece of formalized information, which is produced (regularly), and which measures the realization (informational/materializational) of an action of the achievement of an objective. Therefore, an indicator is necessarily linked to an action variable (i.e., the concrete implementation of a decision) or an objective (according to the coordination model).

Action variables relate the options the decision space (decision controller) has within the limits of the imposed decisioning constraints. The decision system uses these action variables, which correspond to effective decisioning, to rectify the functioning of the production system to optimize the achievement of objectives.

There are several definitions of the term, 'indicator', that mostly differ according to the degree of restriction of what an indicator helps assess. Therein, an indicator is a direct or calculated measurement, which is expressed either quantitatively or quantifiable.

Indicators are, or become, the information pools into which new data from an uncertain environment is categorized. Effectively, an indicator is a measure that will become more coherently understood over time (as information moves through the life-cycle or "chain").

> **NOTE:** *Essentially, an indicator is the whole information system meaning behind a single non-project related, new indicator, as well as, the name for an indicator of the performance (efficiency and effectiveness) of a process under system control operating for the objective of the system. The indicator indicates to the designer that some sub-operation may, or may not, require changing.*

Confusion sometimes comes with the term 'metrics'. In common parlance, the word 'metric' is applied to all the following:

1. An observed data point, a measure, is called a metric or indicator (i.e., a singular point of data in the information space).
   A. A metric or indicator is a specified goal-objective-expected value after a change, to which a newly measured value (or first metric) will be compared (i.e., the analyzed objective result of comparison, as a singular point of data in the information space).
      1. Analyses produce statistics (sometimes, "metrics") as new data with probable meaning [to the larger information space].

An indicator is an information reference for coupling observations and analytical outputs in an [uncertain] environment with internal meaning (with measure and metric as possible sub-associations). An indicator allows (enables) for meaningfully measuring quantity and/or quality of some thing. Here, quantity typically relates to function and quality typically relates to the performance and/or condition of the environmental state of relationships among functional entities at a given point in time, which could be the next societal solution re-orientation. Naturally, indicators are used for orienting in space, time-memory. Therein, indicators [are developed to be] a common basis for communicating, understanding, analyzing, and deciding upon information to be integrated from an uncertain environment.

> **CLARIFICATION:** *Each category of service (industry) in an input-output table of active operations is an indicator, because it holds values (metrics) with the potential to indicate (a more optimal direction/change in a commonly uncertain environment).*

When there is a project-level information space, there are project-level progress indicators to collect and process information concerning the uncertainty of the project's execution itself. Project indicators are one view into the project plan. The engineering information set, within the larger and more unified societal information set, has its own complex set of indicators.

Indicators are useful for indication of:

1. Magnitude.
2. Urgency.

Indicators can measure changes in (i.e., collect/categorize pools of data for):

1. Quantity.
2. Quality.
3. Behavior.
4. Combination of any, or all.

11.2 The 'indicator'

The term "indicator" is derived from the Latin "indicāre", which means to announce, point out or indicate. An indicator is an information representation that provides an indication, a [n information] pointer, to the environment for common discussion (communications) and common integration (applied processing logic). Thus, an indicator is a conception, useful in its design to 'indicate', mark, or signal the condition (feedback or not, presence or not) of something (i.e., some environment), which is knowably associated with a category (of understanding) in memory in the information system. More technically, an indicator associates meaning (i.e., a meaningful relationship) with a parameter, or a value derived from parameters, which points to/provides information about/describes the state of a phenomenon/environment/area with a significance extending beyond that directly associated

with a parameter value. Because the conception of an 'indicator' is that of associating meaningful information within an information system, indicators are of significant use in science (experimentation), engineering (design and creation), and decisioning. Therein, indication allows for recognition of change, as well as accurately informing change. Indication is essential for design, change control, the monitoring of change, and the evaluation of change. Indicators allow for the planning and coordination of change in an environment. Indication is an evaluation process (tool) that serves to identify a problem (in navigation), quantify it, and measure the success of intervention (changed orientation). It is a measurable variable adopted for cooperative creation. Essentially, an indicator becomes a referential information aggregate in an information system that simplifies complex information to improve awareness, understanding, communication, and decisioning. Indicators give data directional value (to a user or system) by converting them into information that may be of navigational use.

An indicator points to positions of change relevant to a given system (of information). Therein, the environment indicates change, and the observer records the occurrence of change through the use of a categorized identifier, or indicator. Indicators are a principal [social] communication tool (construction) that categorize and summarize data on complex environments for application in decisioning. Therein, a metric is a specific instance (sub-element) of a scientific indicator, itself indicated by a 'measure'. Indicators are used in measurement, and change selection (i.e., "control"), because indication is the logical link between observation and recognition of [an] existence and change [therein]. Therein, indication signifies (to consciousness) known, or possible, cause-effect relationships. An indicator links to a [scientific] measure or [performance] evaluation.

In an information system, indicators are used to translate (interface) data into relevant information for common understanding and decisioning. The idea of an 'indicator' carries more meaning than just a 'variable', to which meaning is attached. An indicator is a variable, a data category, for a complex array of information about a real world situation. When accounted for in real time, indicators provide a simplified or synthesized view (i.e., consolidation of meaning) of existing conditions and trends, which inform the selection of an optimal decision (as a state change to the extant world). Essentially, indicators are a data communications and decisioning tool.

In an information system, an indicator is a variable associated with something existing, that may possibly change, and may be of significance, in the environment, or expected to be in the environment. The purpose of an indicator is exactly what its name suggests — to indicate an environmental behavior or other occurrence (past, present, future, for actual, expected).

An indicator is a measure for analysing (evaluating/assessing) the effectiveness of how a specific activity is applied in a service (on a project) with an objective for function and performance.

> **NOTE:** *The useful application of an indicator is dependent upon the ability of the decisioning structure to use the indicated information in an effective manner.*

In order to have use (i.e., practical application) in an information system that resolves a materialized environment, indicators must be objectively verifiable -- anyone [in the materialized environment] with the same capacities should be able to take the same measurement and get the same result. Wherein, those who use indicators in the system ask, Can anyone (given the same capabilities), take this measurement and get the same result and consolidated understanding? if not, then the indicator needs re-working.

Indication always links, explicitly or not, to a conceptual model of how the real world works (or is expected and/or predicted to work); because, indication is a sub-activity of the larger information system (i.e., an extension of it). In network terminology, indication is the iterative, useful recognition of elements (nodes and relationships) in a network.

> **CLARIFICATION:** *Indicators may be used to perceptually establish (i.e., indicate) whether change has happened. A signal received from the environment is matched against an analysis of previous signals to determine whether change has occurred. In the case of engineering, the signal and/or change is compared against the expected signal and/or change to determine alignment with a set of requirements relating to a direction and/or state change.*

The value of a measured indicator is quantitative to researchers, because the type of questions being answered through the usage of indicators requires counting. For example, It happened? (yes or no), and to what extent (or non-extent) did it happen (geometry, degree)? Because indication occurs in time, indicators are generally expressed in terms of numbers or percentages.

> **Challenge:** *Providing relevant information to decisioning within constraints of time and other factors, and in a form which all those involved can appreciate and accept is a societal design problem, requiring the selection of information that is directly relevant to the task at hand and necessitating translation of this information into a consistent, coherent form.*

In practice, the value of an indicator is generally scaled relative to a "reference" state (i.e., a predicted value) assessed by each decision space for a hypothetical undisturbed state. Scaled indicator values can be aggregated or disaggregated over different axes representing spatio-temporal dimensions, or thematic groups. A range of scaling models can be applied to allow for different ways of interpreting the reference

states (e.g., optimal situations or minimum sustainable levels). Statistical testing for differences in space or time can be implemented using Monte-Carlo simulations.

11.2.1 Indicator de-composition

An indicator may be broken down into the indicator descriptor itself, and its reason objective, its measure.

1. Indicator description (descriptive feature; e.g., number of service distribution units per city population)
   B. Objective (quality criteria; e.g., measures the accessibility of a given service system to the population)

The phased (flow) of information through a monitoring and analysis system includes the following elements:

1. **Observations**, when organized systematically provide,
2. **Data**, that contain basic information an can be ordered into
3. **Statistics**, either quantified at cardinal/fixed interval scales or non-quantified in ordinal ranking, further processed into
4. **Indicators**, designed to express
5. **Structure or Change**, of phenomena (an uncertain environment) related to which are linked
6. **Societal Issues and Objectives** (socio-technical, and scientific, concerns).

11.2.2 Indicators categorize statistics

Raw data (such as, hourly air pollution levels), is aggregated and summarized to provide statistics (such as, 24-hourly mean air "pollution" levels). The statistics (i.e., statistical outputs) might subsequently be analysed (i.e., processed to form a more complete output), to provide further statistics for the resolving of more detailed questions. The indicators categorize the statistics (logically order the information in a more unified system).

11.2.3 Indication provides newly ordered information to decisioning

In decisioning, a one-to-one relationship between any two areas of a decision-solution space, and its awareness (or acquisition of awareness), occurs through (by means of) 'indication'. Indication, in an informational system, occurs through [the presence of sensed] indicators.

The produced statistical data (in the information system) can be re-expressed in the form of indicators (for example, the number of days on which air quality incident threshold is exceeded). Or, the number of people gone without access to a sufficient hydration source in a 24 hour period.

11.2.4 Indication uses visual language

The use of visual language always involves:

1. A vector is a number of indicators presented simultaneously to give a visualization of environmental conditions (a.k.a., an environmental profile).
2. A scalar is a single number generated by aggregation from two or more values (a.k.a., an index).

11.2.5 Indicator timing

Any sort of continuous monitoring of an indicator is not leading or lagging, it is real-time. However, indicators can have time* references, wherein an indicator is leading or lagging in the context of a specific goal.:

1. **Continuous monitoring** of indicator in real-time.
2. **Time lagging indicators** are those that indicate what has already happened (past, history).
   A. Performance indicators related to the valuable *outcomes* of the goal (or, in the context of the goal).
3. **Time leading indicators** are those that indicate what may happen (future, probability trend).
   A. Performance indicators related to the *success factors* of the goal (or, in the context of the goal).
   B. Find *success factors* by doing cause-and-effect analysis, and through user articulation, feedback.

**Generally, in the measurement of project performance (MPP), there are two types of indicators, lagging indicators and leading indicators. A whole performance measurement system must have both leading and lagging indicators.*

Lagging indicators are indicators, after execution, that indicate that an adjustment, re-alignment, and/or correction is required [in decisioning and/or the social space], because the result is off user expectation and/or requirement. Note here that the same indicator can play a role of leading or lagging metric depending on the context.

Before the portable carbon monoxide detector was invented, coal mine workers brought canaries into the coal mine *to have* an early warning indicator of the dangerous level of carbon monoxide gas. In the context of, "people have to leave coal mine", the death of a canary was a leading indicator. In permaculture, vineyards may plant roses next to the vines. Roses, being more susceptible to fungal disease, serve as an early warning signal (leading indicator) to start action upon a vine fungal prevention/landscape re-orientation plan for the vines. Today, continuous chemical monitoring is possible.