## 1.2 Basic Terminology, Delineated

A proper understanding of the approach described in this text depends on understanding the terminology being used. Terminology arises as an expression of particular aspects of underlying concepts. This is a particular challenge in the context of software methodology where commonly used terms are often not commonly, or even clearly, defined or understood. Certain basic concepts, organized into four categories—methodological, organizational, operational, and developmental—with associated terminology, are set forth here as fundamental to understanding the described approach.

Terminology as presented here differs in varying degrees from conventional usage (or a reader's prior understanding)—these differences are not arbitrary. They differ for any of several reasons: to broaden prior unnecessarily parochial definitions of terms; to resolve ambiguities in the usage of terms with similar meanings, sometimes used interchangeably; to align more closely with how a term is defined in other fields; or simply to better convey meaning within the scope of the presented methodology. An appreciation of such differences will in fact provide some initial insight into how the approach described here differs from other approaches.

### Methodological Concepts

#### *Elements of a Methodology*

A ***methodology*** is an integrated body of principles, practices, and methods that prescribe the nature and proper performance of the whole of a specified process. A given methodology may accommodate differing approaches by allowing for alternative practices and methods, as long as conformance to its defining principles is maintained.[1]

A ***process*** is a set of delineated activities that are performed as needed to achieve an associated objective. An ***activity*** is characterized by its purpose and the knowledge and expertise required for its performance.

[1] This text describes a *methodology* for the engineering and manufacture of software-based products.

A ***method*** is guidance and criteria that prescribes techniques and practices for a systematic, repeatable performance of a given activity. A ***task*** is an allocation of resources for the performance within a specified scope of one or more method-defined activities.

### *Objectives and Goals*

There is ambiguity in common usage of "objective" and "goal". As used herein, An ***objective*** denotes the motivation for some undertaking—an overarching but informally-defined aspiration for what is to be accomplished. An objective has an associated set of ***goals*** that together provide concrete criteria for evaluating the progress an endeavor has made toward achieving that objective.

### *Models and Documents*[2]

A ***model*** is a (simplified or partial) representation of an item, sufficient as a basis for obtaining approximate answers to a designated set of questions about that item. A model provides indicated information about an associated item without requiring the item itself to be inspected or even accessible. The purpose of a model is determined by the questions it can be used to answer. Answers must be accurate relative to the referenced item but may be only "approximate" in that more exact answers would require a time-consuming or otherwise infeasible direct inspection of the item itself.

A ***document*** (or "documentation") is an identifiable organized communication of information on some topic for a designated audience. A document may be a primary source, serving as a (typically informal) model of an item, a supporting source of explanatory material for another item, or a secondary source providing content derived from models or documentation concerning one or more other items.

### *Specifications and Realizations*

A dependency between two items is expressed as a "specification-realization" pair. A ***specification*** is an item (e.g., a model or document) that prescribes criteria that another item of interest must satisfy to be considered valid.

---

[2] D.Parnas, "Precise Documentation: The Key to Better Software", *The Future of Software Engineering* (Conference), Jan 2010. <www.researchgate.net/publication/221349970>

A ***realization*** is an item whose content is meant to conform to an associated specification. A realization is valid if its content is consistent with all relevant specifications but may be incomplete (e.g., omit or restrict specified capabilities or quality criteria). Any change in either item of a previously consistent specification-realization pair may establish an inconsistency that requires further change in either or both of the items to reestablish consistency.

## Organizational Concepts

An ***enterprise*** is conceived or formally chartered as an agent for one or a cohort of individuals or public or private organizations. An enterprise is the organizing authority for endeavors that target the attainment of designated objectives.

### *Programs and Projects*

An enterprise organizes its associated efforts into one or more programs or projects. A ***project*** is initiated with an allocation of scope and resources to achieve a specified time-limited goal (e.g., to perform an activity, provide a service, or build a product that meets the needs of a targeted customer). An enterprise may initiate multiple projects to address customers having different needs.

If an enterprise has objectives in differing areas of interest or institutional competence, it can delegate responsibility for endeavors of indefinite duration in a given area to a ***program***. A program is established and separately managed as the governing authority over one or more associated projects initiated to address the needs of customers in a designated market corresponding to its specified area of interest.

### *Markets and Customers*

A (potential) ***customer*** is a person or organization having an endeavor that could be achieved more efficiently or effectively through the employment of appropriate products or services (i.e., ones having capabilities conducive to such endeavors).

A ***market*** is a set of potential customers for a given type of product (or equivalently service). A market is "coherent" if the needs of those customers are sufficiently similar that each can be satisfied with a similar product. A "simple" market is a coherent market that consists of a single customer or in which all customers are accepting of the

same product, willing and able to adjust their operations as required to fit the product rather than requiring the product to be customized to fit their specific needs and preferred practices. (By convention, the terms “customer” and “simple market” will be viewed as equivalent and may be used interchangeably.)

## Operational Concepts

### *Environments, Entities, and Systems*

An ***environment*** is a space-time, containing identifiable active and passive elements, within which entities of interest operate. An ***entity*** is a person or device, or coordinated aggregate thereof, having agency (being able to undertake certain actions) within an environment. An environment or entity can be physical, virtual, or augmented (i.e., containing both physical and virtual elements) and is characterized by properties that manifest the effects of associated naturally-occurring and entity-initiated phenomena.

A ***system*** is an aggregate entity, representing the collective behavior of an associated set of entities. The specific purpose and extent of a system, and its composition, is delineated subjectively based on which entities are perceived as being agents of that system. The separate behaviors and interactions of its associated entities may be attributed separately or to the system as a whole.

An ***ecosystem*** is a system—an environment and the entities that operate within it. An entity can be influenced by the behavior of other entities operating in the same environment, either indirectly via an entity’s effects upon the properties of the shared environment or directly via the entities having a means to share information or coordinate action. Entities operating within a shared environment may be seen to be interacting cooperatively, competitively, or equitably with regard to their respective purposes.

People participate in an ecosystem (1) through direct interactions with other people, (2) as ***users*** of devices that provide access to the mechanisms and information content of a system, and (3) as ***operators*** of equipment to monitor and control associated detection and initiation of observable phenomena. Relative to a given system, a person is characterized by the ***roles*** that they are authorized to perform within that system. A role

corresponds to rights a person has to access information and the capabilities of a product according to their responsibilities in the operation of an enterprise.

### *Information, Data, and MetaData*

***Reality*** is the properties and phenomena exhibited in the space-time of an ecosystem (physical or virtual). ***Information*** is a selective characterization of reality from the perspective of one or more observers (i.e., entities). Entities have the means to obtain or change information associated with relevant aspects of reality. Specific aspects of reality can be static, intermittently changing, or continuously changing over time. The information associated with an environment or entity is defined abstractly as its "*information space*", only some elements of which may be susceptible to being directly observed or modified.

***Data*** are a representation of relevant elements of an information space. Data is determined by measurements of observable properties and phenomena, by communication among entities, or by derivation from other data (e.g., data conversion, filtering, inference, signal processing, time-series or probability distribution). Data is valid to the degree that it is sufficiently precise, accurate, and timely in representing corresponding information. However, validity can be limited by the mechanisms available to observe or derive the properties and phenomena associated within a given information space.

***Metadata*** indicate how data relate to corresponding information. Metadata can be used to characterize the identity of associated data, its representation (i.e., how it encodes corresponding information), and its provenance (i.e., when and how its value has been determined). Metadata can further specify the basis for and confidence in the validity of the data, any means for maintaining consistency with changing information and other related data, and any alternatives for rendering the data for delivery to recipient entities.

### *Hardware, Software, and Platforms*

***Hardware*** is any physical apparatus (e.g., biologic, chemical, electronic, mechanical), or appropriate aggregation thereof, whose purpose is to actualize a conceived process.

***Software*** is an encoding by which such a process can be expressed, broadly in terms of obtaining, transforming, deriving, retaining, and dispensing information encoded in an analog or digital form as data. Software operates through the medium of hardware built or determined to be suitable for enacting its intended behavior.[3]

More generally, software is a specification of behavior that is realized in a form that is expected to induce that behavior by means of interactions with entities operating in a shared environment. Specifically, software is a human-decipherable ("source") representation of intended behavior that has an equivalent ("object") representation that specifies the operations that compatible hardware will perform to produce that behavior.

From a software perspective, it is useful to distinguish between two categories of hardware: platforms and devices. A hardware platform is the physical infrastructure (i.e., physical housing and power, network management, communications, computational, and environmental hardware) that enables a collection of devices to coordinate activity and share data. A device (sometimes referred to as an "instrument" or collectively as "equipment" or "machinery") is hardware that provides capabilities for detecting or producing phenomena in its containing ecosystem.

A "software platform" (e.g., an operating system, runtime libraries, translator, or virtual machine/container/hypervisor technology) is software built upon a hardware platform to allow other software to be built without regard for the specific physical structure or properties of underlying hardware. Software-defined behavior can differ depending on the behavioral capabilities of its underlying software platform, which in turn depends upon the behavioral capabilities of any associated devices and its hardware platform.

A "computational platform" is a software platform, and associated devices, that serves one of three product-related purposes: development, evaluation, or operations. A development platform provides the means to build a product. An operations platform

[3] An electronic apparatus directly enacts software-encoded behavior. In other cases, the mechanism may differ: a chemical or mechanical apparatus can be constructed to produce behavior corresponding to a software encoding of its intended operation; similarly, viewing DNA as a biologic form of software encoding, genomic machinery enables an organism to enact DNA-specified behaviors.

provides the means to operate a product in service of a customer enterprise. An evaluation platform is an (actual or facsimile) operations platform, augmented as needed to simulate the customer ecosystem and instrumented for the monitoring and analysis of product operation for conformance to prescribed behavioral qualities.

***Devices***

Devices provide capabilities by which software can interact, via a computational platform, with its ecosystem. A computational platform provides access to one or more associated devices, each having any combination of four types of functionality:

- A "*computational*" device is hardware whose purpose is to effect software-defined behavior
- An "*edge*" device provides services for detecting or producing phenomena in the environment or associated entities
- An "*interface*" device provides services by which a product is able to interact with users/operators or system entities to coordinate actions or share data
- A "*storage*" device provides services for managing access to (locally or remotely) persistently stored data

The behavior of a device is determined by the physical realization of its capabilities (e.g., the timeliness and fidelity with which it acquires data or the speed and precision with which it produces or responds to phenomena), limited by the capabilities of the platform on which it operates (e.g., speed and latency of communications among devices). A device may associate metadata with data that it provides to indicate its provenance (i.e., source and timing of measurement or computation).

In general, software resides on a virtualized computational device operating as a coordinated aggregate of one or more communicating computational devices supported on a shared hardware platform. A set of related devices can be packaged (i.e., physically integrated) or interconnected via a shared platform to form a single logical device that provides aggregate (i.e., unitary whole) or composite capabilities (e.g., a multi-sensor or

sensor-effector package, a computer, a cellphone, a satellite, an automobile, a robot, a factory, a communications network).

## Developmental Concepts

### *Products*

A ***product*** is a composite article that, applied to a designated system, induces or modifies the capabilities, properties, behavior, or information content of that system. Remotely or indirectly accessible capabilities of a product within a system may be construed as being a set of "***services***". The particulars of a product, as the medium for behavior corresponding to a coherent set of capabilities, can change over time.

A product is composed of those raw materials—articles of hardware, software, data, procedures, and supporting materials—needed for it to satisfy its intended use. Each constituent article may be fabricated by the product developer or acquired from a provider of such articles. An article may be replicated, with or without change, for use in multiple products as well as for multiple uses within a product. A product may itself be an article used in the composition of a more complex product.

A product is created with the intent of improving the degree to which an enabled system, within an associated ecosystem, supports a customer's objectives. A product may be deployed to effect changes in a single system, in alternate instances of a single system (whether operating in the same or different operational contexts), or in multiple distinct systems (e.g., having differing behaviors beyond what the product addresses). A product may be built with the means for replicated instances to communicate for sharing data and coordinating action. A device may be realized as a (software-based) product in its own right, to be replicated and deployed as elements of one or more systems.

Each built realization of a product is referred to as a "***version***" of the product. Versions may differ in observable behavior (i.e., capabilities or behavioral quality) or in the realization of the product (e.g., modified to prepare for or reduce the cost of future changes). A product "***subset***" is a version of a product that provides only a designated subset of full product capabilities. A product "***baseline***" is a version of the product that

can be used as the starting point for another version of the product or, optionally, to be deployed into operational use by a customer. All materials created, modified, or otherwise used in the realization of a product version are similarly version-managed within and across product versions. The "***lifecycle***" of a product is the set of baselined product versions that have been deployed into operational use.

### *Product Models*

A ***product model*** is a process-determined set of materials (documents, specifications, realizations, and models) by which all aspects of a product are described and realized. The product as such consists of only a subset of product model content—those materials that are "operationally useful". The set of all other materials, used in or resulting from development of a product, are "developmentally useful", needed for the understanding and sustaining of the product as the customer's needs or circumstances change. These other materials express the derivation of the product, including all relevant background and reference materials, specifications, data, development environment (including process, methods, and tools used), evaluation materials, and associated assumptions, issues, and rationale. During development, these materials are created and revised to provide current developers with a shared understanding of how the product is being developed. Upon completion, these materials provide future developers with information concerning how the product was developed, substantiation as to why the product was developed as it was, and perceived implications of any likely future changes in the product.

### *A Problem-Solution Space*

A problem-solution space represents a set of problems, each distinguished by the needs it satisfies, and a set of alternative solutions associated with each problem. Abstractly, a product is a particular realized solution to a particular corresponding problem. If a problem as understood or its preferred solution changes, a different product is indicated. This is traditionally accomplished by modifying the existing product to be a product that corresponds to the changed problem-solution. An alternative is to envision every problem-solution pair as having an associated already-realized product and

simply replacing the product corresponding to the previously identified problem-solution with the product corresponding to the reconceived problem-solution.

***The Concept of a Product Family***

In general, a *family* is a set of "related" things. By some criteria, a population of things is partitioned into two mutually exclusive sets, those that are included in the family according to that criteria and those that are excluded. The relationship of interest here is a perceived "*similarity*" in the members of the family.

This concept of a family has a consistent interpretation in a variety of disciplines:

- In biology, a family corresponds to a taxonomically-specified set of organisms that are (subjectively) classified as similar, whose characteristics distinguish them from the members of other families;
- In linguistics, a language family is a set of languages that are all derivative of a particular origin language and being similar in particular respects as a result;
- In mathematics, a (parametric) family is a set of objects that are uniformly described by a parameterized equation or function, each instance being uniquely specified by the combination of parameter values that represent how it differs from other instances of the family (in geometry, as an example, a family defines a set of curves or surfaces that all satisfy a specified characteristic equation with each instance differing in shape according to the value of equation parameters).

A ***product family*** is an expression of an envisioned set of similar products[4]. As a practical matter, a product family is a formalization and concrete realization of a bounded problem-solution space. Restating Dijkstra, products are construed as similar and therefore members of a family if they are either alternative solutions to the same problem or similar solutions to similar problems. More formally, a product family can be characterized as a subset of a candidate population, characterized by criteria that included instances must satisfy to be considered members of the family; included instances are selected as being useful for an envisioned purpose that excluded instances

[4] See Appendix, "A Mathematical Formulation of a Product Family", for an extended exploration of this and related concepts.

are not. For software-based products, the similarity considered is the behavior that each product will exhibit in operation. Each product's expected behavior can in turn be characterized in terms of properties that are sufficient to distinguish among the instances of the problem-solution space with which the family is associated.

[Parnas introduced the concept of a program family as a means to account for uncertainty, change, and diversity when developing a program[5]. He suggested three ways to realize a program family: having multiple versions of a software component for alternative implementations, associating configuration parameters with a component as a means to customize the services it provides, and enabling subsets of a component through selective exclusion of unneeded capabilities. He further noted the role of a program family as the basis for a program generator and contrasted this with the option of building an all-encompassing component whose capabilities would then be only selectively used.]

Just as instances of a family are similar according to some specified criteria, they may also differ according to other criteria by which the family can be partitioned into subfamilies. A *subfamily* includes those instances of a family that are more similar according to more restrictive criteria than the family as a whole. Such partitioning into smaller subfamilies can continue as long as useful distinctions are seen.

Consistent with the common practice of deriving new products or product versions by the modification of an existing product, an explicitly enumerated set of similar or otherwise related products that a developer has concretely realized (e.g., as instances of a product family) may be referred to as a "*product line*".

[5] D.L.Parnas, "On the Design and Development of Program Families", IEEE Transactions on Software Engineering SE-2 (1), March 1976, 1-9. <https://doi.org/10.1109/TSE.1976.233797>