1.2 Basic Terminology

The proper understanding of this text depends on understanding the terminology being used. 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. Terminology is here organized into four types: methodological, organizational, operational, and developmental.

Some definitions given here may differ from conventional usage or a reader's prior understanding but these differences are not arbitrary. They differ for any of several reasons: to expand prior unnecessarily parochial definitions of a term; 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.

Terminology, Type 1 (Methodological)

Processes, Methods, and Methodology

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 it requires to perform. A ***method*** is guidance and criteria that prescribe a systematic, repeatable technique for the performance of a given activity. A ***task*** is an assignment of resources for the performance within a specified scope of one or more related activities following a specified method or practices. 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. This text describes a methodology for the engineering and manufacture of software-based products.

Objectives and Goals

Due to ambiguity in common usage of "objective" and "goal", these are defined in terms of how they are to be understood 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,*** each identifying (consistent with quality improvement approaches) specific success criteria. Success criteria is defined in terms of appropriate metrics for measuring (objectively or subjectively) the degree to which each particular goal has been satisfied. Goals provide a concrete basis for evaluating the progress an endeavor has made toward achieving its objectives.

Models and Documents

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 an unwarranted costly 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. 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 part of a

previously consistent specialization-realization pair may establish an inconsistency that requires further change in either or both parts of the pair to reestablish consistency.

Terminology, Type 2 (Organizational)

Enterprises, Programs, and Projects

An ***enterprise*** is a managed undertaking established by one or more chartering entities (including persons, legal entities, and governmental entities) to pursue an identified purpose. An enterprise is managed by means of processes or practices that it institutes to pursue its purpose and operates by its chartering and direction of programs that will contribute toward its purpose.

A ***program*** is a managed undertaking of an enterprise intended to operate within a defined focus and scope corresponding to a specified area of institutional competence. A ***project*** is a managed effort chartered by a program to undertake actions within an allocation of its scope, objectives, and resources. A program may be the governing authority over a set of related projects.

Customers and Markets

A (potential) ***customer*** is an enterprise (person or organization) having an endeavor that could be improved in efficiency and effectiveness through the deployment and use of an appropriate product (i.e., one having capabilities conducive to such endeavors).

A ***market*** is a set of potential customers for a product having an associated purpose. A market is "coherent" if encompassed customers have similar needs, thus being amenable to accepting similar products, according to their specific needs and practices.

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 a product that has been designed to fit their specific needs and preferred practices. (By convention, the terms "customer" and "simple market" will be considered equivalent and may be used interchangeably.)

Providers and Acquirers

A ***provider*** is a program that charters projects for the development of products that support the needs of a specified, internal or external, market. A provider may also be a customer to providers of products that support its own efforts.

Broadly, an ***acquirer*** is a customer in a provider's designated market, or in a narrower sense, a project chartered by such a customer to serve as its proxy, authorized to represent customer needs and interests to providers. Ideally, an acquirer project includes staff who are knowledgeable and experienced in the customer's business, technology, and operations. An acquirer must ensure that customer views are accurately represented to providers in that providers will view the acquirer as speaking authoritatively for the represented customer.

In targeting a simple market consisting of multiple customers, a provider project may need to rely on a more indirect, less definitive determination of their customers' needs. This entails interacting with proxies who understand the endeavors of the envisioned customer community and can represent their needs to a provider. Such proxies may include experienced individual users, qualified subject matter experts, past developers of similar products, and provider marketing or sales personnel in addition to representatives of selected customers.

Terminology, Type 3 (Operational)

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 (a physical-virtual hybrid) and is characterized by properties that manifest the effects of associated naturally-occurring and entity-initiated phenomena.

An ***ecosystem*** is 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.

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 are attributed to the system as a whole.

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. ***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 or a 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 an 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, storing, 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.[1]

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 sorts of hardware: platforms and devices. A hardware platform is the physical infrastructure (i.e., physical housing and power, communications, and environmental hardware) that enables a collection of devices to coordinate activity and share data.

---

[1] 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 type of software encoding, genomic machinery enables an organism to enact DNA-specified behaviors.

A device (sometimes referred to as an "instrument" or in aggregate as "equipment" or "machinery") provides specific capabilities for detecting or producing physical phenomena in the ecosystem within which it operates. Logically, a device provides an associated set of services by which other entities can initiate its capabilities. The behavior of a device is determined by its implementation (e.g., the fidelity with which it detects reality), limited by the capabilities of the platform on which it operates (e.g., speed and latency of communicating between two devices).

A device can have any of three types of functionality: computation, data storage, and portage. The nature of a given device is determined by the relative significance of these to its essential purpose and use (along with any ancillary functionality it needs to monitor and manage its own operation). A *computational device* is hardware whose primary purpose is to effect software-defined behavior. A *data storage device* is hardware whose primary purpose is to persistently store data representing information over time about the environment and entities with which computation is concerned. A portage device receives, analyzes/filters/converts, and transmits data that corresponds to information associated with its containing environment or associated entities. An *edge device* is portage hardware whose primary purpose is to obtain information from or induce effects on elements of the ecosystem (i.e., as a "sensor" or "effector"). An *interface device* is portage hardware whose primary purpose is to provide the means for coordinating actions and sharing data among entities operating within a shared ecosystem.

A set of physically distinct devices can be packaged or interconnected by a platform to provide an aggregate or composite capability. This capability can then be expressed either as a single or multiple logical devices or as a system (e.g., a multi-sensor package, a sensor-effector package, a computer, a cellphone, a satellite, an automobile, a factory, a communications network). In this way, a logical device or system can be viewed as an integrated unit even if physically distributed.

In addition, a device may be virtualized, emulated, or simulated. A "virtualized" device is software-enabled to modify or enhance the device's physical capabilities (e.g., to

validate received data, maintain a log of activity, temporarily store transient data, transform data representations, monitor hardware behavior for errors).

An “emulated” device is software that, based on a specification of a device’s expected observable behavior, approximates that behavior. An otherwise inaccessible entity (e.g., prior to manufacture, not connected for access, or having failed in operation) may be emulated on a computational device to serve as a surrogate.

A “simulated” device is software that replicates the approximate internal operation of a hardware device. Based on a model of its behavior, a device can be simulated for the purpose of analyzing and understanding its properties and sensitivity to external influences and to compare how variations in its behavior as modeled would cause these effects to differ. This can also be a motivation for viewing an entity or some portion of the environment as analogous to a device as a basis for simulating that behavior as well.

In general, software operates on a virtualized computational device consisting in aggregate of one or more communicating computational devices, interacting with other devices (physical or emulated), all via an integrating hardware platform. A virtualized computational device includes certain software (e.g., an operating system, runtime libraries, translator, or virtual machine/container/hypervisor technology, referred to in aggregate as a “software platform”) that allows other software to be built to operate without regard to 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 its underlying computational devices and hardware platform.

Finally, a “computational platform” is an interconnected set of virtualized computational devices on which software operates. Three purposes for a computational platform can be distinguished: development, evaluation, and operations. A development platform provides the means to build a product. An operations platform 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.

Terminology, Type 4 (Developmental)

Products and Product Models

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*".

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.

The particulars of a product, as the medium for behavior corresponding to a coherent set of capabilities, can change over time. A product changes a system by adding or removing elements or modifying the behaviors of existing elements. A product views elements of a system in terms of their roles relative to the product's purpose. A product may include any hardware, software, practices, and supporting materials needed for it to operate effectively within the targeted system.

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.

A ***product model*** is a process-determined set of materials (documents, specifications, and models, both developmentally and operationally useful) by which all aspects of a product are described and realized.

Articles of hardware, executable software, data, and associated procedural guidance are the "operationally useful" raw materials from which a product is constructed. Each of these articles 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.

The set of all additional materials, used in or resulting from development of a product, are "developmentally useful" for the understanding and sustaining of the product as the customer's needs or circumstances change. These 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 defines a set of problems, each characterized by the needs it satisfies, and a set of solutions associated with each problem. A product is abstractly the realized solution to a corresponding problem. If the problem as understood or its preferred solution changes, a different product is indicated. This is traditionally accomplished by changing the form in which the product is represented. However, imagining that every problem-solution pair has an associated already-realized product,

this can instead be viewed as simply replacing the product associated with the previously identified problem-solution with the product associated with the reconceived problem-solution.

Abstraction and the Concept of a Family

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 behavior that each of an envisioned set of products will induce within an ecosystem. Restating Dijkstra, programs (i.e., software-expressed articles) 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.

This concept of a family has a consistent interpretation in other disciplines as well:

- 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).

The program family concept of both Dijkstra and Parnas[2], as a set of potential solutions to one or a set of similar problems, is a formalization of an abstract problem-solution space. Generalizing this concept, a ***product family*** is a unified representation of an

[2] 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>

envisioned set of similar products. As a practical matter, a product family is a concrete realization of a bounded problem-solution space.

A product family is more precisely specified by an associated ***abstraction***. An "effective" abstraction for this purpose is one that is expressed by a similarity predicate over properties of the instances of a candidate population (such a predicate is the denotation of an intensionally-defined set). By this means, the abstraction specifies the criteria that every member of the family must satisfy; any instance that fails to meet that criteria is excluded. Included instances are relevant for some purpose that excluded instances of the population are not. Since every member of the set satisfies this criteria, any of them can serve as an example of the abstraction[3].

All selected instances are "similar" based on the predicate but will differ from each other in other respects that may partition the family into subfamilies. A *subfamily* corresponds to a subset (e.g., of a problem-solution space), being those instances of a family that are more similar according to more restrictive criteria than the family as a whole. Any family ultimately reduces to a set of unitary subfamilies, each consisting of a single instance that is distinguishable by its abstraction-associated criteria from every other instance of the original family[4].

An extensionally-defined (i.e., explicitly enumerated) set of similar products, developed under the auspices of a particular enterprise to solve similar problems, is referred to colloquially as a "*product line*". A product line may include instances of a product family but may also include other products that are excluded by prescribed product family criteria (i.e., lacking required or including proscribed behavior).

---

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

[4] To be precise, it is sufficient to reduce to multi-instance subfamilies wherein all members of a subfamily exhibit equivalent behavior from a customer perspective (i.e., not necessarily identical behavior); in this case, a single, arbitrarily chosen instance of each subfamily can be taken as representative of all and the others can be dismissed as redundant.