Specification and Modeling Problems

- Capturing and Management of Requirements is big problem, especially when the project is large.
- There are several different domains in which ESL specifications is used and they needed different approach to specification and modeling.
- Specification should not constraint the imeplmentation but it will be good if it is somehow expressed formally.

Requirements Management and Paper Specifications

- Requirement is a:
  - condition or capability needed by a user to solve a problem or achieve an objective;
  - condition or capability that must be met or possessed by a system or a system component to satisfy a contract, standard, specification, or other formally imposed document;
  - documented representation of a condition or capability as above.
- Requirements emerge from the problem domain. They are used to describe the needs of customers.
- A requirement represents the customer's (the higher-level entity's) understanding of a need.

Requirements Management and Paper Specifications

- During product line development, a model of requirements can be constructed and used to make selections to generate a new product.
- It can be helpful to organize the model as a forest, in which the requirements are related to each other in parent–child relationships.
- A requirement that is considered to elaborate on another requirement can be made a child of that requirement.
- This reflects many requirements document structures.
- Requirements management is a process that takes care of making all requirements visible and traceable.

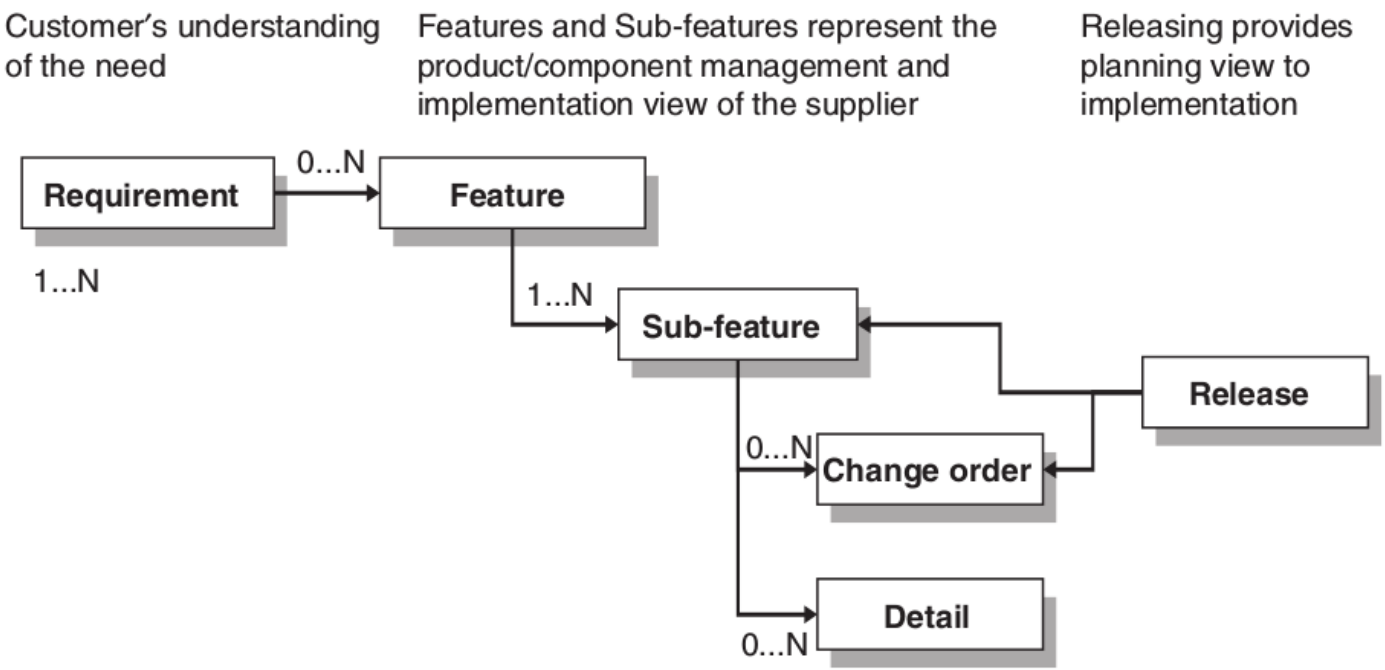
Requirements Management and Paper Specifications

- The requirements need to be tracked through the design process so that they are approved at the right level of decision making and followed up in appropriate design reviews.
- A requirement management system depends on the size and complexity of the organization, but generally placing trust only in paper documents will not suffice. Some database automation is required.
- A typical telecommunication product involves thousands of individual requirements that come from customers, standards, implementation technologies, and other sources. The nature of these requirements is diverse. One requirement may state that the product must have a specific type of voice codec, another that all voice processing is to be implemented with a specific DSP processor, and yet another specifies the microphone amplifier electrical characteristics.

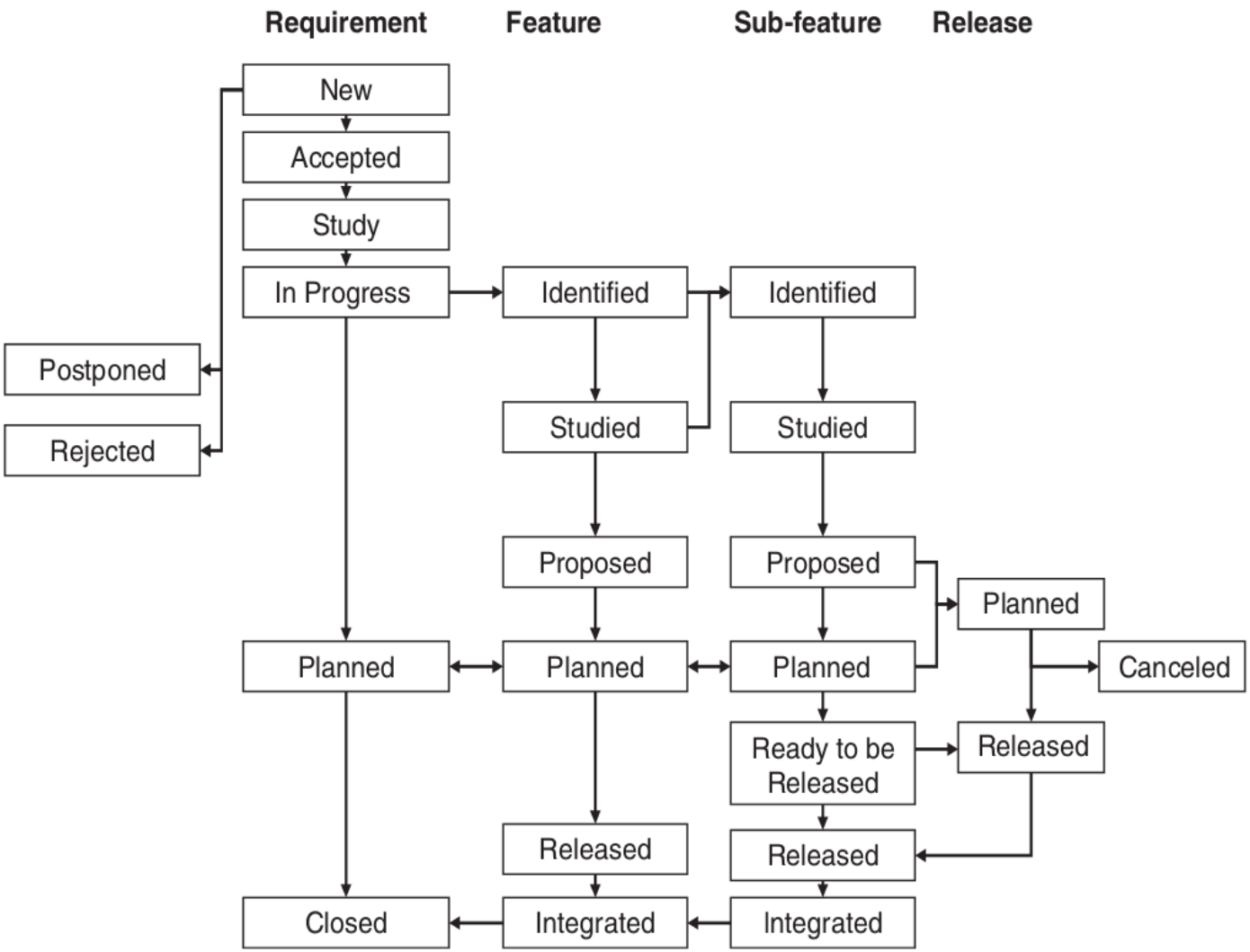
Requirements Management and Paper Specifications

- Each of these requirements is related to the same end-user functionality, but requires capturing different kinds of information from different sources.
- Each requirement affects different implementation teams or groups. Eventually, some requirements can be described formally, some can be made executable models, and some can only be in natural lan- guage.
- Although some people prefer the expressive richness of natural languages, the ability to analyze or automate operations from formal or executable models makes them the primary choice where possible.
- In addition, they are less subject to unintentional ambiguity and interpretation than natural language.
- We have also talked about the preference to describe requirements rather than executable models because these models can constrain the implementation space that can be explored.

Feature Tree



Document Status Hierarchy



ESL Domains

- ESL Domains are:
  - Dataflow and Control Flow
  - Protocol Stacks
  - Embedded Systems

ESL Domains - Dataflow and Control Flow

- Within the dataflow domain, we mean such things as modem receiver and transmitter data paths, video processing, and signal processing elements.
- Although dataflow always requires some manner of control flow, early development and analysis of algorithms that define the input-output data transformation usually ignores or abstracts away the control flow.
- In early stages of DSP algorithm development, the useful things to specify and model are static in nature, such as signal-to-noise ratio in one particular operating mode of an algorithm.
- Algorithmic performance modeling is usually done with sequential programming languages, such as C/C ++ and MATLAB.

ESL Domains - Dataflow and Control Flow

- System models that connect algorithm modules together are also sequential. Because only the flow of data is interesting from one module to another, the connections are usually modeled as infinitely deep FIFO buffers.
- Control flow is about system state and state transformations.
- Examples include protocol stacks in wireless communications and user interfaces in all manner of embedded consumer devices, as well as various subsystem control units in automo- tive applications.
- Dataflow and control flow together define a system's functionality at a high level.
- Control flow can be described at various abstraction levels, with just the main system states or modes, with and without implementation architecture and partitioning, and with various abstractions of time.
- For practical purposes, time and architecture are elemental for modeling control flow.

ESL Domains - Protocol Stack

- Protocol stacks are subsystems or communication systems that manage the flow of data on a communications channel according to the rules of a particular protocol, such as TCP/IP.
- They are called stacks because they are typically designed as a hierarchy of layers, each supporting the one above and using the one below.
- The protocol stacks usually identify a user stratum and a control stratum.
- The former deals with data channels carrying payload data and the latter with control messages that are internal to the system.
- The protocol stacks are often layered according to ISO's OSI model [ISO 1994].

ESL Domains - Protocol Stack

- Protocol stacks are implemented predominantly in embedded software, although in some wideband systems like WiMax, the media access layer may be implemented at least partially in hardware for performance reasons, and some wireless modems used with personal computers often run the higher protocol layers in the PC's processor.
- The physical layer or layer one of a communication system is logically a part of the protocol stack, but when categorizing from an ESL domain perspective, it falls under the dataflow/control flow domain.
- The common features of protocol stacks from the ESL modeling point of view are hard real-time requirements for processing and communication, very modular architectures driven by standards, communication that naturally yields to message- passing implementation, and limited needs for (instruction level or hardware) parallelism.

ESL Domains - Embedded Systems

- Embedding: Embedded systems, by their name and nature, are often parts or components of other systems. They may require the existence of other com ponents to function correctly. Typically, they also include mixed hardware and software components, real-time operating systems, special-purpose device drivers, and the like.
- Real-time properties: The events to which an embedded system reacts will have some kind of timing component. This may mean that the events have a valid lifetime before their effects become invalid. Some events, called hard real-time constraints, define that if they are not processed before a certain deadline then their effect or result becomes dangerous for the system, whereas others, called soft real-time constraints, are less critical in nature, and the "value" of the event decreases over time.

ESL Domains - Embedded Systems

- External environment interaction: The interaction with the external environment is often not based on a traditional user interface but rather on a low-level, system-specific set of commands or signals. For example, an embedded system found in the engine management unit of a car will not have a human user interface but rather sensors to various parts of the engine. Outputs may be made to some other control unit. It is possible that some instruments may be provided that allow the user to monitor certain aspects of the systems operation, or to set some high-level operating parameters of the car, such as if it should run in economy or performance mode.
- Nonfunctional constraints: Embedded systems are significantly constrained by the physical size of the device, power consumption, the aforementioned real-time constraints, and available hardware, software, development environments, and so on. These nonfunctional constraints, although existing within so-called "enterprise" systems, are not as pronounced or even relevant compared with the embedded domain.

Executable Specifications

- The executable specification is a behavioral description of a component or system object that reflects the particular function and timing of the intended design as seen from the object's interface when executed in a computer simulation.
- The primary purpose of an executable specification is to verify that the specified behavior of a design entity satisfies the system requirements when integrated with other components of the system and to verify whether an implementation of the entity is consistent with the specified behavior.
- One caveat on executable specifications should be noted. Because they are exe- cutable, executable specifications are an "implementation" as well as a manifestation of specification characteristics.

Executable Specification

- There are measurable characteristics that can be derived from the model execution or simulation; some of these are relevant to the system being specified, and some are pure artifacts of the executable model.
- The wise and experienced systems engineer or architect will know how to separate those characteristics of an executable specification model that are relevant to the functionality under analysis, from those that are artifacts of the implementation of the model.
- The executable specification in an early design phase is most likely a simulation model of the system function. By refining the model and analyzing the different architectural choices, we end up with an executable specification of a real implementation with a good match to the final physical architecture.
- Using this approach, integration starts when building the first executable specification.
- When the final integration phase happens, all functions and interfaces have been verified several times during the design process.

Some ESL Languages For Specification

- Languages used for ESL specification are:
  - MATLAB
  - SystemC
  - UML
  - XML

Future - Model Based Development