# Chapter 8
# Constrained Information Models

A central idea of the HL7 V3 approach is that of constraining or refining a general model, for the specific use case being considered, by limiting optionality. This idea of constraining a general model to create an agreed subset and interpretation of the specification is widespread in the standards world. Such constrained specifications are called profiles.

Many standards have a large number of optional aspects, and if different suppliers do not implement the same subset, they will fail to interoperate. The use of profiles is a way to enforce a particular interpretation to ensure interoperability.

The idea of constrained information models creates a tree-like hierarchy of possible models. At the root lies the RIM. Everything else is a constraint on the RIM.

## 8.1 Types of Constrained Information Model

The following types of constrained model are recognized within HL7 V3, starting with the broadest, proceeding to the narrowest.

DMIM Domain Message Information Model
RMIM Refined Message Information Model
HMD Hierarchical Message Description
MT Message Type

### *8.1.1 Domain Message Information Model*

Domain Message Information Models (DMIM) have been defined for many subject areas.

A DMIM is a general model of a domain, in HL7 notation, from which a related family of message specifications can be derived. A DMIM may be created top-down from domain experience or bottom-up as a superset of messages in a domain. Once created, a DMIM can be used as reference from which further messages may be defined.

T. Benson, *Principles of Health Interoperability HL7 and SNOMED*, HI,
DOI 10.1007/978-1-84882-803-2_8, 

A DMIM does not have a hierarchical structure and cannot be serialized, which means that it cannot be implemented as it is but needs to be further constrained as RMIMs. Not all projects use DMIMs, but their purpose is to provide a common point of reference to ensure compatibility between all RMIMs in the same domain.

### *8.1.2 Refined Message Information Model*

The most widely used constrained information model is the RMIM (Refined Message Information Model), which is a diagram of a message specification. RMIMs and DMIMs use the same notation. One important difference is that an RMIM can only have one point of entry and can be expressed in a serialized format, which is essential if a message has to be transmitted as a string of bits over a wire.

### *8.1.3 Hierarchical Message Description*

An RMIM can also be expressed in a tabular format, known as a hierarchical message description (HMD). HMDs and RMIMs can contain the same information, but most people find that RMIMs are easier to use and understand.

### *8.1.4 Message Type*

A message type (MT) is a particular specification of a message which can be used in a data interchange. Any one RMIM or HMD can be further constrained in various ways to create a set of closely related message types, which are then exchanged as a linear string of XML and validated using an XML schema.

## 8.2 Types of Constraint

The RIM, DMIMs, and RMIMs can be constrained in several different ways.

### *8.2.1 Omission and Cloning*

The simplest form of constraint is by omission. Classes or attributes with classes are simply left out. All classes and all attributes (apart from structural attributes) in the RIM are optional, so you only use the ones you need.

### 8.2.2 Cloning

The same RIM class can be used many times in different ways in DMIMs and RMIMs. This process is referred to as cloning and the classes selected for use in constrained models are referred to as clones.

The metaphor being that you take a clone of a class from the RIM and then constrain that clone in the constrained information model. Cloning limits the number of classes that need to be defined in the RIM, leading to a small stable RIM.

### 8.2.3 Multiplicity and Optionality

The next form of constraint is to constrain multiplicities in terms of repeatability and optionality. Most associations and attributes in the RIM are optional and allow any number of repeats.

These can be constrained by making such multiplicities nonrepeatable mandatory (1..1) – you need to have one, but only one; or nonrepeatable optional (0..1) – if you have any, you can only have one.

In HL7 Version 3 specifications, the correct verb form for indicating a requirement is "SHALL." The correct verb form for indicating a recommendation is "SHOULD." The correct verb form for an option is "MAY." Universally accepted standardization terminology does not recognize "must." "SHALL" is used to indicate a mandatory aspect or an aspect on which there is no option. The negatives are SHALL NOT, SHOULD NOT, MAY NOT.

### 8.2.4 Data Types Constraint

The third type of constraint involves constraining data types. The HL7 V3 data types have been designed with a hierarchical structure. For example, there are four code data types: CS (code simple), CV (coded value), CE (code with equivalents), and CD(concept descriptor) in increasing order of complexity. A more complex data type, such as CD can be constrained to a simple data type such as CV. Similarly, the data type GTS (General Time Specification) can be constrained to IVL<TS> (Time Interval) or to TS (Timestamp).

### 8.2.5 Code Binding

The final type of constraint involves code binding – specifying what code value sets shall be used. The coding strength of a code may also be restricted to CNE (Coded No Exceptions) or may be specified as CWE (Coded With Exceptions).

This may all sound quite complex but is a lot simpler than it sounds. The simple rule is that you only specify what you need, leave out everything else, or make it as simple as possible.

## 8.3 Vocabulary and Value sets

The HL7 V3 standards talk about vocabulary domains and value sets and it is important to understand the difference between them.

A value set is the set of codes that may be used to populate a specific attribute in a message instance, and is usually specified by the message designer. A value set may be a single code only, for example to specify a structural attribute, a subset of an HL7 defined code, or all or part of an externally defined coding system.

A vocabulary domain is the set of codes available to the message designer for a specific attribute. For example, the vocabulary domain for the *Act.moodCode* is the set of all *mood code* values defined and maintained by HL7.

Message users are concerned with value sets, message designers need to think about vocabulary domains and select their value sets from these.

The concept of vocabulary domains is most applicable to HL7's own internally defined vocabulary tables, which are quite extensive. These must be used for structural attributes and are widely used within Data Types.

Each concept normally has a mnemonic code, which is the code value used; a print name which explains its meaning; a concept ID, used for internal reference; a level; and type. Mnemonic codes are unique for a particular coding scheme. These tables have a hierarchical structure, with each concept being allocated a level, so a level 2 concept is the child of the preceding level 1 concept and so on. The code type may be

- Abstract (A) does not have a code, but does contain child concepts
- Specialized (S) has a code and contains child concepts
- Leaf (L) has a code, but no child concepts

## 8.4 Artifact Naming

HL7 V3 artifacts are identified using a common naming scheme, which is at first sight a bit complex. The format is SSDD_AAnnnnnnRRVV.

The first four characters identify the subsections and domains.

| | |
|---|---|
| COCT | Common Message Elements |
| COMT | Common Message Content |
| FIAB | Accounting & Billing |
| FICR | Claims & Reimbursement |
| MCAI | Message Act Infrastructure |

| | |
|---|---|
| MCCI | Message Control Infrastructure |
| MFMI | Master File Management Infrastructure |
| POLB | Laboratory |
| PORX | Pharmacy |
| PRPA | Patient Administration |
| PRPM | Personnel Management |
| PRSC | Scheduling |
| QUQI | Query Infrastructure |
| RCMR | Medical Records |

The first four characters are followed by an underscore character "_" and then the artifact type, identified with a two-character acronym.

| | |
|---|---|
| AR | Application Role |
| DM | D-MIM (Domain Message Information Model) |
| DO | Domain |
| EX | Example |
| HD | HMD (Hierarchical Message Descriptor) |
| IN | Interaction |
| MT | Message Type |
| NC | Narrative Content |
| RM | R-MIM (Refined Message Implementation Model) |
| ST | Storyboard |
| ST | Storyboard Narrative |
| TE | Trigger Event |

The artifact type is followed by a six-digit identifier allocated by the committer responsible. The final characters are a 2-character Realm Code, identifying which international affiliate of HL7 is responsible for this. The default is UV (Universal) followed by a version number in the range (00–99).

For example: PRPA_RM001234UV00 may be recognized as an RMIM in the Patient Administration, used universally, revision 00. This artifact-naming convention is a bit awkward, but it is worth taking the trouble to memorize the main acronyms.

The name of each cloned class in an RMIM is derived from its structural attributes (Fig. 8.1).

## 8.5 A Simple Example

The entry point or focal class is an ObservationEvent. This is the default name for any Act with *classCode* = OBS (observation) and *moodCode* = EVN (event). This has three other attributes: a unique identifier *id* (such as a UUID), a *code*, which states the type of report, and an *effectiveTime*, which refers to the date/time of the observation (Fig. 8.2).

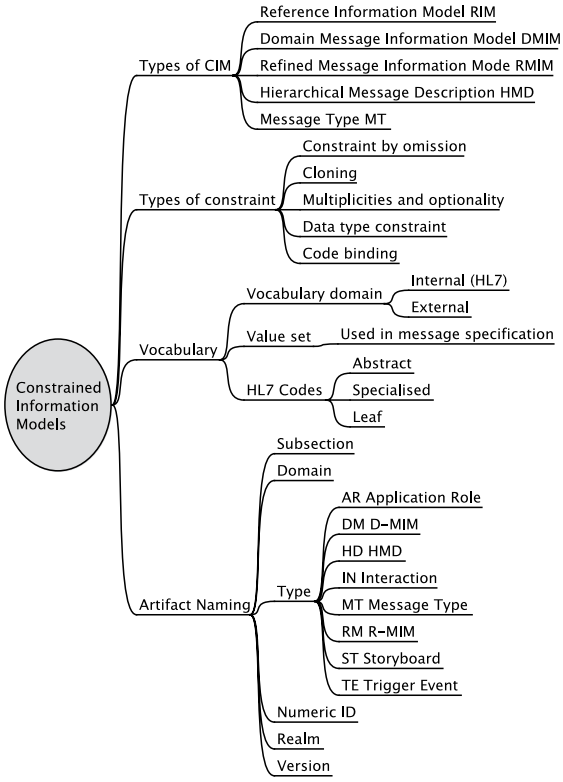


**Fig. 8.1** Constrained information models

This report contains one or more InvestigationEvents (*classCode* = INVSTG, *moodCode* = EVN), each of which has an *id* (such as a UUID or a line number), a *code* (in this case a CPT4 code to indicate what it is) and a *value*, which is a simple text string (ST).

The report has two Participations: Subject and Author. The way to read the Participations is that the ObservationEvent has subject Patient and has author Agent.

The subject is a Patient, with an *id*, such as a hospital number. The Patient is scoped by an Organization, which has an agreed identifier (*id*). The combination of the Organization *id* and the Patient *id* should be globally unique.

The patient has an optional *name*, in the Person class (Entity). The playing association (patientPerson) between Person and Patient is indicated as [0..1], and is not in bold font, indicating that this is not mandatory. Similarly the *name* attribute in Person is not in bold font and is annotated as [0..1].

The author is an Agent, which could be a clinician, technician, or a machine. The Agent has a unique identifier.

In this RMIM all elements are mandatory (and therefore required), which is why they are all written in bold font and suffixed with the "*" indicator.

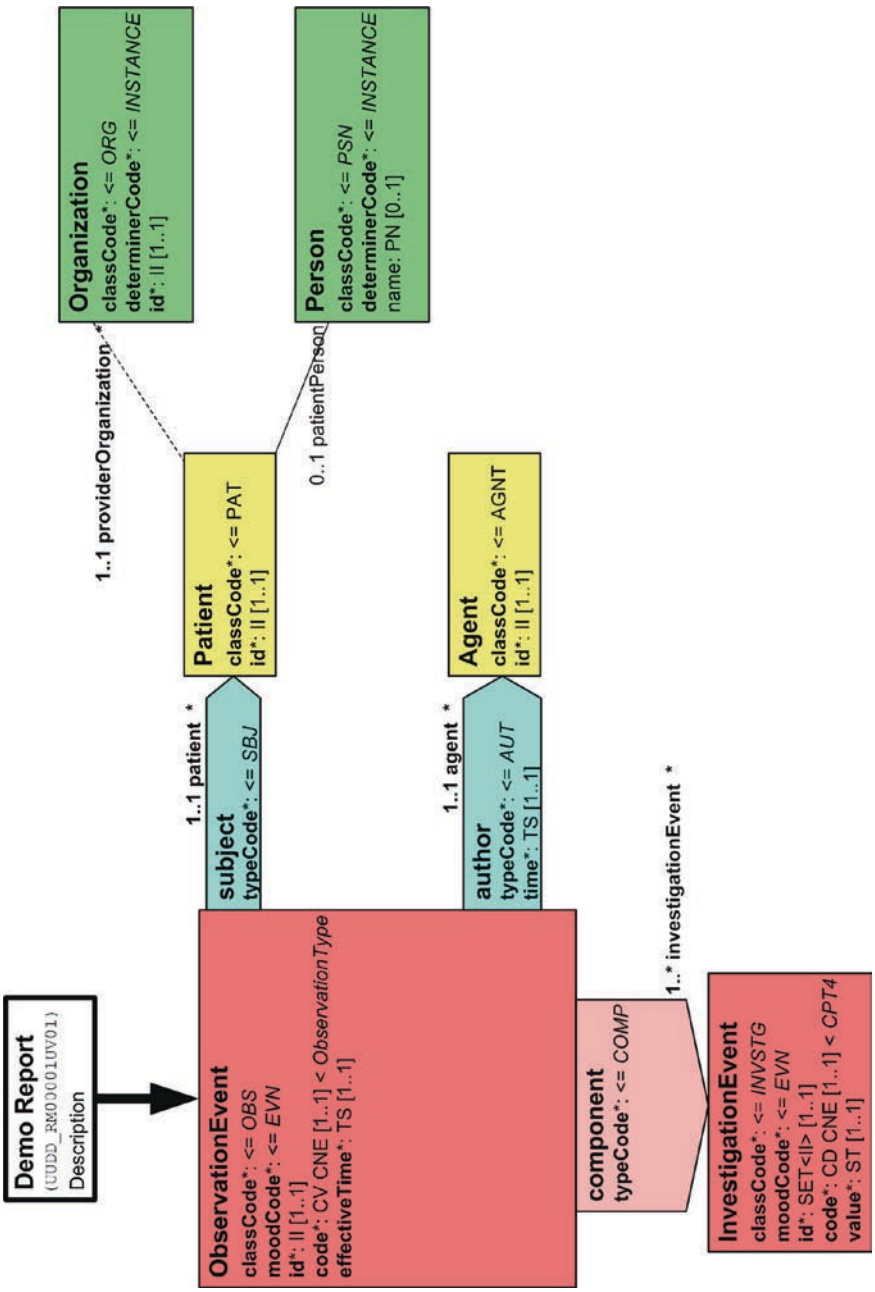


**Fig. 8.2** A simple RMIM for an investigation report

## 8.6 Diagram Notation

### *8.6.1 Classes*

HL7 uses a special graphical notation for specifying RMIMs and DMIMs (Fig. 8.3).

- Each Act is represented as a red rectangle
- Role as a yellow rectangle
- Entity as a green rectangle
- ActRelationship is usually shown as pink (salmon), arrow-shaped pentagon
- Participation as a cyan (light blue) pentagon
- RoleLink as a light yellow pentagon.

Each of the arrow-shaped pentagons has a source for the relationship and a target. The direction of the arrow indicates the meaning of the association, but this is not always the way that the diagram should be navigated.

The direction of navigation (the way you read the diagram) is indicated by the location of the multiplicity shown just outside the class. This may sound confusing, but the important thing to remember is that the direction of the arrows is not always the way that the diagram should be read.

ActRelationship and RoleLink may be recursive, that is, point back to itself. This is indicated by a "pig's ear" box with a notched out corner which fits around one corner the Act or Role.

### *8.6.2 Attributes*

Each attribute uses exactly the same attribute name as is in the RIM. The attributes selected for use in RMIMs are formed by constraining or limiting the attributes as defined in the RIM. This allows checking and validation and is the key reason why the RIM may not be changed.

- The attribute name in an RMIM diagram may be in bold print. This indicates that this attribute is mandatory, it must always be present, null values are not allowed. This is a responsibility of the sender Application Role.
- The attribute name may have a star "*" next to it. This indicates that this attribute is required to be present in messages. If data is not available a "null" value may be sent.
- The multiplicity or cardinality of the attribute is denoted within square brackets [ ] to indicate how many times this attribute may be repeated. [0..1] indicates zero or one; [1..1] indicates exactly one. "*" indicates no upper limit, so [0..*] indicates zero to many.
- The attribute's Data Type is specified after the attribute name, separated by a colon ":". The specified Data Type must be either the same as or a valid constraint on the RIM Data Type for that attribute.

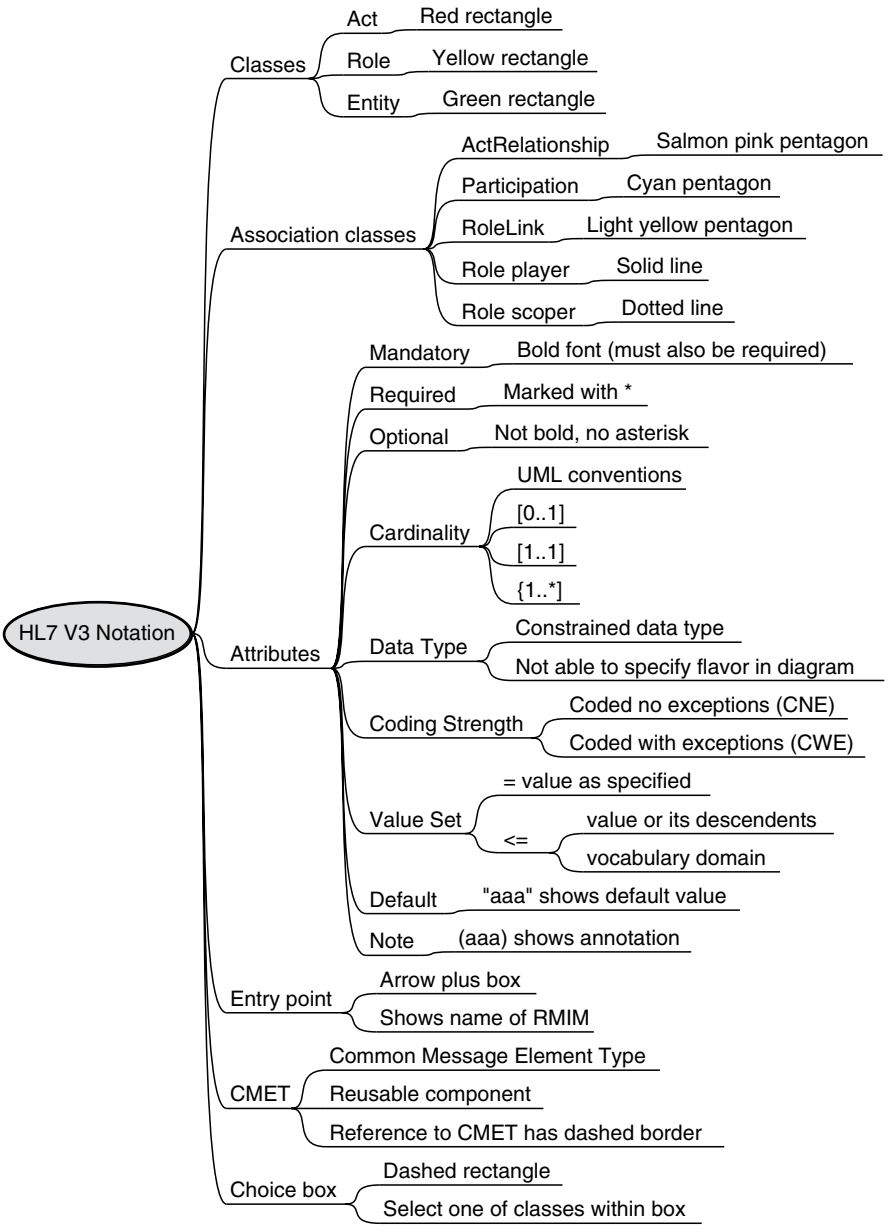


**Fig. 8.3** HL7 V3 Diagram notation

- If the data type is a code, then the coding strength may be denoted by adding either CNE (coded no exceptions) or CWE (coded with exceptions) after the data type designator.
- The value set or vocabulary domain to be used with this attribute is specified after either an "<=" or "="; "<=" indicates that the value may be taken from a vocabulary domain or the code specified or any of its descendants in a hierarchy. The equals sign indicates that the value should be as specified. The domain specification must be either a domain name defined in the vocabulary tables, or a single code value from the appropriate domain.
- A string in quotes (e.g., "string") indicates a default value for this attribute.
- Finally, a brief description of attributes may be included, enclosed within parentheses (..).

If the attribute information extends beyond one line, then second and subsequent lines are indented.

### *8.6.3 Entry Point*

Every RMIM has an entry point, which points to the focal class. This also states its name, identifier, and any descriptive notes the author has provided.

### *8.6.4 CMET*

Common Message Element Types (CMET) is a reusable module, which can be used in multiple messages, rather like a program subroutine. Using CMETs can speed up the process of developing messages and increase consistency between different specifications.

Each CMET has two parts. The CMET reference is a special class, which can be added to an RMIM. When a CMET is referenced, or used in another diagram, it is shown with a special notation, a box with dashed edges. It contains the name of the CMET, its artifact *id*, its *class code*, and its level of attribution. It is color-coded in a manner consistent with its root class. Each CMET has a unique artifact identifier (beginning with COCT_), which is the primary link between each CMET reference and its content.

The CMET content itself is defined as a small RMIM, which is stored in a CMET library. This is included automatically in messages when they are constructed.

Each CMET has a single entry point, which is the point at which it is attached to any containing message, which references it. CMETs do not have exit points, which means they have to be at the terminal or leaf point in the hierarchical structure of a message.

Because CMETs are designed for common use by any HL7 committee they are kept in a separate common components library.

### *8.6.5 Choice Box*

HL7 RMIMs show choices or options by using a "choice box." Each of the options is shown in a box with a dashed line border, from which a single choice is made.

Associations may be made to a specific class within the choice box, or to any of the classes, irrespective of which is selected.

## 8.7 Tooling

RMIMs are built using a special tool set developed by HL7. The original tools, based on Microsoft Access and Visio are in the process of being replaced by a new generation of tools, which use a slightly modified notation.

The basis of these tools is a set of interrelated XML schema, known as Model Interchange Format (MIF). MIF defines the primary artifacts that can be developed or exchanged as a result of HL7 V3 standards development and implementation.

## 8.8 Templates

In HL7, templates are used to constrain and verify conformance to profiled HL7 Version 3 Refined Message Information Models (RMIMs). A template is an expression of a set of constraints on the RIM, which is used to apply additional constraints to a portion of an instance of data expressed in terms of some other Static Model. Templates are used to further define and refine these existing models within a narrower and more focused scope.

Each template is identified with a *templateId*, a globally unique identifier.

## 8.9 HL7 Development Framework

HL7's standard development methodology is documented in the HL7 Development Framework (HDF)[1]. The HDF is written for HL7 members who are developing standards within HL7 committees. However, much of what it says is of universal relevance. The HDF adopts a project-oriented approach, based on a Product Life Cycle for Product Development.

[1] HL7 Development Framework. Version 1.3, 2009

The HDF identifies the following major stages (Fig. 8.4):

- Project Initiation Process
- Domain Analysis Process
- Specification Design Process
- Standard Profiling Process
- Technology Specification Process

### *8.9.1 Project Initiation*

The Project Initiation Process (PIP) includes initiation, planning, and approval substages, including the development of a detailed Project Scope Statement (PSS) and plan. The project plan identifies the business case and objectives,

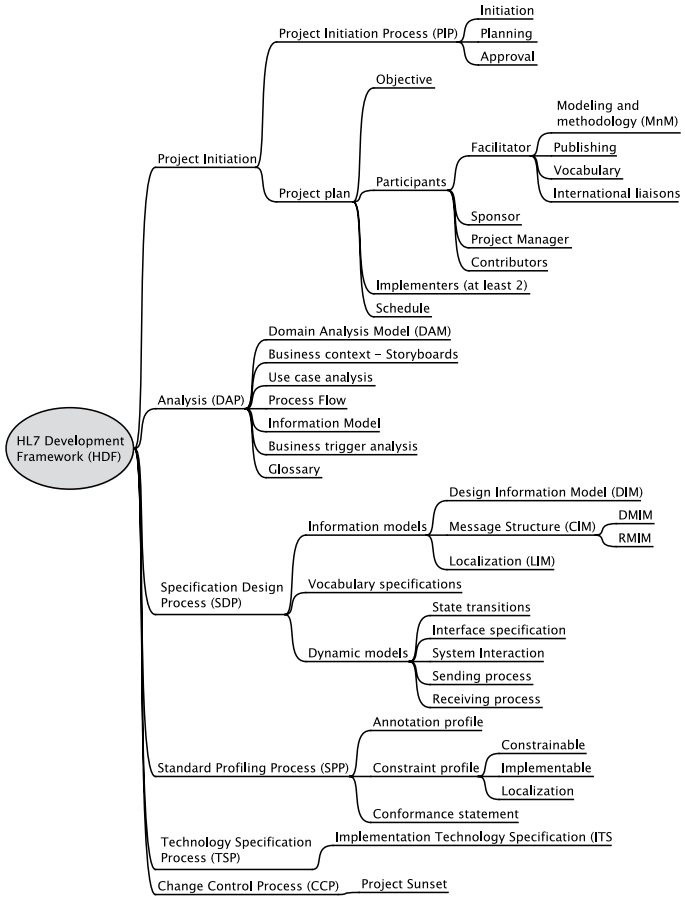


**Fig. 8.4** HL7 Development framework

participants including sponsor committee, project leader, contributors and early implementers, and a time schedule.

### *8.9.2 Domain Analysis*

Domain Analysis Process (DAP) includes analysis and requirements documentation, including the development of a Domain Analysis Model (DAM), which includes:

- Business context including documentation using storyboards and identification of relevant actors and interactions
- Use case analysis documenting use cases and actors
- Process model, documented using activity diagrams
- Information model, documented using classes and attributes
- Business rules including trigger events
- Glossary

### *8.9.3 Specification Design*

Specification Design Process (SDP) is the core of the process. It involves mapping the requirements as set out in the Domain Analysis Model to the HL7 RIM, data types and vocabulary to specify the message structures, value sets, and dynamic processes.

### *8.9.4 Standard Profiles*

A profile is a set of information used to document system requirements or capabilities from an information-exchange perspective and is expressed in terms of constraints, extensions, or other alterations to a referenced standard or another profile. Profiles of HL7 Version 3 are derived from a Version 3 specification, as balloted either by HL7 or by one of its affiliates.

The categories and use of profiles include annotation, constraint, localization implementable, and conformance profiles.

Annotation profiles document the standard exactly but with more information to further explain the base document to educate prospective users and/or implementers.

Constraint profiles may contain unchanged and constrained elements, reducing the optionality and cardinality of the base specification (i.e., the HL7 V3 standard) in order to make the specification more exact.

Localization profiles meet the same objectives as a constraint profile, with the addition of some additional elements (extensions). HL7 Version 3 allows localization of some parts of the standard but not others. In particular, HL7 does not allow anyone, apart from HL7 itself through a formal process, to change or modify the RIM or any of the Data Types. Localization can make full use of the constraint mechanisms and make certain changes to RMIMs, Data Types, Message Types, CMETs, and Vocabularies.

Implementable profiles are the most constrained constraint profiles and eliminate all optionality in the base specification (the HL7 V3 standard) in order to make the specification exact and approach "plug-and-play" interoperability. Optionality for a profile is eliminated when the conformance indicator for every attribute and association is either Required or Not Permitted and every vocabulary domain is bound to a value set.

Conformance statements set out a computer system's conformance claim to a set of interactions. A conformance profile indicates the set of interactions that a computer system (or application role) supports. It implies a commitment to fulfill all of the responsibilities of the interactions specified and to implement faithfully the artifacts that constitute the interactions and any further constraints or extensions.

## 8.10 Implementation Technology Specification (ITS)

The XML implementation technology specification describes how individual instances of message types shall be rendered in XML for serial transmission over the network and the structure of schemas used to validate each instance. Note that the HL7 generated XML schemas are not able to test all of the constraints defined in a HL7 message definition.

The generation of the schemas and message representation is done automatically. Those not directly involved in that process do not need to understand the technical details. The key points are as follows:

One XML element is defined to correspond to each row in the HMD grid, with the exception of structural attributes which are expressed as XML attributes.

Each Data Type has a defined XML representation. The "restriction base" feature in an XML schema is used extensively to define how data types are implemented.

The schema files for CMETs are supplied separately and then used by each message schema as required.

The XML schemas defined support V3 Data Types, and Data Type refinement, through the use of the W3C Schema restriction element. Additional standard schema sections support RIM classes and the HL7-defined vocabulary definitions. These schema sections can be selectively combined with a specific message schema through the "include" function in the XML schema standard.

HL7 messages all share the same XML namespace.

Message version information is conveyed as attributes within the message rather than by changes to the namespace identifier.

## 8.11 Documentation

HL7 V3 documentation is voluminous. The full set of HL7 V3 standards is published annually and can be downloaded from the HL7 web-site (www.hl7.org).

Foundation documents are the basis of the standard. The Foundation documents in 2008 contain 359,000 words (about 30 h reading at 200 words per minute. In addition there are 30 or more domain-specific standards (Fig. 8.5):

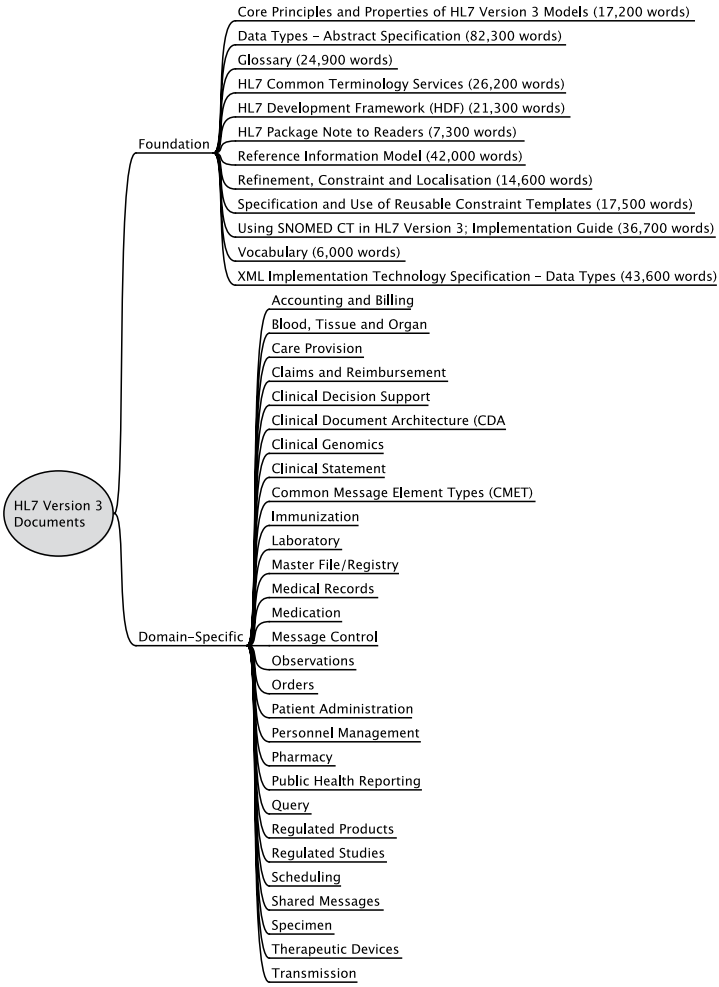


**Fig. 8.5** HL7 Version 3 documentation