Chapter 3
Models

A deep understanding of what it means to create or apply a model underpins the way we interact with the world and how likely we are to be successful in achieving our goals. Models define the way we learn about the world, interpret what we see, and apply our knowledge to affect change, whether that is through our own actions or through the use of technology like a computer.

Enrico Coiera (Coiera 2003)

3.1 Modeling Basics

Models play a central role in interoperability, and a clear understanding of modeling is an important foundation skill. Most people can learn to understand models quite easily, but it is much harder to create good models.

Models are either representations of aspects of interest in the real world, such as maps, or specifications for things that have not yet been built, such as blueprints used by architects and engineers. Each model is a simplified representation of aspects of the real world or the world we wish to create.

For example, an architect produces hundreds of different drawings or diagrams when designing a building – each diagram having a specific purpose and relating to a single project. Any single type of diagram shows only certain aspects of a situation – everything else is ignored. This simplification provides both the power (it makes the situation understandable) and the weakness of diagrams (each diagram has a limited scope and some things are left out).

If a single model is used to document each major stage of a project, then each diagram is a single view of that model, ensuring that all parts are coherent. Each component is recorded only once, irrespective of how many diagrams it is used in. This makes it easier to make changes and to ensure that the whole model and its diagrams retain coherence and consistency.

T. Benson, *Principles of Health Interoperability HL7 and SNOMED*, HI,
DOI 10.1007/978-1-84882-803-2_3, © Springer-Verlag London Limited 2010

3.1.1 Model Driven Architecture

In addition to models of the real world (AS IS) and specifications (TO BE), we can build models at different levels of abstraction. The Object Management Group (OMG), which is the group responsible for the major information modeling standards, has defined Model Driven Architecture (MDA) as a framework for the development of object-oriented software (Mellor et al. 2004). MDA uses four types of model, with mappings between them:

- Computational-Independent Model (CIM)
- Platform-Independent Model (PIM), which describes the conceptual design of a system
- Platform-Specific Model (PSM), which specifies the implementable design
- Code, the actual software code written, sometimes referred to as "wire-format"

A key feature of MDA is formal mapping between each of the deliverables: CIM to PIM, PIM to PSM, and PSM to Code. This provides traceability between each stage in the process.

In healthcare, Mead has argued that computable semantic interoperability needs four pillars (Mead 2006). These are:

- A common model across all domains of interest
- This model to be grounded on robust data type specification
- Methodology for binding elements of the model with terms from concept-based terminologies
- A formally defined process for defining specific structures to be exchanged between machines – the data interchange standard itself

3.1.2 The CEN Model

The European working group responsible for healthcare communications and message standards (CEN TC251 WG3) was one of the first groups to adopt this approach.[1] The story of how it came about is worth recounting. The problem was how to choose an interchange format (syntax) to be used in European interoperability standards. At that time, there were plenty of candidates. This was a time of "syntax wars," with a range of competing standards and observers such as Tanenbaum could quip: "the nice thing about standards is that there are so many of them to choose from."[2]

A project team was established to investigate the problem. First, it created a small set of technology-neutral general message specifications (GMD), using an object model similar to UML. Secondly, the team tested whether each GMD could

[1] The author was the founder convenor of CEN TC251 WG3 from 1991 to 1997, when the work described here was performed

[2] Tanenbaum A, *Computer Networks Second Edition*. http://en.wikiquote.org/wiki/Andrew_S._Tanenbaum

be implemented in each of five target interchange languages, by developing an implementable message specification (IMS) in each target interchange language (CEN Report 1993).

The main conclusion of the study was that whilst none was ideal, each interchange format was adequate for current needs and that the "selection of an interchange format was not the largest obstacle for realizing message exchange in healthcare."

While doing this, the project team recognized the power and value of the GMDs. They proposed, and CEN TC251 accepted, that the specification of syntax-independent specifications (GMDs) should be the core of message standardization. These could then be implemented in any syntax of choice, such as EDIFACT or HL7.

The approach was further developed over the following years to identify three main types of specification (CEN Report 1996):

- A single domain-wide model, which acts as a reference for all others. This is now referred to as a reference model.
- Technology-independent message specifications, each of which is a constraint on the domain wide model. These specifications may have various levels of granularity and scope from the broad and general to the narrow and stringent.
- Implementable message specifications, which are direct mappings from the technology-independent message specifications into the selected syntax.

These ideas were also adopted and modified by HL7 in the development of HL7 Version 3 (Beeler 1998) and are also found in ISO 13606.

3.1.3 Modeling Maturity

Human beings find it easier to work with formal graphical models and diagrams than with unstructured narrative. For example, if you wish to describe a place to someone else, it can be done in various ways:

- A face-to-face conversation
- A text narrative
- A structured narrative with headings and sections
- Ad hoc diagrams and pictures
- A formal map using standard conventions

It needs little imagination to recognize that these are listed in increasing order of preciseness and ability to convey meaning accurately and reliably. The same applies to specifications, which is why engineers use formal blueprints to specify how a machine is to be built. Computer engineers and analysts also use models.

Kleppe and Warmer (Warmer and Kleppe 2003) classify the role of modeling in a project using six Modeling Maturity Levels (0–5).

- Level 0: No specification; the specification of software is not written down. It is kept in the minds of the developers. At this level we find conflicting views between developers and users and it is impossible to understand the code if coders leave (and they always do, sooner or later).

- Level 1: Textual specification; the software is specified by a natural language text (e.g., English), written down in one or more documents. Such specifications are usually ambiguous, because natural language is ambiguous; it is impossible to keep this type of specification up-to-date when code is changed.
- Level 2: Text with models; a textual specification is enhanced with several models to show some of the main structures of the system. This is easier to understand, but still difficult to maintain.
- Level 3: Models with text; the specification of software is written down in one or more models. In addition to these models, natural language text is used to explain details, the background, and the motivation of the models, but the core of the specifications lies in the models.
- Level 4: Precise models; the specification of the software is written down in one or more models. Natural language can still be used to explain the background and motivation of the models, but it takes on the same role as comments in source code. At this level, coders do not make business decisions and incremental development is facilitated by direct transformation from model to code.
- Level 5: Models only; the models are precise and detailed enough to allow complete code generation. The code generators at this level have become as trustworthy as compilers; therefore, no developer needs to even look at the generated code (Fig. 3.1).

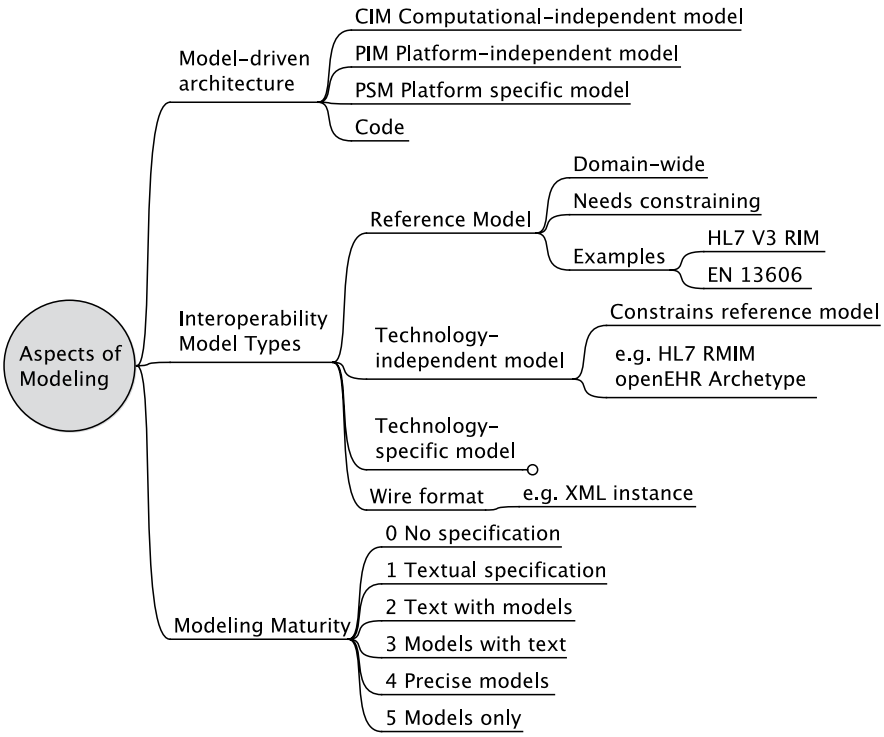

Fig. 3.1 Modeling foundation concepts

3.2 Life Cycle

The life cycles of all software development projects, including interoperability projects are broadly similar. The stages include:

- Scope and objectives
- Process analysis and design
- Conceptual design specification
- Technology specific specification
- Coding
- Testing
- Deployment, including user education, data migration, and installation
- Support and maintenance

The first step is to understand the domain and the opportunities for improvement. This phase articulates the problems which need to be solved and the benefits that can be achieved.

The next step is to refine the scope and define the detail needed to deliver the benefits in a logical, platform-independent way. At this stage a range of alternative implementation platforms may be considered. In MDA terms, this defines the platform-independent model (PIM).

Only when all stakeholders have agreed and accepted the PIM, should we start to work on the platform specific implementation. Interoperability requires a shared understanding of what we are trying to do, shared between all interested stakeholders including users, such as clinicians, sponsors, and software developers.

This is difficult. It is a challenge to achieve a shared understanding on a human-to-human level between one user and one developer, but the challenge is enormously greater as the number of parties grows. An underlying assumption of computable semantic interoperability is that both sender and receiver share the same understanding of what each and every data element means (Fig. 3.2).

3.3 Preliminary Business Analysis

The preliminary business analysis may have four main deliverables, which may form a single report:

- Scope statement provides the big picture of what it is all for, including the case for action, objectives, and scope delineation.
- Story-boards provide an intimate view of how individuals use the system to obtain value.
- Requirements description provides a structured description of the most important aspects of the system using activity diagrams and class diagrams.
- Glossary and value sets.

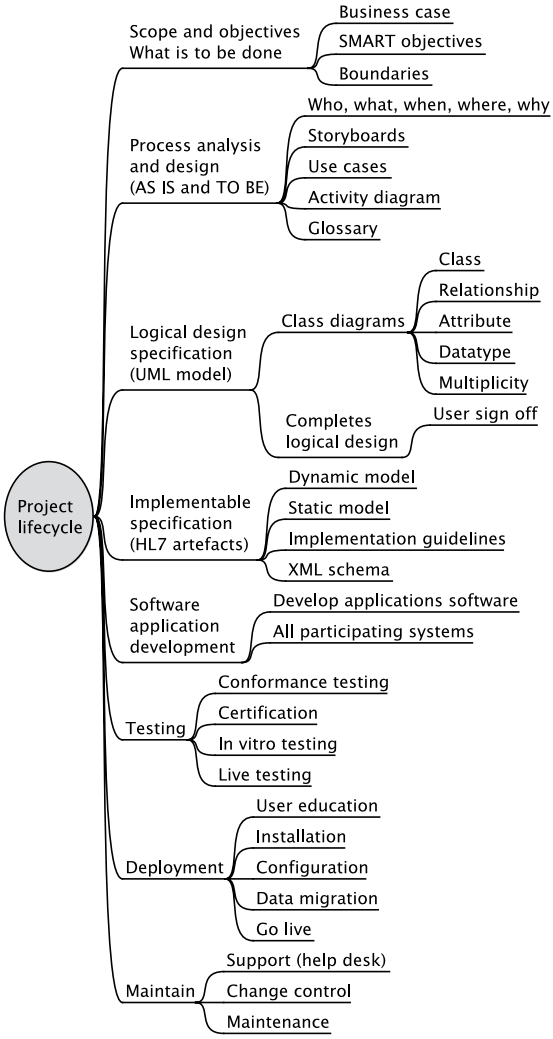

Fig. 3.2 Project life cycle

3.3.1 Scope

The scope statement provides a management summary of the whole project. It summarizes why the project is needed, what it should achieve, and what is excluded, showing the boundaries and responsibilities of the system of interest. Many problems with both systems and stakeholders occur when developers set the boundaries and/or responsibilities differently from those expected by the users.

Any change in scope must be incorporated into the scope statement, to prevent scope creep. However, changes in scope are an inevitable part of most projects, as more is understood about the domain and the user needs. It can be useful to set out

the Scope Statement under the three subheadings: the case for action, objectives, and scope delineation.

- The case for action is a concise, comprehensive, and compelling case for why the project is needed. It describes the context, the problem, user needs, and the consequences of doing nothing.
- The objectives provide a clearly stated focus of what this particular project is about. These should be SMART goals:
- Specific, clearly stating what you want to achieve
- Measurable, with a means of measuring whether or not you are achieving the objectives
- Attainable within the available resource constraints
- Relevant to the organization's needs
- Time-bounded, with clearly dated milestones
- Scope delineation clearly shows the boundary of the project – what is in or out of scope. This may also include constraints such as standards and other work that must be used, as well as functionality that are explicitly outside the scope.

3.3.2 Storyboard

Storyboards provide a useful means of capturing domain knowledge, providing specific detail, as opposed to the high level of the scope statement. They provide examples of the participants involved, the information flow, real-world situations where the services may be used and provide an informative overview, which assists in the whole development process. Storyboards may also provide a starting point for the development of test data.

Several different storyboards should be developed to cover each of the ways the system may be used.

Each storyboard is a story, told in the present tense, describing in detail how a set of named actors use the system to carry out a single instance of a task. Storyboards do not contain options. If there are two ways of doing something, then two storyboards are needed, one for each option.

Storyboards describe situations that represent either typical or extreme cases. They provide useful context that everyone can understand and also provide a starting point for developing test data.

Each storyboard can be written by a domain expert, checked by a business analyst, revised, and then discussed in a group to ensure that it captures the process accurately. Storyboards of the AS IS situation should not change, but TO BE storyboards may well need to be updated as the design of the system evolves.

Storyboards generate documentation that is useful at all stages of the development. For example, storyboards illustrate the participants involved, indicate variances in information flow, describe real-world situations where the services may be used and provide an informative overview.

An example of a story board for breast cancer triple assessment is the following:

> Jane Sharp attends the One Stop Breast Clinic, having been referred urgently by her GP after noticing a lump in her breast. She sees Dr Lee who takes her history (presenting symptoms, appropriate medical, and family history) and performs a physical examination. Jane then proceeds to mammography, where a fine needle aspirate (FNA) is also collected. The mammograph is reported by a radiologist and a pathologist reports the FNA. Jane is asked to return later to hear the result of these tests. On her return she is relieved to find out that the results are negative.

The development of storyboards provides an opportunity to gather information concerning representative forms and information sets currently being used; the functionality of existing computer applications, including details of data dictionaries; relevant standards and specifications from national and international bodies; regulatory constraints, including security and data protection requirements; and future developments and the potential to simplify and rationalize communications.

3.3.3 Business Analysis

Clear understanding of the business information flow is critical; mistakes made here affect everything else. Failure to fully nail down and specify the exact information flows now and in the future system is one of the most common causes of systems failure.

It is useful to prepare two documents; a description of the present system (AS IS) and a separate description of the proposed system (TO BE). The AS IS description is an accurate description of what happens now and can be easily checked by existing users as to whether it is right or wrong. The TO BE system does not yet exist, so it is much harder to check. Some people find it useful to visualize and test out how imaginary systems will work using storyboards (see below).

The business analysis is best done as collaboration between one or more business analysts and domain experts, such as users. The requirements are not frozen finally, but continue to evolve, being updated as more is learnt and understood about the domain.

There is no one right or wrong way to understand and elucidate the business processes. Each experienced analyst uses his or her own approach. One approach I have used and found useful to capture key aspects of the business processes is to use structured narrative descriptions, under the headings of service overview, transactions, participants, locations, identification, evidence, transaction outcome, and rules:

1. *Service Overview* elaborates the domain scope, describes each service provided and expected to be of value. A service typically represents the outermost use case of interest. Each service can be decomposed into sub-services and transactions

(see below). The broad scope of the service needs to be considered to ensure that the whole system is developed in a joined-up way, avoiding silos.

2. Each of the main *Transactions* (when) is described, together with an indication of its timing, origin, trigger event and pre-conditions, destination, purpose, volume, and outcomes. Transactions are the main use cases of interest. Each transaction is typically an exchange of information with a common set of participants, requiring evidence and generating some outcome(s); each transaction achieves some useful goal for the primary actor and has a trigger event and pre-conditions.
3. *Participants* (who) are the parties, things, and systems that are involved and how these relate to one another. Participants may be physical such as people, things (e.g., specimens), machines (e.g., computer systems), or abstract such as organizations. They are the things about which information is recorded, whether they are active participants in transactions or third parties.
4. *Locations* (where) are physical or virtual places associated with a service or transaction, such as where things take place or the origin and destinations of data. They may be physical or virtual (on a computer network). It is particularly important to specify exactly where transactions take place when considering the differences between the AS IS and TO BE models.
5. *Identification* (what) of participants, locations, and information objects is crucial, because computers systems need unique identifiers. We need to identify what identifiers are used, who or what assigns them, and what information may be accessible as a result of knowing this identifier, including any legal restrictions in using it outside of the main purpose. Many organizations assign their own identifiers (e.g., NHS number), but there are important legal restrictions in using these outside of their main purpose. Furthermore, significant numbers of individuals do not know their assigned identifier or do not have one. Soft identifiers, such as name, address, date of birth, and gender may be needed to match up individual people. These are "soft," because people can change their names and addresses, give false dates of birth, and even change gender. An identifier may refer to an individual instance (such as a person or the serial number on a machine) or to a category of things (such as the bar codes printed on packets of corn flakes).

    (a) Computer systems need unique identifiers. In every context, we need to identify what identifiers are used, how the assigner is identified, and what information is accessible as a result of knowing this identifier. For example, we may need to know what data elements are likely to be made available to each user and what mechanism is in place to ensure that all users share the same meaning – for example, do they have access to the same reference database?
    (b) *Evidence* is the information needed to support each transaction, which needs to be known prior to the transaction being triggered; it may be obtained either by direct data input or by querying a database. Evidence is usually associated with one of the participants in the Interaction.
    (c) *Outcome* describes the possible results of each transaction, including the post-conditions and responsibilities of each participant. For example, updates to records and letter generation would be described here.

(d) *Rules* include the regulations and constraints governing the transactions, including nonfunctional requirements, security, and privacy. Rules may be legislative or policy, logical, procedural, or temporal. A Rule can be free-standing or part of a multilevel hierarchy of decision-making criteria. Rule documentation should include all rules, regulations, error handling, reference data, and coding schemes related to the Transaction, which have not been documented elsewhere. Rules are often important in determining how to handle failures and errors.

3.3.4 Glossary

A glossary of terms used is another important deliverable. It contains the name of each term, description/definition, and source (if obtained from another reference). Whenever terms are used they should always have the meaning specified in the glossary. The glossary may be populated from material such as: forms and information sets currently being exchanged, the functionality of existing computer applications in this domain, including data dictionaries and relevant national and international standards.

This process described above is iterative, with continual feedback between work on finalizing the scope statement, business process analysis, storyboards, and glossary development. This stage may be fairly short (in comparison with the total project), but it is most important. Mistakes made at this stage can be expensive to correct later.

These early deliverables should not be "frozen" but need to be reviewed regularly throughout the project and updated and necessary.

Business processes can be described using use case descriptions, activity diagrams, and sequence diagrams. The static structure is described using class diagrams. These tools are described in Chapter 4 (Fig. 3.3).

3.4 Conceptual Design

Related conceptual specifications for different use cases should be modeled as views into a larger conceptual model. A single conceptual model may contain any number of consistent specifications, each tailored for a different use case. The conceptual specification should be part of a contract for the technical work to be done later.

The conceptual design specification specifies the detailed design of each part of the system in a way that is technologically neutral and that both users and technical staff can understand, check, and sign off. This may comprise a set of UML (Unified Modeling Language) class diagrams.

Each conceptual design specification should meet the following criteria:

Fig. 3.3 Preliminary business analysis

3.4.1 Criteria

The following criteria (the 8 Cs) are proposed:

1. *Comprehensive*. The model should include all high-volume transactions within its scope, being sufficiently expressive and general to fully represent and describe each at different levels of sophistication. It needs to be extensible to incorporate new requirements and local needs.
2. *Context explicit.* The model should describe not only the static structure of transaction payloads, but also the business processes surrounding each transaction, specifying trigger events, delays, timing and other constraints, business rules, outcomes, and error handling.
3. *Complete in itself*. The model should represent both data structure and processes (dynamic behavior). The sequence of activities must be clearly indicated (e.g., whether the order in which tasks are performed is significant or not). The model

should be documented internally so it stands alone without need to reference other documents or manuals.

4. *Consistent*. Each term used should be defined unambiguously and the source of each definition referenced. The same concept should only have one name, avoiding synonymy, and the same term should not ever be reused for a different concept, to avoid homonyms. A common architecture, notation, and terminology should be used to define each and every element of the model. Any problem may be divided into several sub-problems, whose solution may then be pursued separately and can be expanded or consolidated.
5. *Compatible*. Data element definitions should be compatible with international standards. Platform-independent models facilitate transfer across institutions and allow different end-user suppliers to implement compatible solutions using different proprietary technologies and to migrate to new technologies in the future.
6. *Composable*. Parts may be reused and combined freely with each other to produce new ones, possibly for purposes quite different from that for which they were originally designed. There should be minimum dependency between parts, so that any change or error in one is not propagated to others.
7. *Comprehensible*. Understandable, so that each part can be understood and reviewed on its own by clinical end users, domain experts, and implementers. A simple graphical notation is ideal, which ought to be easy to learn and use. Names and definitions should be written in the language of the user. Abstract terms and neologisms (words or phrases with newly coined meaning) should be avoided. Each element needs to be able to be understood separately by a human reader without need to consult external reference manuals.
8. *Conformance-testable*. Messages based on the model need to be tested against the model to demonstrate conformance. As few alternative methods as possible should be provided for doing any business task. Navigation across associations should be in one direction only. Recursive structures should be avoided as much as possible. Many-to-many relationships should be avoided and zero-to-many multiplicities used as little as possible. Basis for test data (Fig. 3.4).

Software engineers use technology-specific specifications to design the software that creates and reads the actual messages sent. If there is any doubt as to the meaning of any part of the specification then they need to consult the whole specification (both the technology-specific and conceptual parts). The conceptual specification should be regarded as the ultimate authority, because this is what the domain experts can understand and approve.

The mapping from a conceptual specification to a technology-specific interchange language should not permit any changes in the semantic content either by addition or constraint. The technology-specific specification indicates the “wire format” that is implemented, tested, deployed, and supported.

A conceptual specification supplements the traditional technology-specific specification and can be understood by all stakeholders. The full specification should

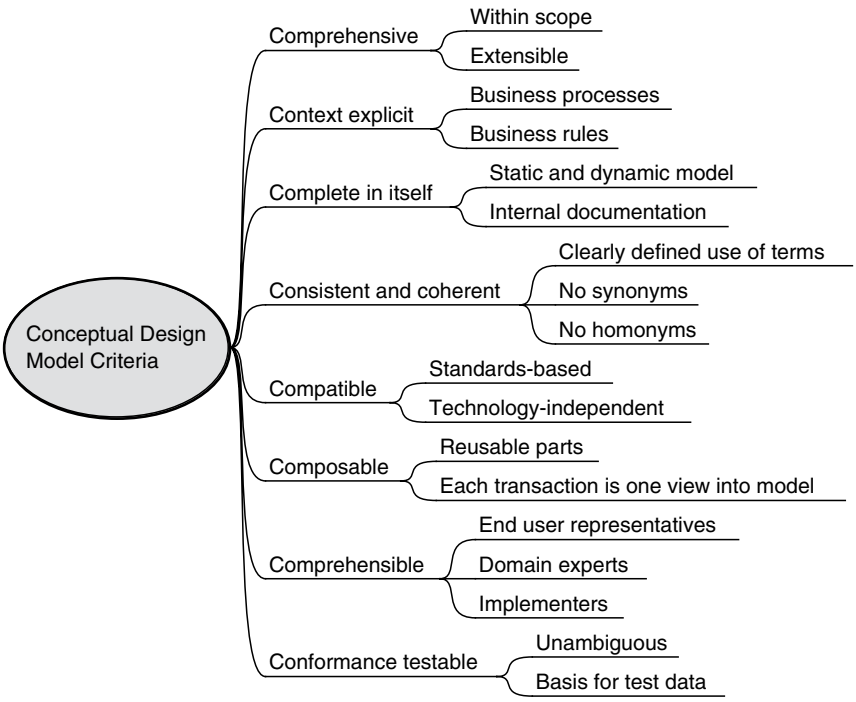

Fig. 3.4 Conceptual design model criteria

comprise both the conceptual specification and the technology-specific specification (Benson 2007).

It can be part of a contract for the technical work to be done later.

The conceptual design specification is the detailed conceptual design of what is to be provided. It is not a model of the real world or of an existing system and does not usually cover the nonfunctional requirements of the system. Being technologically neutral, it does not specify what software shall be used.

Each use case can be modeled as one view into a larger model. A single UML model can support a large number of different use cases, and the conceptual design specifications can be output in several formats including diagrams, hyperlinked documents, and as XML Metadata Interchange (XMI), which is a standard used to pass model data as XML between UML tools and software generation tools.

The crucial point about the conceptual design specification is that it is the most detailed specification that users should, with help, be able to understand, criticize, review, and sign off. It can be the basis of a contract between users and the developers. All stakeholders need to share and agree the conceptual design specification, but the technology-specific specification may be private to each developer.

3.4.2 Technology-Specific Design

The technology-specific specification sets out exactly what is to be implemented in the chosen interchange language (e.g., HL7 Version 3), including implementation guidelines and XML schema if required. This is the specification of what is to be built, tested, deployed, and supported. Technologies change and evolve much faster than the conceptual design specification. The same conceptual design can be implemented in many different ways. The mapping from the conceptual design specification to any specific implementation should not involve any changes, either by addition or constraint, in the semantic content.

The objective of business analysis is to capture the complete requirements in a form that is fully understood by users and technical staff. This feeds into the conceptual design, platform-specific specifications, and ultimately forms the basis for testing and conformance. The development of the Business Analysis is iterative with a number of parallel strands. Business Processes are documented initially using UML Activity Diagrams and Class Diagrams, supported by detailed definitions (Glossary) of every data item. UML models provide an effective and flexible way of sharing understanding about the system under consideration.

The next part of this report shows how these technologies may be applied to a simple example, Colorectal Cancer referrals.

3.5 Colorectal Cancer Referral

3.5.1 Scope and Background

This example describes the process when a GP refers a patient suffering from colorectal symptoms for urgent endoscopy to diagnose or exclude possible cancer. This example uses a two-stage process, based on NICE Guidelines (NICE 2004) and the work of Selvachandran and colleagues at the Leighton Hospital, Crewe (Selvachandran et al. 2002).

Colorectal (CR) cancer is the second most common cancer, in terms of both incidence and mortality in England and Wales. With about 30,000 cases a year, each GP is likely to come across about one new case each year. Survival is strongly related to speed of diagnosis and the research literature shows evidence of delays, often lasting a year or more, between the onset of symptoms of colorectal cancer and diagnosis. This is due to patient delay in reporting symptoms, and to a lesser extent, delays by the GP and hospital. For example, a national survey of NHS patients in 1999/2000 found that 37% had to wait over 3 months for their first hospital appointment and 13% waited 7 or more months.

The scope is limited to the business process and decision criteria used to make the “two-week possible colorectal cancer” referral decision by the GP in his or her surgery. All other aspects of the problem are out of scope. In particular, the processes

used to book the GP and the endoscopy appointments, and consideration of problems other than possible colorectal cancer, are out of scope.

3.5.2 Objectives

The objectives of the process are:

1. Reduce the number of days between initial reporting of symptoms and final diagnosis. NB. Survival is strongly related to speed of diagnosis. Five-year survival is:
   (a) Eighty-three percent for Dukes Stage A (localized within the bowel wall)
   (b) Sixty-four percent for Stage B (penetrating the bowel wall)
   (c) Thirty-eight percent for Stage C (cancer in Lymph nodes)
   (d) Three percent for Stage D (distant metastases, most often in the liver)
2. Reduce the number of false negatives (cancer cases missed) and false positives (the number of urgent referrals that are subsequently shown to be free from cancer)
3. Reduce the number of appointments required

3.5.3 Participants and Locations

Referral for possible CR cancer involves not only the direct participants, such as the patient, the GP, and practice staff, but other stakeholders, notably the staff at the units to which the patient may be referred, including doctors, nurses, managers, and clerks.

The patient complains of symptoms and may have cancer. The patient is the primary source of information about history and symptoms and must be present for physical and endoscopic examinations and diagnostic imaging as well as for providing samples of blood, feces, etc. for laboratory tests.

The second key participant is the GP, who takes the decision of whether or not to refer the patient for endoscopy. Reception and secretarial staff in the practice may also undertake some tasks. Although CR Cancer is the second most common type of cancer, each GP sees about one new CR cancer patient a year. The incidence of symptoms that warrant detailed assessment is not an every day occurrence, so any tools used to facilitate this need to be unobtrusive. Perhaps a couple of patients a month present with symptoms that warrant further consideration and half a dozen patients a year need to be referred for urgent endoscopy.

Other actors, such as the e-Booking service and the Endoscopy Unit, receive outputs from the interaction, but are not involved in the decision of whether or not the patient should be referred.

Although the main interaction takes place at the GP surgery, the patient may not know some of the information at the time and may need to consult relatives about details of family history or cross-check the dates at which they first complained of symptoms. For these reasons, detailed history may be collected at the patient's home using a web-based questionnaire. Patients who cannot use a web-browser can be given a paper questionnaire, which can be scanned or transcribed. Much of the information used to make this decision is relevant to subsequent care and treatment and may be collected in a form suited for use in a referral letter.

3.5.4 Outcome

The outcome is a decision of whether or not to refer for urgent (possible cancer) endoscopy. The process can be thought of as two "yes/no" decisions.

1. Does this patient have any CR symptoms that might be indicative of CR cancer, sufficient to warrant more investigation – this "triage" decision is based on the NICE criteria, which include presenting symptoms, physical examination, and the patient's age. If this decision is positive, then take detailed history.
2. The second decision – whether to refer the patient for urgent hospital investigation – is based on a detailed structured history covering: symptoms and presenting history, family history, and past medical history. If this is also positive, then refer urgently for endoscopy.

3.5.5 Storyboard

This section describes a single storyboard, providing a brief description of how the GP referral process might work in the future.

> John Reeves is 64 years old. Over the past couple of months he has noticed that his bowel movements have become loose and more frequent. He makes an appointment to see his GP, Dr Ann Price.

Dr Price sees John, takes his history, examines his abdomen, and suggests that he complete a detailed Colorectal History questionnaire, to be completed at home. John has access to the Internet at home, and the surgery emails John a set of details of the URL for his web-based questionnaire and his instructions.

John completes the form on his computer at home with some help from his wife who reminds him about some details of family history. Next morning, the surgery telephones him to say that the data is complete and asks him to come in and see Dr Price the next morning.

Next morning, he sees Dr Price, who now has the details of his history on her computer screen. The decision support algorithm indicates that there is some cause

for concern. Dr Price notices this and that the symptoms and history warrant urgent endoscopic investigation.

She explains the situation to John and makes a referral to the local Endoscopy Unit via an electronic booking service (Choose and Book). The information collected by the OCR scanner is sufficient to produce a structured referral letter, which Dr Price checks, authorizes, and sends.

John is naturally anxious and so Dr Price goes into an electronic reference (Map of Medicine), where it lists the main reasons for referral for possible CR cancer as well as other data. She prints out a copy of the relevant page and gives it to John.

The next day, John is contacted by the Endoscopy Unit and makes arrangements for the test to be done the next week.

Here only one storyboard has been provided, but in any real project a number of storyboards should be developed covering all of the main scenarios.

3.5.6 Business Process Diagram

The flow can also be shown as a business process diagram using the BPMN notation (Fig. 1). This is similar to an activity diagram (Fig. 3.5).

The main locations (GP Surgery, patient's home, and specialist Endoscopy Unit) are shown as pools. The GP Surgery is subdivided into two lanes (reception and GP consulting room). The rounded rectangles represent separate tasks and the circular icons represent discrete events. The diamond shapes represent decision branches and the "O" icon inside states that the branches are mutually exclusive (OR). The clock icon represents a time-specific event or delay, while the envelope icon represents a message.

The BPMN notation is a formal notation and the diagram can be exported in XML format.

The evidence and rules used to make these decisions are discussed below. We do not discuss what action to take if the answer to either question is "no."

3.5.7 Data and Rules

There are many possible causes of colorectal symptoms and it is important to take note of those combinations of symptoms that may indicate cancer and those that do not. Decision support tools may help in ensuring that the right patients are selected for urgent referral. However, the relatively low incidence of cancer means that any such tools should not be intrusive in normal day-to-day clinical activity.

NICE has listed criteria for urgent referral, based on combinations of symptoms and signs, from which a mind-map has been derived. Seven questions relate to presenting

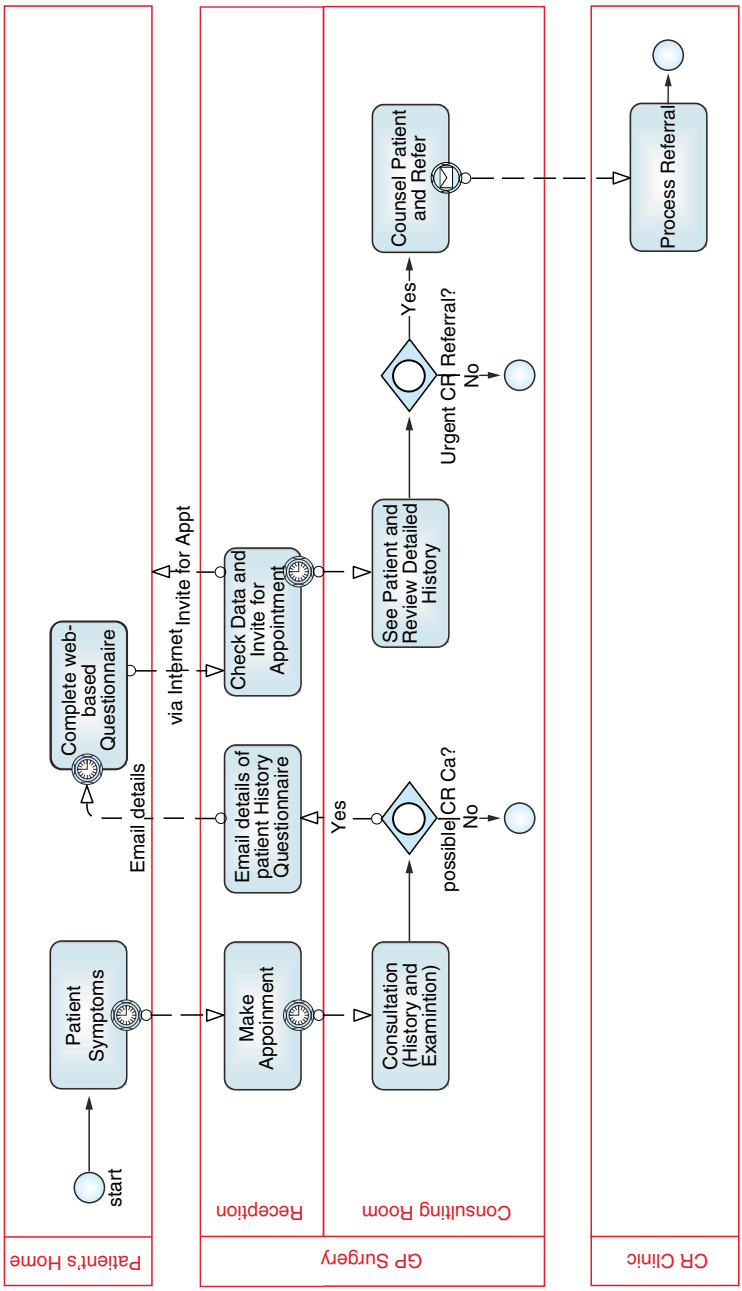

Fig. 3.5 Colorectal cancer referral BPMN diagram

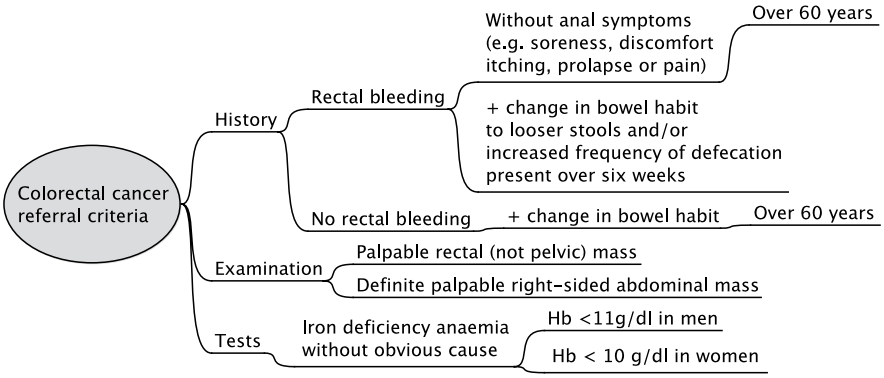

Fig. 3.6 NICE colorectal referral guidelines

history, three to physical examination, and one each for age and hemoglobin (iron deficiency anemia). The decision to refer urgently is based primarily on the patient's report of his or her symptoms and medical history, the patient's age, evidence (or lack of it) from physical examination, and blood tests (hemoglobin).

This set of criteria is used for initial triage. Although about 85% of patients with colorectal cancer meet these criteria (sensitivity), the large proportion of all patients who have these complaints do not have cancer (specificity) (Fig. 3.6).