ELECTRONIC HEALTH RECORD

A Systems Analysis of the Medications Domain

ELECTRONIC HEALTH RECORD

A Systems Analysis of the Medications Domain

Alexander Scarlat, MD

Foreword by John Halamka, MD

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

A PRODUCTIVITY PRESS BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

© 2012 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works
Version Date: 2011913

International Standard Book Number-13: 978-1-4398-7854-5 (eBook - PDF)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (http://www.copyright.com/) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

For Evemine, Adrian, Yuval, Dan, May, and Adam

And especially for my best friend and better half

Dahlia

Contents

List of Figures

Foreword

What is the ideal medication workflow?

Simply put, it is from a doctor's brain to a patient's vein without handwriting, handoffs, or hassle.

Although Beth Israel Deaconess Medical Center (BIDMC) has not used handwritten medication orders since 2001 and has an e-prescribing rate of 96%, the medication workflow is not ideal.

For 20 years, we have automated many existing processes: provider ordering, pharmacy supply chain, positive patient identification, medication administration records, and billing. However, automating an imperfect process does not make it better. Automation just makes it faster. Over the past year, we have spent 3,000 hours reexamining the entire medication workflow using a multidisciplinary Lean process. What is the best way to package and deliver medications? How should care delivery teams reconcile medications and communicate issues? How do we involve the patient?

We are now turning this idealized design into a detailed specification that we can automate.

Doing this work takes three kinds of people: medication experts with domain knowledge of the workflow, information technology (IT) experts with domain knowledge of available hardware/software, and project managers who glue the two together.

As an IT professional, I have worked on hundreds of projects. The most successful are those with business owners who understand their requirements and can communicate them to IT implementers. In one recent large project, we taught systems analysis to the stakeholders because the project bottleneck was communicating the idealized workflow. The project is now on time to go live.

Dr. Alexander Scarlat's *Electronic Health Record: A Systems Analysis of the Medications Domain* is important for two reasons. First, it provides a framework that will enable clinicians to communicate with technologists. As the pace of IT change accelerates, so does demand for workflow automation. Technology is no longer the rate-limiting step. Translating user needs into IT products and services is the biggest challenge we face. Dr. Scarlat's clear explanations empower healthcare professionals with tools that will enhance any IT project.

Second, the medication domain is the highest-priority area for healthcare stakeholders to improve quality, safety, and efficiency. Meaningful Use Stage 1 included numerous medication lists, e-prescribing, allergy, reconciliation, and decision support requirements. Meaningful Use Stage 2 will include even more medication-related activities, emphasizing the importance of automating these processes correctly.

I plan to use this book in the BIDMC medication work, which seeks to achieve zero defects, cost reductions, and patient engagement. Both clinicians and IT professionals should find the book is a valuable resource as they create the re-formed healthcare delivery system of the future, beyond Meaningful Use.

John D. Halamka, MD
Boston, Massachusetts

Preface

Why This Book?

During the years I have worked in healthcare information technology (IT), either with vendors of electronic health records (EHRs) or the hospitals implementing this technology, I have been asked many times by colleagues to recommend a practical book on the subject. The clinicians—physicians, nurses, pharmacists—are looking for a book to introduce them to the modeling techniques and engineering tools the IT department or their vendor uses, while the information technologists are interested in an introductory text on the clinical domain, something that can be assimilated without going to medical school. Realizing these two highly educated groups of professionals rarely speak the same language, they need a communication tool.

Systems analysis in general and structured systems analysis specifically offer a rigorous methodology and can create such communication tools in the form of models, diagrams, and other succinct, systematic artifacts. Systems analysis has been successfully applied in thousands of IT projects in other industries, but somehow it just never caught on in the healthcare IT domain. One can find books and articles on the vendor selection process, best practices for user training, and the psychological, sociological, and financial aspects of informatics in medicine as well as the cultural and organizational change management that accompanies the implementation of IT in clinical environments. However, to my best knowledge, very little if any has been published in the domain of healthcare informatics systems analysis.

Even when an organization does have the systems analysis related to EHRs it has developed or implemented, this reference is considered an intellectual property to be guarded accordingly, as a secret possession. Unfortunately, a software blueprint kept in a vault cannot be improved, lessons learned by others cannot be applied to or derived from it, and the professionals involved in the development or implementation of EHRs may find themselves "reinventing the wheel" solving problems others have already solved or worse—making the same mistakes others did before them.

Finding no better (published) alternative, I decided to write this book with two goals in mind:

1. Introduce the clinicians to a methodology and tools set so they can better communicate with the IT specialists developing or implementing EHRs
2. Clarify the complexity of the medications domain, a major part of an EHR—the prescription, dispensation, computerized physician order entry, and administration of drugs—for IT professionals to better communicate with their clinical colleagues

In addition, the last chapter on interoperability presents to both clinicians and informatics professionals the standards and vocabularies, rules, and regulations governing the EHR as recommended by U.S. authorities in the recently published American Recovery and Reinvestment Act (ARRA). According to ARRA, meaningful use of certified EHR systems in the United States will entitle clinicians to substantial funds from the U.S. government over the next couple of years. Thus, the need for a structured, systematic analysis of EHRs has aligned with a major, current development in the U.S. healthcare IT industry: the publication of government-backed standards for the industry to adopt to be considered a meaningful user of this technology.

What Is in This Book?

The first chapter is a short primer on structured systems analysis intended to introduce beginners to the tools used in the book: conceptual and logical data models, entity relationship diagrams, context and data flow diagrams.

The second chapter initiates the analysis of the medications domain from two perspectives: workflows and data structures. Workflows are decomposed into processes, activities, and tasks, while data structures are partitioned in the conceptual, entity-relationship, and logical data models.

The analysis continues in a more detailed fashion in Chapters 3 through 6, which respectively discuss the process to prescribe, order, dispense, and administer drugs.

The next three chapters illustrate how the system is expected to interact optimally with users: Chapter 7 is on the user interface; Chapter 8, "Clinical Decision Support," involves what the system can and should do with the information it possesses; and the output expected is discussed in Chapter 9, "Report."

The final chapter presents the standards, vocabularies, rules, and regulations, which if adopted and implemented correctly will enable various disparate EHR systems to play together in the proverbial healthcare IT sandbox.

What Is Not in This Book?

Throughout this book, we examine together the EHR medications domain using structured systems analysis methodology and tools. The system described is a vendor-, platform-, and computer language-independent generic EHR. The reader will not find any vendor or hospital, technology or specific database, application or product mentioned by name. At least initially, system analysis is blind to these issues.

The goal of the book is not to provide the complete, correct, and consistent technical specs for such an ideal, hypothetical system to be built. The book presents the tools and methods to enable and facilitate the communications between clinicians and IT so they in turn can interpret, develop, and implement such clinical-technical specifications. Thus, do not expect to find actual software code, hardware specs, testing scripts, or physical data model minutiae in the book.

In the same spirit, the pharmacological examples used are for IT demonstrative and illustrative purposes only; the examples are not intended for actual clinical or medical use.

Is This Book for You?

The book should be accessible to both clinicians and IT professionals: No initial knowledge of medicine or computer sciences is assumed or required.

Clinicians: Physicians, nurses, pharmacists, C-level administrators, and other healthcare professionals contemplating or in the midst of the implementation of EHR technology will benefit from the succinct diagrammatic representations of the medications domain: workflows and data elements.

IT: Software engineers, architects, data and process modelers, product and project managers, database administrators, analysts, testers, and consultants will find the book useful in clarifying the terms and verbiage used by the clinicians in their work, while reviewing modeling techniques used by other industries for more than four decades.

Whether in the process of requirements elicitation, analysis, design, code, test, configuration, implementation, or maintenance of EHRS, both clinicians and IT professionals can benefit from better techniques to facilitate the abstraction of complex ideas into a set of easily understood models and diagrams. It is hoped both groups should find a common ground, clear and easy-to-use methodology and tools to make their collaboration on the medications informatics domain more productive.

Each chapter ends with a set of questions that should help the reader review the material covered; answers are found in the back of the book. Students and teachers of healthcare informatics will find the book is both methodical and

didactic for their academic purposes; thus, it could be used as a textbook for a course on the subject.

When applied correctly, systems analysis can accelerate development, decrease costs, and increase the quality of a software product and eventually user satisfaction. I believe systems analysis can be applied to healthcare informatics with the same successful results demonstrated in other industries.

Acknowledgments

I am thankful to many individuals who have provided useful feedback and encouragement during this journey: John Halamka, William Bria, Pini Matia, Joseph Finn, Karl Matuszewski, Philip Anderson, Dorel Abramovici, Jane Brokel, Laurette Floru, Steven Lins, Jon Patrick, Ruth Tamari, Mansnimar Singh, Scot Silverstein, Brian Berenbach, John Frownfelter, Saswat Mohanty, Ziv Mayanski, and Eyal Frank.

Many thanks to Kristine Mednansky and others from Taylor & Francis/CRC Press for helping bring the idea to the masses.

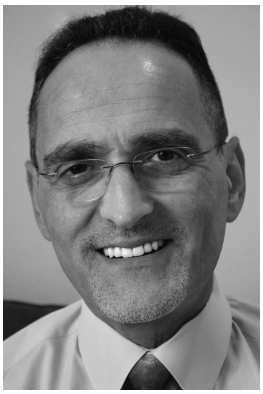
Suggestions?

This book may contain errors, and most probably can be improved. If you want to contribute suggestions, additional material, ideas, or diagrams, feel free to provide feedback at the book's Web site: http://drscarlat.wordpress.com/.

I hope you will find the book useful and enjoy it as much as I derived pleasure from writing it.

A. Scarlat, MD

About the Author



Alexander Scarlat, MD, is a physician with a degree in computer sciences and a strong background in clinical informatics. Dr. Scarlat's career spans more than two decades, during which he has worked with both vendors developing electronic health records and healthcare organizations implementing this technology. Knowledgeable and experienced in medicine and informatics, Dr. Scarlat is an efficient liaison between these two highly educated groups of professionals—clinicians and information technology—that rarely speak the same language.

Chapter 1

Short Primer on Structured Systems Analysis

Everything should be made as simple as possible but not simpler.

Albert Einstein

The first chapter is a short primer on structured systems analysis: It explains what a system is, why systems analysis is needed, and the reasons this book is based specifically on structured systems analysis.

We discuss the methodology and introduce a set of tools that will be used throughout the book. If you are familiar with structured systems analysis, you may skip to the next chapter.

What Is a System?

According to the Merriam-Webster dictionary online (1), a *system* is

> *a regularly interacting or interdependent group of items forming a unified whole … an organized set of doctrines, ideas, or principles usually intended to explain the arrangement or working of a systematic whole … an organized or established procedure … harmonious arrangement or pattern.*

We are continuously interacting with natural and human-made systems all our lives. Some systems have been thoroughly analyzed (digestive system, highway system, solar system), and some like Electronic Health Record (EHR) systems and the medications domain are only sparingly documented.

Why Systems Analysis?

As no one would build a house without a set of blueprints, no one should consider building a nontrivial computer application without a systematic, structured set of plans. As seen in this book, healthcare information technology (IT) in general and software applications for prescribing, ordering, dispensing, and administering medications specifically are far from trivial systems. The rationale for writing this book is to present the medications domain in a structured, systematic manner. The goal of systems analysis is to create an efficient model of users' needs, the plan for the system under consideration (2,3). When done correctly, systems analysis:

- Creates a set of communication tools for clinicians and IT professionals
- Optimizes all stages of a system life cycle: scope definition and control, requirements elicitation, analysis, design, code writing, testing, implementation, training, and maintenance
- Defines the system in a complete, correct, and concise manner
- Emphasizes certain aspects of a system while deemphasizing other areas
- Partitions a huge thing—"the system"—into smaller work chunks
- Increases consistency between the system parts: modules and data structures
- Minimizes redundancy between components of the system
- Improves maintainability of the system

Can we achieve the same goals using only text documents? Can systems analysis be done with text and spreadsheets? This is not really possible for the following reasons:

- Text-based and spreadsheet documentation of a system tends to grow into intimidating monsters of hundreds and even thousands of pages. Nobody

really likes to read this stuff. The old saying about a picture being worth a thousand words is painfully true when documenting IT systems.

- Text and spreadsheets are monolithic and thus cannot be systematically chunked or partitioned into smaller working units. IT systems are usually developed by a number of teams (database, application, user interface, testing, etc.), and the capability to work in parallel on several parts of a system can be advantageous from a productivity perspective.
- It is easier to document ambiguous statements and requirements in text and spreadsheets than in structured diagrams.
- Due to the dynamic interactions between users and developers while defining and refining the system and the ever-changing nature of the requirements, the moment a text document or spreadsheet is published, it is already outdated, irrelevant, and out of sync with reality. Maintaining text documents and spreadsheets is more difficult than maintaining a set of diagrams.
- Text documentation is more prone to suffer from redundancy issues: The same fact—be it a requirement or a solution—may be documented in more than one place. If a fact is updated on one page, there is an immediate need to update all other pages that may document this fact. Redundancy breeds inconsistency unless a significant effort is invested in keeping multiple documents in sync.

The truth is systems analysis is not optional; this is a paraphrase on *Simsion and Witt's* famous adage, *"Data modeling is not optional"* (4). While it may seem to be a self-evident point, I have seen my share of software applications being developed without a blueprint, usually with less-than-satisfactory outcomes. "Some text and spreadsheets in a folder" do not count as a blueprint, and one cannot expect become efficient in systems analysis if one uses only text and spreadsheets as the sole tools.

Why Structured Systems Analysis?

There are two main systems analysis methodologies: *Structured Systems Analysis* (*SSA*) and *Unified Modeling Language* (*UML*) (5). When comparing the two methodologies, I prefer *SSA* to *UML* since the former is easier to understand, learn, become efficient with, and explain to nonexperts. *SSA* is also less prone to misunderstandings and errors than *UML*. In his book, *Modern Structured Analysis, Ed Yourdon* listed the characteristics of SSA tools (3):

- It is graphical, with appropriate textual support.
- It allows the system to be viewed in a top-down, partitioned fashion.
- It suffers from minimal redundancy.
- It helps predict the system's behavior.
- It is transparent and easy to learn, understand, and use.

To this list, I would like to humbly add a sixth characteristic:

- It is technology and implementation independent—the modeling tools should not be limited to any particular implementation or platform technology during the analysis phase. Nor should the clinicians or IT professionals waste precious time on discussing technology-related issues at this early embryonic stage.

Processes and Data

Using the house-building analogy, as the builders need more than one kind of plan (architectural, structural, plumbing, electricity/wiring, etc.), the IT craftspeople need several modeling tools as well. If we agree that an information system is a data-processing system, it will not come as a surprise that *SSA* proceeds from two main perspectives: processes and data (3,6). A top-down analysis of system functions reveals the following concepts at an increased resolution and higher details:

System → *Workflow* → *Process* → *Activity* → *Task*

A system may have multiple workflows. Each workflow may be a collection of several processes. A process can be decomposed into several activities. An activity can be decomposed into several tasks. An activity or task can be defined in one page of graphic or text. For example,

System: Medications → *Workflow:* Prescribe → *Process:* Modify Rx [prescription] → *Activity:* Select medication → *Task:* Select dose unit

Similar top-down analysis of system data structures uncovers the following terms:

System → *Repository* → *Database* → *Table* → *Record* → *Field*

A system may use multiple repositories. Each repository may be composed of several databases. One database usually has many tables. A table has many records, and one record is composed of several fields. For example,

System: Electronic Health Record → *Repository:* Clinical Repository → *Database:* Medications → *Table:* Patient Medication → *Record:* Patient ID, Drug ID, Strength, Strength Unit, … → *Field:* Strength

The following modeling tools are used throughout the book:

- Dataflow diagram (DFD) details the functions the system is to perform and represents data in *motion.*
- Entity-relationship diagram (ERD) details data structures used by the system for storage and represents data at *rest.*

These modeling tools create diagrams that show the main components of a system and the interactions between these components. A technical specifications document (but not this book) may also have two sets of highly structured supporting documents to provide the meaning and definitions of the components and interactions: *Data Dictionary* and *Functional Primitives Specifications.*

If the system under consideration has aspects of a "real-time system," then a third tool, a *State Transition Diagram,* may be employed (3). A real-time system is defined as a hardware/software configuration that can receive high-volume and process at high-speed information coming from the environment in a way that it can control some aspects of the environment or direct a process that takes place in the environment. Real-time systems are embedded in missile guidance, automatic pilots, or mundane washing machines and elevators. Since the medications system rarely if ever exhibits characteristics of real-time systems, we do not detail *State Transition Diagrams* any further.

Dataflow Diagram

Also known as workflow diagram, bubble chart, bubble diagram, or process model, the DFD is a tool used to model data in motion—the system functions (Figure 1.1):

- The origins and destinations of data within a system
- The workflows, processes, activities, and tasks of the system
- The transformation of inputs into outputs

The components of a DFD are

1. *External entity*: a rectangle representing people, groups of users, personae, roles, other computer systems or organizations that interact with the system.
2. *Dataflow*: an arrow representing data in motion, the path for data to move from one part of the system to another, the input needed or the output generated by a component of the system.
3. *Process*: a bubble representing an activity or task performed by the system. It is something the system does: the transformation of an input into an output or the process of moving data from one place to another.
4. *Data store*: two parallel lines representing data at rest, a location for the data to be stored for future use by the system's components.

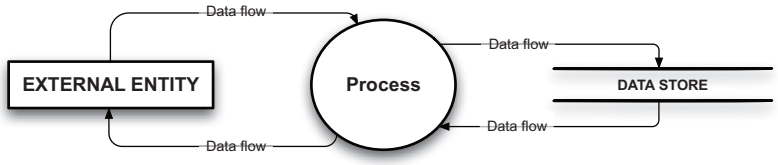


Figure 1.1 DFD symbols.

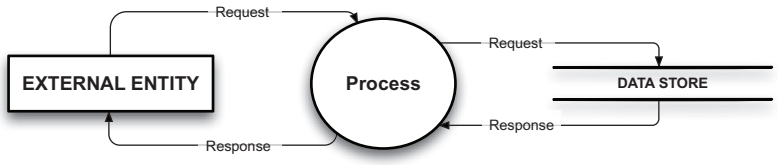


Figure 1.2 DFD request/response format.

Most of the dataflows represent a series of asynchronous messages, a request/response type of communication: A component requests information, and another component responds accordingly (Figure 1.2).

Since some of the DFDs we are to encounter can become quite complex and thus potentially cluttered, we can simplify the diagrams using dataflows with double-headed arrows—a dialog format. Usually the request name is on the top and the response name on the bottom of the dataflow arrow (Figure 1.3).

Let us examine a DFD example showing the process of prescribing a new medication for a specific condition or indication, such as hypertension (Figure 1.4). What can we learn from this DFD?

PRESCRIBER, an external entity, sends the *Patient ID* as a request for a patient's clinical information to process 1.1 *Review Patient Clinical Data.* Process 1.1 *Review Patient Clinical Data* queries the *CLINICAL REPOSITORY* data store, which responds with patient-specific information: age, weight, gender, conditions patient may suffer from, medications patient is taking, allergies, and so on. Process 1.1 displays the information returned, the query results, to *PRESCRIBER.*

PRESCRIBER sends an indication (such as hypertension) to process 1.2 *Select Drug*, which queries a different data store: *RECOMMENDED DRUG DOSE.* The query results, a list of drugs indicated for treating the condition, is returned to process 1.2, which displays the list to *PRESCRIBER.*

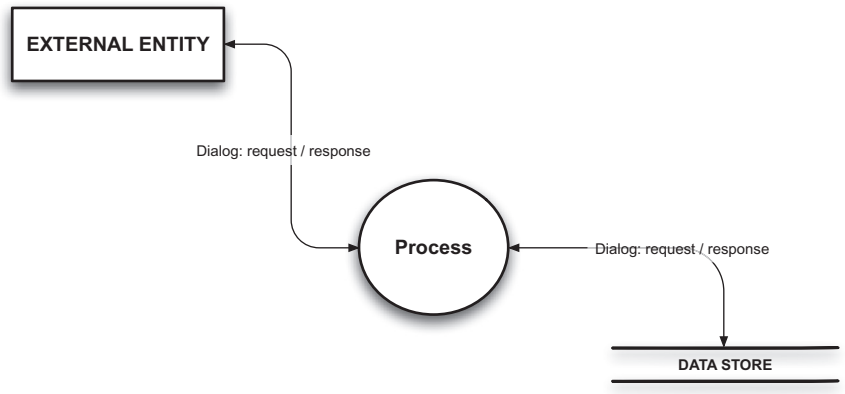


Figure 1.3 DFD dialog format.

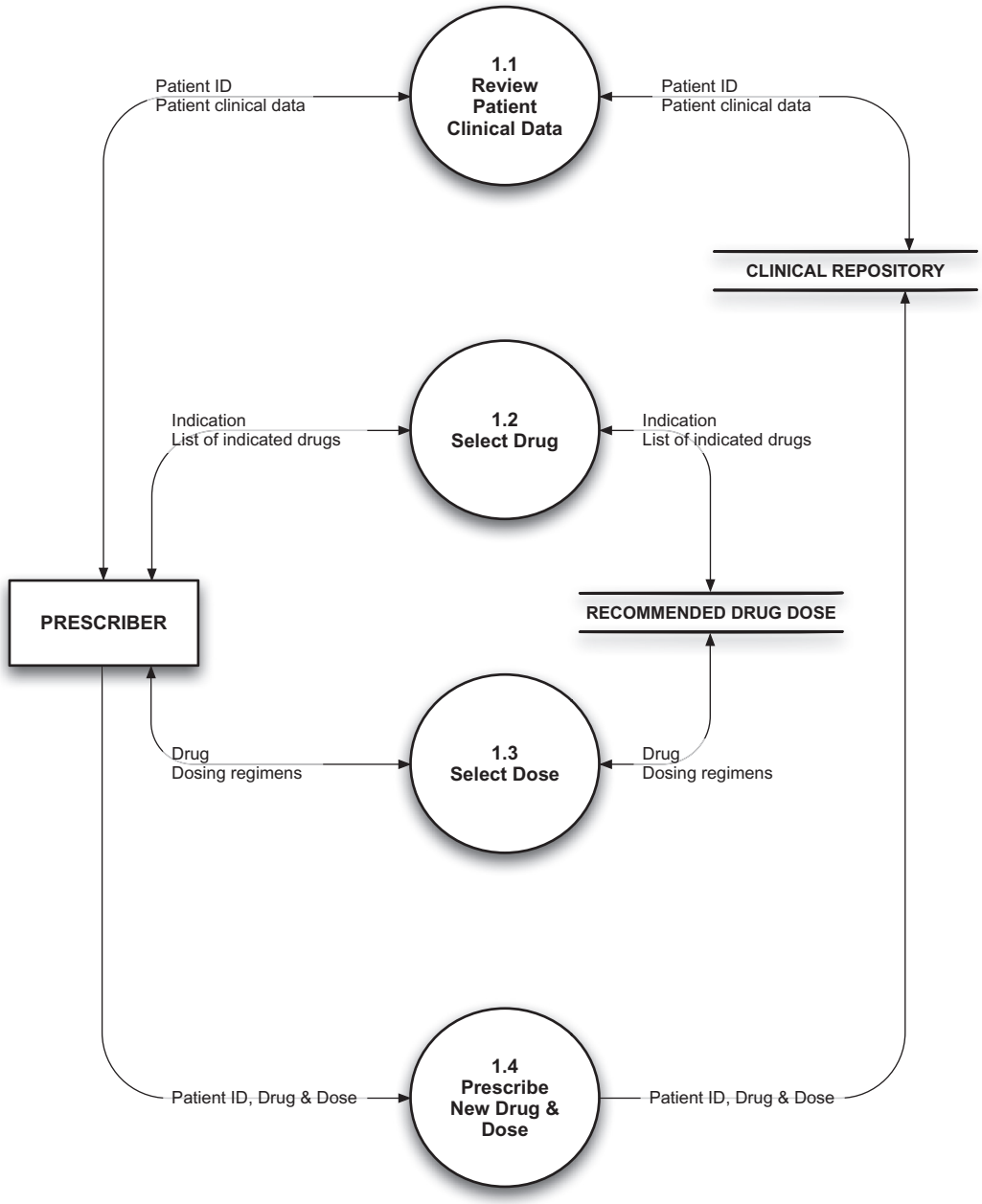


Figure 1.4 DFD example: New Rx (prescription).

Once *PRESCRIBER* selects a drug, it is sent to process 1.3 *Select Dose.* Process *1.3 Select Dose* queries *RECOMMENDED DRUG DOSE* data store and retrieves the dosing regimens for the specific drug (usual dose, maximal dose, age and weight considerations, etc.) and displays the results to *PRESCRIBER.* Finally, *PRESCRIBER* sends the *Patient ID, Drug,* and *Dose* to *CLINICAL REPOSITORY,* where it is stored.

This DFD example can convey to both user and developer a complex process using a simple diagram. In addition, the DFD is easier to maintain than the text that explains it.

Let us discuss some aspects to consider when building a set of DFDs:

The DFD design process starts with a *context diagram*: In this simple, highest-level DFD, there are no data stores, and the system is represented as one bubble/megaprocess interacting with the external entities.

The next step in DFD decomposition is to draft the *DFD 0*: the black-box "system" represented by one bubble in the context diagram, also known as *process number 0,* is decomposed into its main workflows. We construct the context diagram and DFD 0 in the next chapter when we begin the top-down analysis of the medications domain.

The following are some recommendations to consider when drafting a DFD (6,7):

Avoid an overly complex DFD. A DFD should fit nicely into one page. As a rule of thumb, if there are more than a half dozen processes on a DFD, it usually means either the partitioning of the parent, higher-level DFD is incorrect or the current DFD needs a different decomposition approach.

The process of decomposing, partitioning, or exploding the system using DFDs is also called *leveling*. Leveling uses a series of increasingly (top-down) or decreasingly (bottom-up) detailed DFDs to represent a system from a functional perspective.

When is the work of decomposing DFDs supposed to end? The process of decomposing DFDs ends when we reach a stage at which a process, activity, or task is small enough to be detailed in one page. This detail level is called a *functional primitive,* and it is documented using a *flowchart, structured English* (*pseudocode*), or a *decision tree* in a manner that programmers can actually write software for it.

The system is responsible for all the interactions between external actors. Any other interaction between external actors that is outside system boundaries is not to be detailed in a DFD. The corollary is that there should be no direct dataflows between external entities.

Data stores cannot directly communicate with each other. Data stores are representations of data at rest; thus, data stores cannot initiate or complete a data transfer or data processing on their own volition. The corollary is that there should be no dataflow without a process between two data stores.

An external entity cannot communicate directly with a data store. The *PRESCRIBER* cannot retrieve the list of drugs directly from the *RECOMMENDED DRUG DOS*E data store, and the *PRESCRIBER* cannot directly retrieve information about the patient from the *CLINICAL REPOSITORY*. There must be a process, software code, that extracts the information from the

data store. The corollary is that there should be no direct dataflows between external entities and data stores without a process in between.

Avoid *black hole* data stores: data stores that have only inputs and no outputs.
Avoid *spontaneous generation* data stores: data stores that have only outputs and no inputs.
Avoid *gray hole* data stores: data stores that have at least one input and one output, but the data are obviously insufficient or irrelevant to produce the output

A DFD represents a set of *asynchronous* processes. The processes numbers do not reflect their sequence. Thus, in Figure 1.4 the process 1.3 *Select Dose* may or may not be processed after process 1.2 *Select Drug*. Since processes are asynchronous, there is no guarantee the correct or expected sequence will be followed by the system. According to the DFD example in Figure 1.4, a dose may be selected before a drug was chosen, a situation that does not make much sense.

There are no components on a DFD to describe *conditional logic*. A DFD has no condition-action rules, such as "If condition X is fulfilled then act according to Y. Otherwise do Z." The time-dependent, sequential nature of a process and the conditional logic a process has to obey are defined with flowcharts, pre- and postconditions, and sometimes *control processes*: bubbles that control the behavior of other bubbles.

Then, why bother numbering processes? The numbering system of the processes is used to *level and balance* the DFDs (3,6). Since a system may have hundreds and thousands of processes and since we cannot clearly depict more than half a dozen "bubbles" on one diagram, we use the numbering system to level the DFDs.

Each and every diagram in the top-down analysis process is named according to the higher-level process it details. For example, the parent diagram named DFD 1 (Figure 1.4) has four processes: 1.1, 1.2, 1.3, and 1.4. The child diagram named DFD 1.2 *Select Drug* will represent the decomposition of the process (bubble) numbered 1.2 from the parent diagram. Assume DFD 1.2 *Select Drug* has its own three processes (not shown in Figure 1.4): 1.2.1, 1.2.2, and 1.2.3. A lower-level child diagram of DFD 1.2 can continue the partitioning, top-down analysis, for example, detailing the 1.2.3 bubble. That diagram will be named DFD 1.2.3 and its own bubbles will be numbered 1.2.3.1, 1.2.3.2, 1.2.3.3, and so on. Leveling keeps the processes (bubbles) in sync with the diagrams.

During the leveling activity, one has to *balance* the processes with the diagrams: The inputs and outputs shown on a high-level, parent bubble have to correspond to the same inputs and outputs detailed on the child DFD. Ignore the internal dataflows and consider only the inputs and outputs to and from external entities: If a process on a DFD has two inputs and three outputs to and from external entities, then the DFD detailing the process must have the same two inputs and three outputs to and from external entities for the process and diagram to be considered balanced.

We are done with data-in-motion analysis and its tools: the DFD. Let us introduce the ERD, an analysis tool for data at rest/databases.

Entity Relationship Diagram

The ERD (also known as an E-R diagram) is a tool used to model data structures used by the system and their relationships (4). Since the data structures and relationships may be quite complex, *SSA* recommends analyzing them in parallel to and separately from the system workflows.

The components of the ERD are as follows (Figure 1.5):

1. *Entity*: a rectangle (not to be confused with the external actor in DFD, which is also a rectangle) represents "something" that the system has to collect, store, and usually retrieve information about, something the system has to "remember." Usually, an entity will evolve during the design phase into a database table. A row in such a table is called an entity *instance* or *occurrence*. An important rule to remember is that each and every row or instance in an entity must be *uniquely identified*.
2. *Attribute*: data elements that describe the entity. The attributes are the characteristics of an entity. In a database table implementing the concept of an entity, the attributes are the table columns. Attributes can be optional or mandatory: A required mandatory attribute must have a value and cannot be left empty, while an optional attribute can be left empty with no value stored.
3. *Relationship*: lines between the rectangles describe associations between entities. The relationships can be defined according to *cardinality*: *one to many* (*1:M*), *many to many* (*M:N*), or *one to one* (*1:1*). Relationships can also be defined by *optionality*: whether the relationship is optional or mandatory.

Figure 1.5 should be read as follows: An instance (occurrence) of *Entity A* may have one or more relationship *name A to B* with *Entity B*. The small circle near *Entity B* means that the relationship is optional. The "crow's foot" near *Entity B* means a cardinality of many.

Now, we traverse the diagram in the opposite direction, from *Entity B* toward *Entity A*. This should be read as follows: An instance (occurrence) of *Entity B* must have only one relationship *name B to A* with *Entity A*. The lack of a crow's foot near *Entity A* means a cardinality of one. The small vertical line near *Entity A* means the relationship is mandatory in that direction.

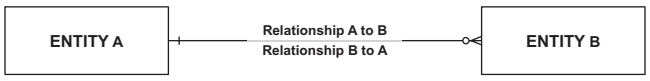


Figure 1.5 ERD symbols.

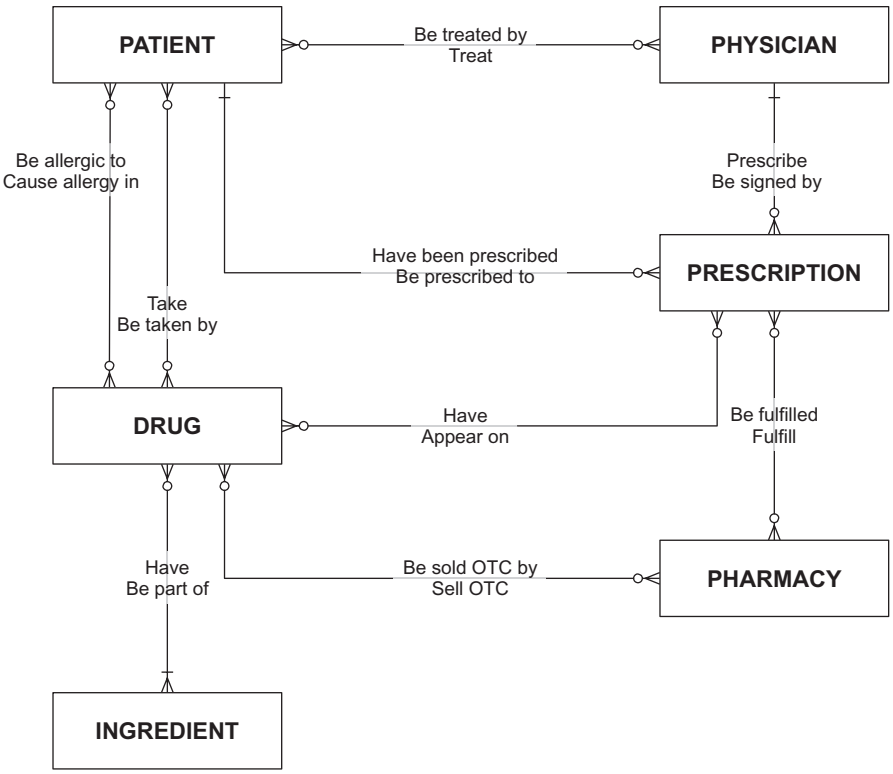


Figure 1.6 Conceptual ERD example.

Let us examine Figure 1.6, an example of a conceptual model. The example is a partial *conceptual* ERD model since it does not have all entities involved. Remember that a medium-size system may have hundreds of entities. It is a conceptual model and not a fully developed logical model ERD since the entities do not have at this stage any attributes defined.

What can we learn from this conceptual ERD?

- A *PATIENT* may be treated by one or more *PHYSICIANs*. A *PHYSICIAN* may treat one or more *PATIENTs*.
- A *PHYSICIAN* may prescribe one or more *PRESCRIPTIONs*. A *PRESCRIPTION* must be signed by only one *PHYSICIAN*.
- A *PATIENT* may have been prescribed one or more *PRESCRIPTIONs*. A *PRESCRIPTION* must be prescribed to only one *PATIENT*.
- A *PRESCRIPTION* may have one or more *DRUGs*. A *DRUG* may appear on one or more *PRESCRIPTIONs*.
- A *PATIENT* may take one or more *DRUGs*. A *DRUG* may be taken by one or more *PATIENTs*. Note that a *DRUG* does not need a *PRESCRIPTION* since it can be purchased as over the counter (OTC).
- A *PATIENT* may be allergic to one or more *DRUGs*. A *DRUG* may cause an allergic reaction in one or more *PATIENTs*.

A *DRUG* must have one or more *INGREDIENTs*. An *INGREDIENT* may be part of one or more *DRUGs*.
A *PHARMACY* may fulfill one or more *PRESCRIPTIONs*. A *PRESCRIPTION* may be fulfilled by one or more PHARMACY.
A *PHARMACY* may sell one or more OTC *DRUGs*. A *DRUG* may be sold OTC by one or more *PHARMACY*.

The following are some aspects to consider when building the ERD:

The conceptual model is the first, rough draft, and simplest modeling tool representing the system's data structures.
Entities are mentioned with the singular name, not the plural, hence: *PHYSICIAN* and not *PHYSICIANS, PATIENT* and not *PATIENTS*. It makes the task of constructing statements like, "An instance of *DRUG* must have one or more instances of *INGREDIENT,*" much easier if naming entities is kept at the singular level.
Usually, the relationships shown on the conceptual diagram are the representation of business rules applying to the system. Relationships should always be named.

The conceptual model is translated into a *logical* data model, which specifies what are the system data structures and their relationships. There are more details on a logical model than on a conceptual model.

Attributes are added to the entities in the logical model. The attributes (table columns) can belong to several types or categories (also known as data types): Boolean (yes or no/true or false), integer or floating point number, currency, fixed- or variable-length string, or binary object type. The conceptual and logical models are usually technology independent.

The logical model is further developed into the *physical* model, which shows how the model actually is implemented in a database system. The physical model is more detailed than the logical model, and it considers the attributes' types, physical storage, performance, indexing, referential integrity, access mechanisms, and load distribution. The physical model is usually technology dependent.

Normalization

In the DFD example (Figure 1.4) there is a *CLINICAL REPOSITORY* data store. This is a representation of a cluster of several databases—a "mega" data store. For the sake of brevity and clarity, the DFD at this high-level view does not illustrate the various tables the *CLINICAL REPOSITORY* may have: patient demographics, patient diagnosis, surgical procedures, physical exams, immunizations, medications, allergies, lab values, consultant notes, and so on. As the DFDs are decomposed into more detailed views, it will reveal more granularity and

uncover the actual tables that may comprise *CLINICAL REPOSITORY*. While it is okay to refer to a cluster of databases in the high-level DFDs as well as in the conceptual model, these repositories must be decomposed into more granular data structures—tables in the logical and physical models.

The developers are employing a formal and quite rigorous technique of *normalization* during the modeling process. The normalization rules, introduced by Edgar Codd (who conceived the relational database model in the 1970s), govern the manner in which data are allocated to the various tables. The purpose of the normalization is to eliminate *redundancy, inconsistency,* and *incompleteness* from the data model and prevent *anomalies* during database updates (3–8).

The normalization process of a data model usually starts with one large entity (such as *CLINICAL REPOSITORY*), and it divides this megastore into smaller tables according to a set of rules. These entities/tables in turn will have *primary keys* (*PKs*) defined. Attributes are then allocated to the tables and finally the tables are connected using *foreign keys* (*FKs*) based on the relationships—the dependencies discovered in the analysis of the business rules governing the system.

One of the normalization rules calls for the resolution of all the *many-to-many (M:N)* relationships. The *M:N* relationships that may still exist at the conceptual level model are not acceptable in a fully normalized model. In a normalized ERD, these *M:N* relationships are usually evolving into an *intersection entity* (also known as an associative, junction, bridge, or composite entity). The intersection entity is resolving the *M:N* relationship in a way that each original entity will have a *1:M* relationship with the newly created intersection entity.

Let us examine a more detailed ERD—a sample of a logical data model—evolved from the conceptual model (Figure 1.7). The main points to note in this example are as follows:

Each and every instance/occurrence of an entity must be uniquely identified. This is achieved using *PK*. The *PK* needs to be stable and not change during the system lifetime. Any other *non-PK* attributes may be updated/changed without disturbing the *referential integrity* of the database. For example, the *Patient last name* does not have to be repeated in the entity/table detailing patient's allergies. Nor does the *Drug name* need to be repeated in the table *PATIENT ALLERGY*. For the connectivity purpose, we use the *Patient ID* and the *Drug ID* as *FKs* in the *PATIENT ALLERGY* intersection entity. Thus, the *PK* from the table *DRUG* and the *PK* from the table *PATIENT* are used to connect the *PATIENT* and *DRUG* entities in the intersection entity named *PATIENT ALLERGY*, where they appear as *FK*.

Attributes have been added in the logical model: *Patient gender* in the *PATIENT* table, *Drug name* in the *DRUG* table, *Encounter actual start time* in the *ENCOUNTER* table, and so on. The ellipsis (…) symbol in the depicted entities is there to remind us that the list of attributes shown is partial, tentative, and by no means exhaustive. Reiterating through the

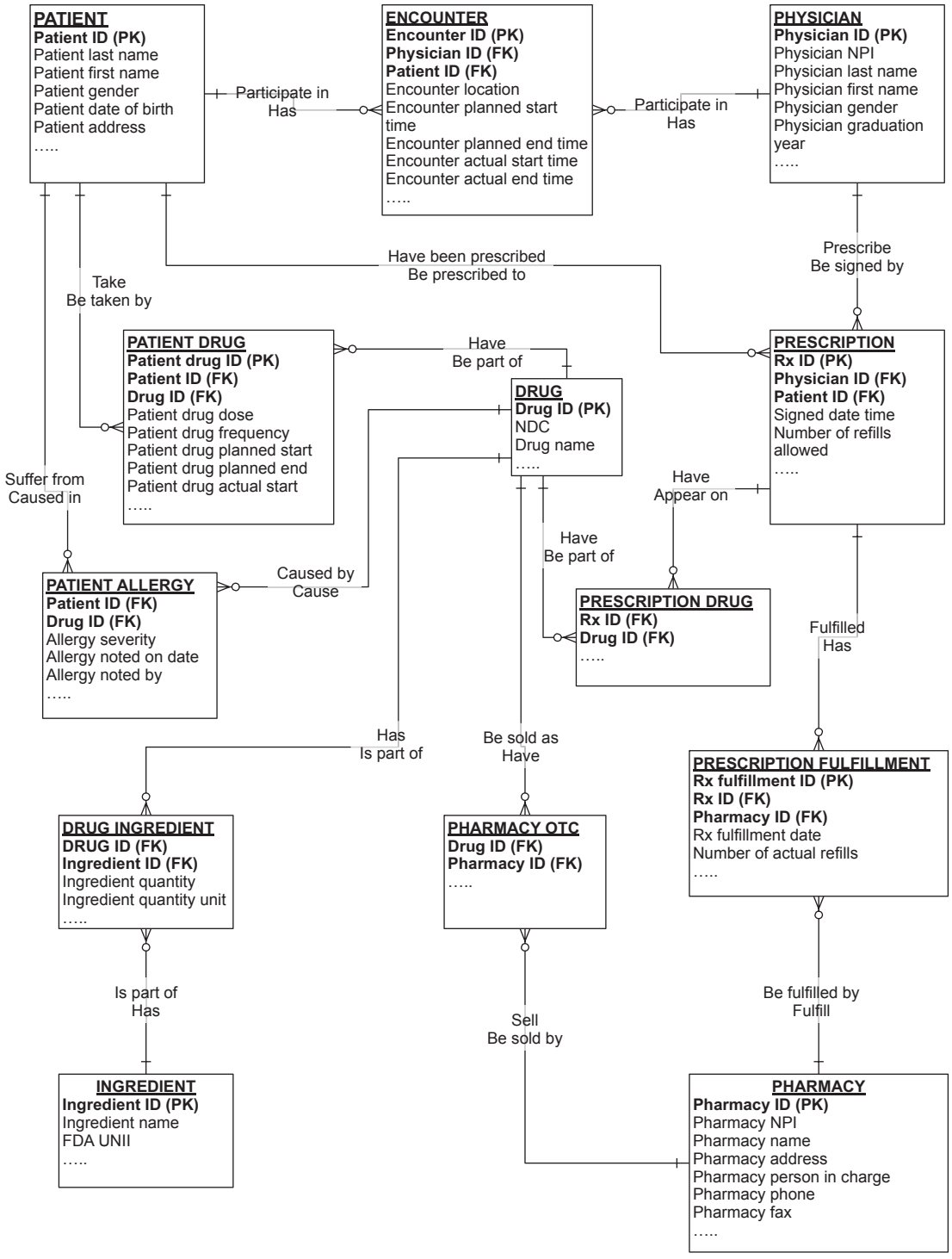


Figure 1.7 Logical model: ERD example.

analysis and refining the ERD, attributes may be added, removed, or relocated from one entity to another.

There is no need to repeat attributes between entities. Any redundant replication of data is a potential source for inconsistency. If the address of a patient changes, for example, we need update only the *Patient address* in the *PATIENT* table. The unchanged *Patient ID* (the *PK* and unique identifier for the patient) will remain the same in all the related, referenced tables (where it appears as an *FK*), and we need not worry about updating the *Patient address* in any other tables.

Sometimes, using the combination of two or more foreign keys forms a unique identifier. If the combination of the foreign keys *Patient ID* and *Drug ID* in the table *PATIENT ALLERGY* uniquely identifies an instance of this entity, an occurrence of a specific allergy in a specific patient, then it is perfectly okay to use the combination as the *compound PK* for this table.

The *M:N* relationships have been resolved using intersection entities. For example, instead of the *M:N* relationship between the *PHYSICIAN* entity and the *PATIENT* entity that was present in the conceptual diagram, the logical model shows an intersection entity named *ENCOUNTER*. Sometimes, there is no clear naming solution for an intersection entity; in these cases, we use the concatenation of the names of the two entities we are intersecting. For example, instead of the *M:N* relationship between the *PATIENT* and *DRUG* entities we had initially on the conceptual model (one *DRUG* may be taken by one or more *PATIENTs* and one *PATIENT* may take one or more *DRUGs*), the logical model will have an intersection entity named *PATIENT DRUG*. The intersection entities created usually will have a different optionality than the *M:N* relationship it has resolved. The initial many-many relationship between *PATIENT* and *PHYSICIAN* was optional in both directions ("may" have the relationship). Once resolved, it became a mandatory relationship in one direction: a *PATIENT* may participate in an *ENCOUNTER,* but an *ENCOUNTER* must have a *PATIENT* defined. In the same manner, a *PHYSICIAN* may participate in an *ENCOUNTER* with a *PATIENT*, but an *ENCOUNTER* must have a *PHYSICIAN* defined. In other words, for an *ENCOUNTER* occurrence to be properly defined according to the business rules, it must have both a *PATIENT* and a *PHYSICIAN*.

Avoid mandatory relationships in both directions. Usually these types of relationships should be regarded with suspicion since the business rules may have been represented incorrectly (4,8).

Usually, but not always, *one-to-one* relationships can be reduced, with one of the entities incorporated into the other. In most cases, one entity participating in this 1:1 relationship is just an attribute of the other.

The ERD examples we have used so far are simplifications of the medications domain. The data model in Figures 1.6 and 1.7 is incomplete: There is no representation of the relationship between the concepts of *Brand name* (*Tylenol,*

Aceta, Actamin) and *Generic name* (*Acetaminophen*); there is no *Drug Form* entity (*tablet, capsule, syrup, solution*), *Strength* information (*10 mg/ml, 50 mg/tablet*), *Indications, Contraindications, Precautions;* and there is no information related to *Maximal dose or Usual dose*. We build more accurate and complete models of the data structures in the following chapters.

Data Dictionary

Although it is lacking the graphical appeal of the other modeling tools mentioned and it is also quite tedious to build one, the data dictionary (DD) is nevertheless an important part of the system specifications. Without a DD, the DFD and ERD are just sketches; beautiful as they may be, their usefulness remains limited.

The DD is a central repository of information about the data and processes of the system under consideration. It is to be used by both users and developers as a communication tool to clarify terms during the system life cycle: analysis, design, code build, testing, implementation, training, and maintenance.

To keep in sync the various diagrams with the DD, a *computer-aided software engineering (CASE)* tool is highly recommended. A CASE tool can help automate most of the documentation process; for example, when drafting an ERD, a CASE tool can automatically create an entry in the DD for a newly defined entity. When a data store is mentioned on a DFD, CASE can automatically create an entry in the DD. A CASE tool can also minimize the inconsistency that may be introduced in the system specifications while detailing the models and components in multiple diagrams and the DD. A CASE-based DD can be searched for a term like any other electronic document, and it can generate reports, such as all the workflows or all the entities of a system. What should a DD keep information about?

Fields are the discrete data elements (also known as attributes or stored data items). A field is the smallest piece of information that has any meaning to the user. It is an elementary or *atomic* item, so from a user perspective, it is meaningless to try decomposing it any further. A Social Security number (SSN) may be considered an indivisible or atomic data element. Even if one could break the SSN further into three groups of numbers, it would probably be meaningless to the user. Gender is another example of a discrete, atomic data element since it would be pointless to try breaking down the *M* or *F* symbol any further. In a database table, the fields are the columns. The documentation of fields in the DD should include

Field name: *Patient heart rate*
Field definition: *Patient heart rate per minute*
Field aliases, synonyms, or alternate naming convention: *HR, H.R., Patient HR, Patient H.R.*
Field type and length: *Numeric/Integer*

Whether a null value is allowed: Can the field contain nothing? Is it a mandatory or an optional field?
Allowed values: For example, the system may allow in the *Gender* field only *Male*, *Female*, or *Undefined*. A *Patient heart rate* field may allow values between 0 and 300.
Default value: If no input is provided, what value should the system assign by default to the field? For the *Date Rx signed*, the default value may be the current date: *Today*. For *Time Rx signed*, the default value may be the current time: *Now*.
Source: What are the possible sources for the field? The sources for the *Patient heart rate* field could be *manual* (nurse input) or *from monitor* (through a medical device interface)
Security: which system, individual, or role has *Read/Write* access to the field.
A couple of examples and any other notes that may be helpful clarifying the concept.

Records are the structured combination of data items. In a database table, the records are the rows. For example, the data items *Patient ID, Patient last name, Patient middle name, Patient first name, Patient gender, Patient date of birth, Patient SSN* could be considered as a meaningful combination of related data elements in a *Patient demographics* record (row).

Data stores are the records storage structures (also known as tables or entities). In this category, we could include the repositories, aggregated clusters of databases such as the *CLINICAL REPOSITORY* mentioned. A DD will document the following for each data store:

Data store name: the name of the data store/entity as it appears on the DFD and ERD: *PHYSICIAN*
Data store definition and its purpose: *ENCOUNTER is the intersection entity between PATIENT and PHYSICIAN and contains information about the planned and actual location, planned and actual date and time frames for these encounters.*
Aliases and synonyms: *PATIENT ALLERGY* may be a.k.a. *Allergies, Drug allergies, Patient drug allergies.*
Table name: the name of the table as it appears on the physical model. The table name is not always the same as the entity name or data store in the conceptual model. The *CLINICAL REPOSITORY* data store in the DFD example in Figure 1.4 is actually a high-level representation of a cluster of data stores: *Demographics, Present and past diagnosis, Active problems list, Allergies, Lab values, Medications* etc. — to be implemented in the physical model in a number of tables within a number of databases.
Primary Key (PK)
Foreign Key(s) (FK)
Dataflows that may enter or leave the data store.

Volume: is it a high-volume data store (such as *Patient vital signs* captured every 30 seconds from monitors) or low-volume data store expected to rarely change — *Patient date of birth* or a slow changing dimension such as the *Patient name* or *address*.
Frequency: whether data store is accessed frequently or infrequently.
Security: which system, individual or role has *Read / Write* access to the data store.

Dataflows should be documented in the DD with:

Dataflow name as it appears on the DFD: *Rx details*
Dataflow description and purpose: *prescription details*
Dataflow aliases or synonyms: *Prescription*
Dataflow point of origin: *Process 1.2.1*
Dataflow destination: *Pharmacy*
Dataflow volume*: Low*
Dataflow frequency*: High*

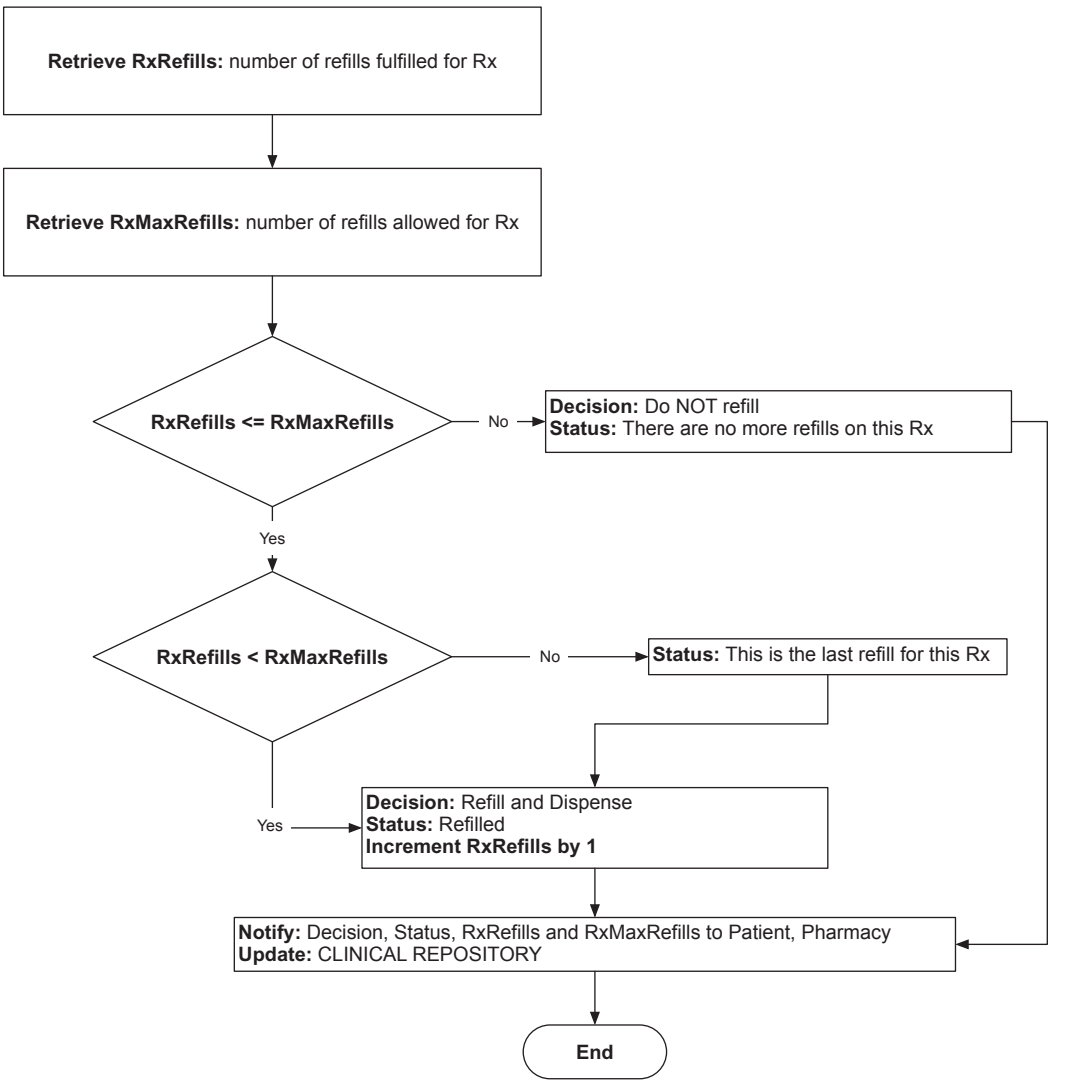
As mentioned in the DFD section of this chapter, the system *processes* are decomposed systematically in a top-down fashion from a high-level view of workflows into more detailed processes, activities, and tasks to the point of low-level, high-detail *functional primitives* (see next section). A DD should document the following for each process:

Process number and name as it appears on the DFD: *1.2.4. Consider Formulary*
Process description and purpose: *Check with Pharmacy Benefit Management (PBM) whether drug is covered by patient's pharmacy benefits plan and the corresponding drug formulary tier*
Process aliases or synonyms: *Query PBM on quantity limitations, Check drug tier, Check PBM coverage of drug*
Process input: *Patient ID, Drug ID*
Process output: *Drug coverage by PBM, Drug tier, Patient's co-pay, Limitations on quantity*

Functional Primitives Specification

In partitioning the workflows into processes, activities, and tasks, the *functional primitive* is the bottom-most bubble, which is reached when further decomposition would yield the function's detailed logic rather than additional child bubbles. The *functional primitive* is one step before the actual software code. It is used to detail tasks that should not be further decomposed into other activities or tasks. A functional primitive can be documented using flowcharts or *structured English* (also known as pseudocode). It is good practice to state for each functional primitive the *preconditions* to be satisfied before the actual processing as well as the

Precondition:
· A valid Rx exists



Post conditions:
· If Rx was refilled, then RxRefill was incremented by 1.
· If Rx was not refilled, RxRefill has not changed.
· The decision, status, RxRefill and RxMaxRefills have been notified to Patient and Pharmacy.
· CLINICAL REPOSITORY has been updated.

Figure 1.8 Flowchart example: Rx refill.

postconditions the system is expected to fulfill and accomplish once the processing is done.

Flowcharts can be used to document a functional primitive using three logical control structures: *sequence, selection,* and *iteration*.

What can be learned from the flowchart in Figure 1.8?

Before proceeding with the *Rx refill*, a precondition must be satisfied: There has to be a valid Rx.
System retrieves (*sequence*) from data stores two parameters for the uniquely identified Rx: number of refills allowed and number actually fulfilled/refilled.
System decides (*selection*) whether there are still refills on the Rx.
If there were more than one prescription to refill for the same patient, the system would also loop (*iteration*) through the process (not shown in Figure 1.8).
System updates *CLINICAL REPOSITORY* data stores and notifies *PATIENT* and *PHARMACY* on the refill status.

At the end of the flowchart, there is a set of postconditions that must be fulfilled by the system. If the postconditions have been fulfilled, then the processing of the *functional primitive* is to be considered correct and complete. Think about the postconditions as a safeguard checklist of facts the goals system needs to achieve once the flow through the chart is done. Pre- and postconditions are also a good starting point for designing test scenarios.

Structured English is a functional primitive so near the actual software code that it is also known as pseudocode:

> *Structured English is the use of the English language with the syntax of structured programming. Structured English aims at getting the benefits of both the programming logic and natural language. Program logic helps to attain precision while natural language helps in getting the convenience of spoken languages. (9)*

There are a couple of points to remember regarding structured English:

Use only the three control structures mentioned: *sequence, selection,* and *iteration*.
Use a (very) limited vocabulary of verbs and nouns and have all terms defined in the DD.
A functional primitives specification in pseudocode is usually less than one page long.
The highly detailed specification can be used to develop actual software code.

The following is an example of pseudocode:

```
For a given RxID
Precondition: A valid Rx exists
Retrieve RxRefills and RxMaxRefills
If RxRefills <= RxMaxRefills then
        If RxRefills < RxMaxRefills then
```

Dispense:
Decision: Refill and Dispense
Status: Refilled
Increment RxRefills by 1
Proceed: Notify and Update
Else
Status: This is the last refill on this Rx
Proceed: Dispense
Else
Decision: Do NOT refill
Status: There are no more refills on this Rx
Proceed: Notify and Update
Notify and Update:
NoteRx: Date and time (now), Decision, Status, RxRefills, Rx MaxRefills
Send NoteRx to Patient
Data Stores: Clinical repository
Store NoteRx in Data Stores
End

Since the book does not use this level of granularity, *structured English* is not discussed further.

Balancing the Models

When modeling a system from different perspectives (data, functions, sequence, conditional logic) as well as maintaining a dictionary and a set of functional primitives specs, there is always a concern with developing *inconsistent* interpretations of the reality the system is supposed to model. A fact mentioned in one model may be missing from the others. Terms and concepts appearing in a DFD may be missing from the DD or vice versa. Naming conventions may not be strictly enforced, and the same concept may have slightly different names (singular vs. plural is a frequent "suspect" issue, such as *PATIENT* vs. *PATIENTS*).

Errors introduced at an early phase such as during systems analysis are likely to be propagated and magnified during the system life cycle. These errors become notoriously more expensive to correct further down the road. So, it makes sense to try eliminating at an early stage as many errors as possible from the models we create. Balancing the models is one way to minimize errors and inconsistency in the analysis deliverables. There are several straightforward techniques recommended for balancing the different models (3): DFD versus DD, ERD versus DD, DFD versus functional primitives' specifications, and ERD versus DFD and functional primitives' specifications.

DFD versus DD

Every dataflow, process, and data store appearing in a DFD must be defined in the DD.
Every dataflow, process, and data store defined in the DD must appear in a DFD.

ERD versus DD

Every data store appearing in the ERD must be defined in the DD.
Every data store defined in the DD must appear in the ERD.

DFD versus Functional Primitives' Specifications

Every bubble in the DFD must be associated with a lower-level DFD or a functional primitive specification, but not both.
Every functional primitive specification must have a bottom-level bubble in a DFD.
Inputs and outputs related to a bubble must match the inputs and outputs (READ and WRITE statements) in the functional primitive spec.

ERD versus DFD and Functional Primitives' Specifications

Every data store in a DFD must have a representation in the ERD.
An entity in the ERD must have a data store representation in a DFD.
Every data store in a functional primitive specification must have a representation as an entity in the ERD.
An entity in the ERD must have a representation on a functional primitive spec.

The techniques mentioned for balancing the models require little intelligence or creativity. Still, balancing the models is a necessity for the modeling deliverables to be considered complete, correct, and consistent. It is a tedious, time-consuming activity and thus well suited for a machine: a CASE tool.

Summary

This chapter introduced a methodology and a set of tools. The methodology—structured system analysis or SSA—recommends focusing separately and in parallel on the data and functions aspects of a system. SSA provides the tools to model a system from these two perspectives: ERD and DFD. In addition, the technical specifications of a system (but not this book) should define a DD, a set of control processes, state transition diagrams as well as flowcharts, and structured English/pseudocode.

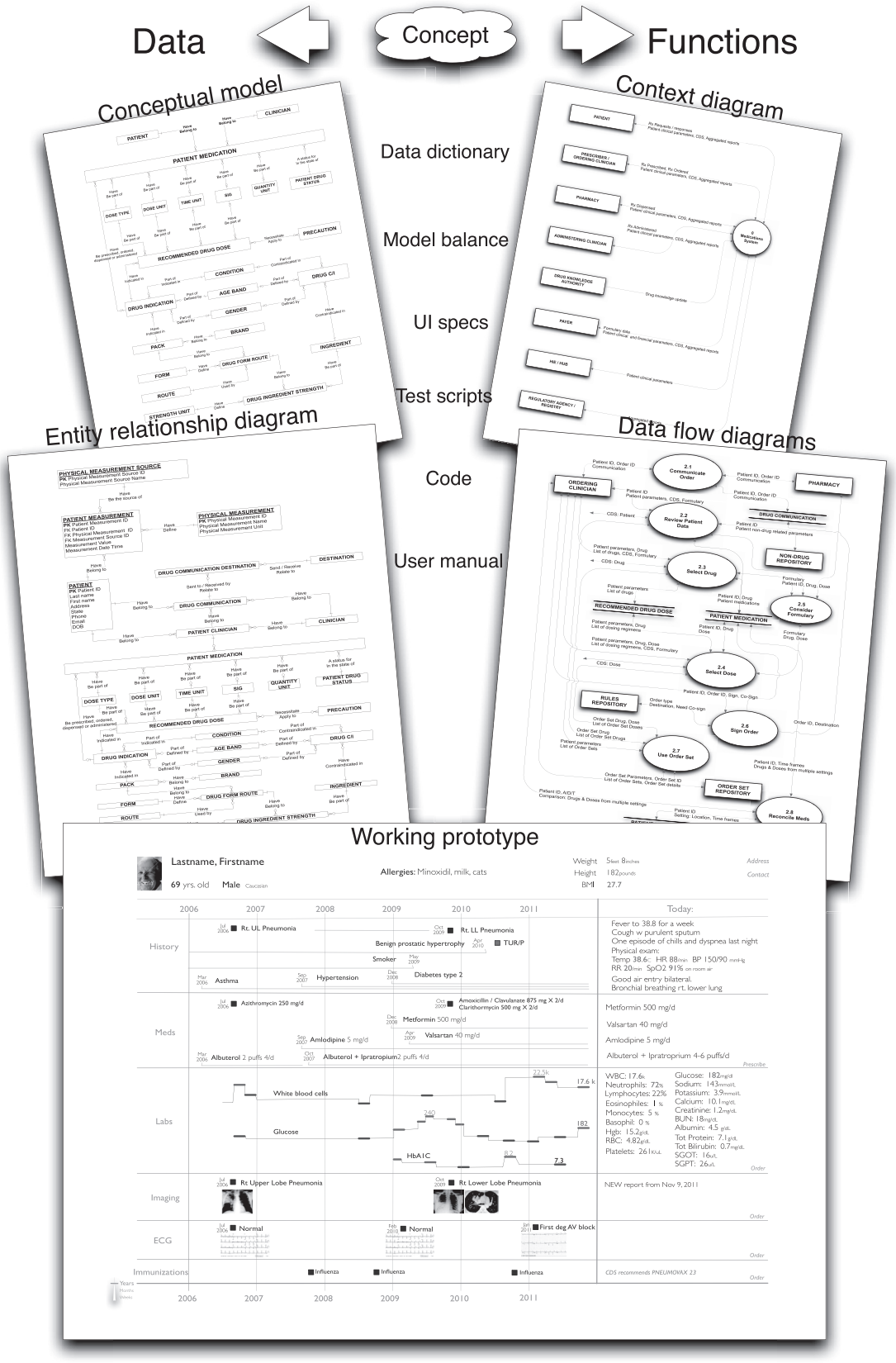


Figure 1.9 Summary of SSA.

There is an increasing level of details while evolving from the conceptual to logical to the physical data model and from the context diagram to DFDs and to functional primitive specifications. Thus, the audiences for these models may differ: The conceptual model, context diagram, and first level of a DFD may be considered appropriate high-level communication tools for business stakeholders, directors, and product/project managers. The logical, physical models, low-level DFD, and pseudocode delve into more details and should be used for in-depth discussions between users, programmers, application engineers, and database designers.

Modeling a system is an iterative, recursive activity. Models are rarely perfect the first time drafted, and usually they need to be refined several times before they can be considered correct, complete, and consistent. Once done, models are not static artifacts, and they need to be updated as often as the system requirements may change during the life cycle.

In the next chapters, we use the principles and tools introduced here while systematically analyzing the medications domain. We show that this ever-daunting analysis job is much more fun with diagrams than text and spreadsheets.

References

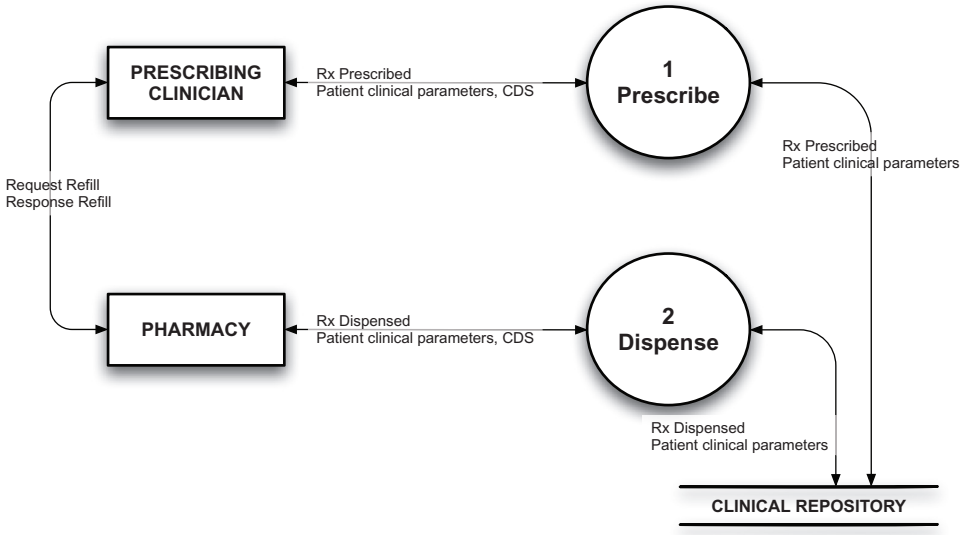
1. Merriam-Webster online dictionary. http://www.merriam-webster.com/dictionary/system (accessed December 2, 2010)
2. Wiegers, K.E. 1989. *Software requirements*. Microsoft Press.
3. Yourdon, E. 1989. *Modern structured analysis*. Upper Saddle River, NJ: Prentice-Hall.
4. Simsion, G.C., and G.C. Witt. 2005. *Data modeling essentials*. New York: Elsevier.
5. Fowler, M., and K. Scott. 2000. *UML distilled*. Boston: Addison-Wesley.
6. Shelly, G.B., T.J. Cashman, and H.J. Rosenblatt. 2008. *Systems analysis and design*. Boston: Thomson Course Technology.
7. Rob, P., and C. Coronel. 2009. *Database systems. Design, implementation and management*. Boston: Thomson Course Technology.
8. Silverston, L., and P. Agnew. 2009. *The data model resource book—Universal patterns for data modeling*. New York: Wiley.
9. Wikipedia. Structured English. http://en.wikipedia.org/wiki/Structured_English (accessed December 2, 2010).

Review Questions

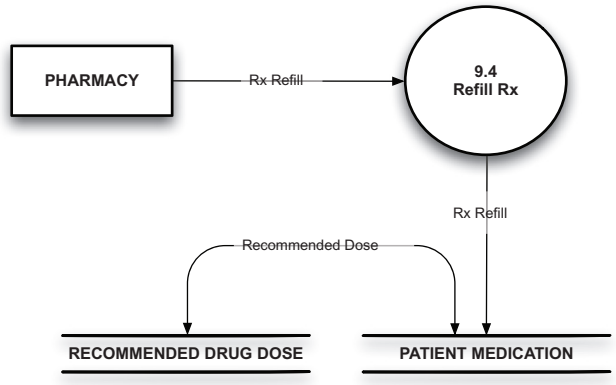
1. List five reasons why narrative text is not the best tool for systems analysis.
2. List six characteristics of structured systems analysis.

Find the errors in the following diagrams:

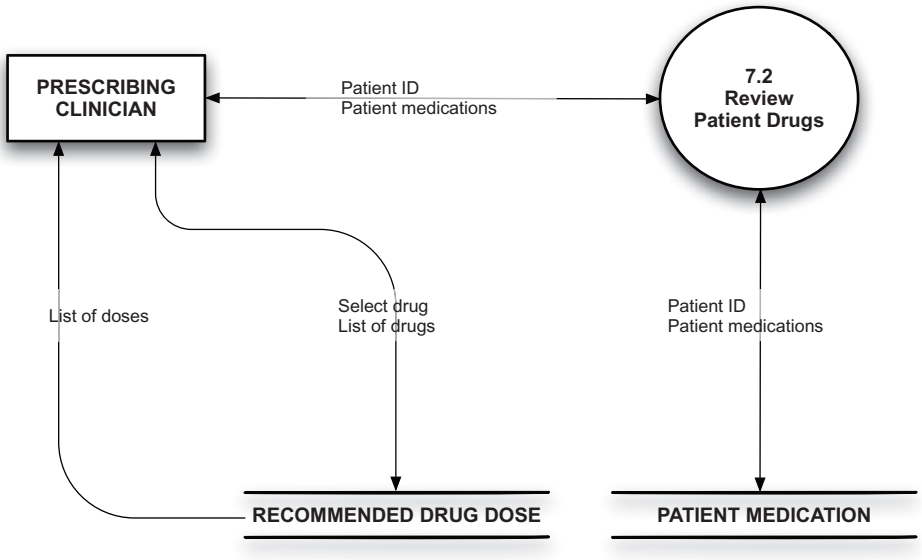
3.



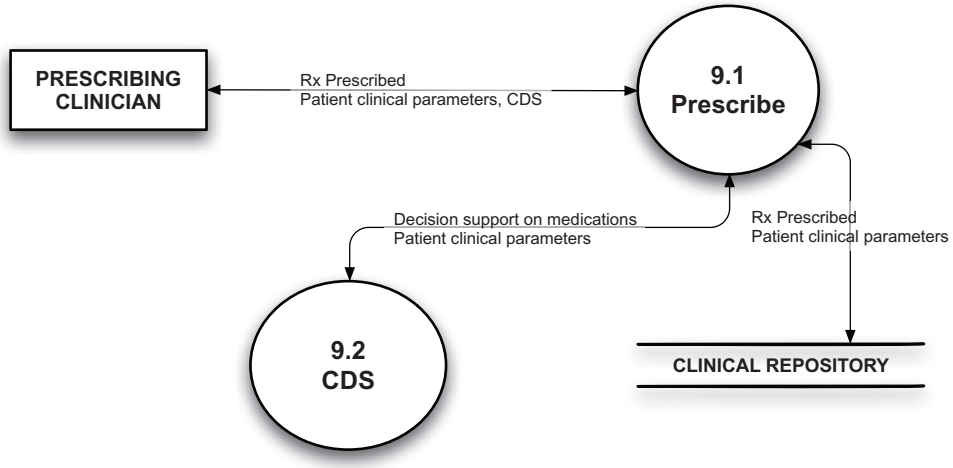
4.



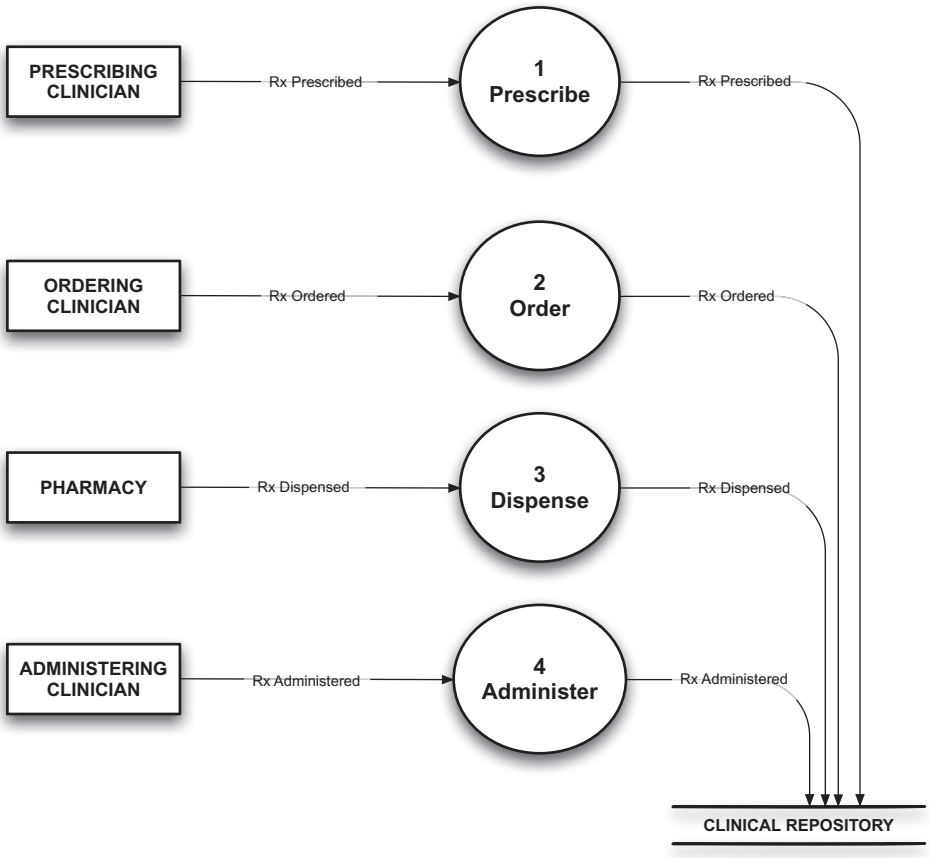
5.


PRESCRIBING CLINICIAN
Patient ID
Patient medications
7.2
Review
Patient Drugs
List of doses
Select drug
List of drugs
Patient ID
Patient medications
RECOMMENDED DRUG DOSE
PATIENT MEDICATION

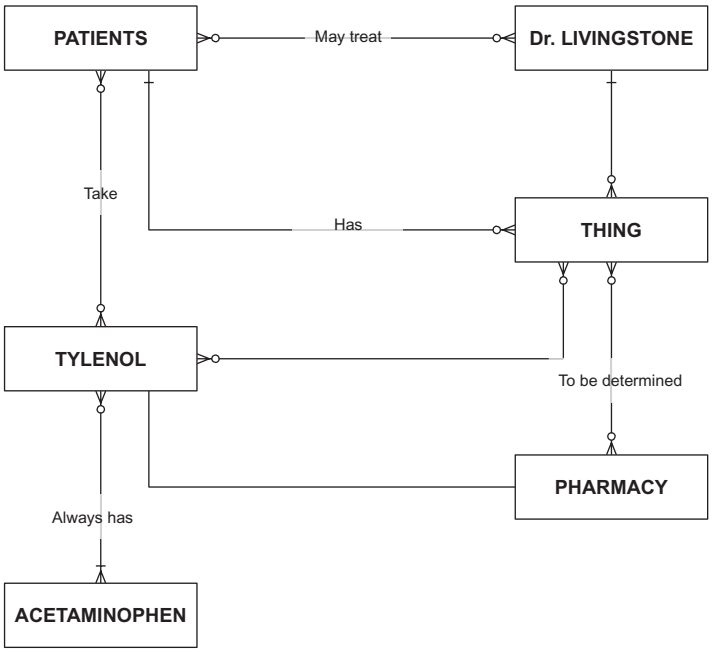
6.


PRESCRIBING CLINICIAN
Rx Prescribed
Patient clinical parameters, CDS
9.1
Prescribe
Decision support on medications
Patient clinical parameters
Rx Prescribed
Patient clinical parameters
9.2
CDS
CLINICAL REPOSITORY

7.



8. List the processes in the order defined in Figure 1.4.
9. Find all the errors in:



10. Draft the following relationship at the conceptual level and then resolve it for a logical model:
    - An ingredient may be part of many drugs.
    - A drug may have many ingredients.
    - How is the PK of the new entity called?

References

1 Chapter 1: Short Primer on Structured Systems Analysis

26 ▪ 5. PATIENT MEDICATIONRECOMMENDED DRUG DOSE Select drug List of drugs List of doses PRESCRIBING CLINICIAN 7.2 Review Patient Drugs Patient ID Patient medications Patient ID Patient medications 6. Rx Prescribed Patient clinical parameters, CDS Rx Prescribed Patient clinical parameters Decision support on medications Patient clinical parameters 9.1 Prescribe PRESCRIBING CLINICIAN 9.2 CDS CLINICAL REPOSITORY ▪ 7. Rx Prescribed Rx Ordered Rx Administered 1 Prescribe PRESCRIBING CLINICIAN ORDERING CLINICIAN PHARMACY ADMINISTERING CLINICIAN 2 Order 3 Dispense 4 Administer CLINICAL REPOSITORY Rx Dispensed Rx Prescribed Rx Ordered Rx Dispensed Rx Administered 8. List the processes in the order defined in Figure 1.4. 9. Find all the errors in: Has Take Always has May treat PATIENTS Dr. LIVINGSTONE THING TYLENOL PHARMACY ACETAMINOPHEN To be determined

28 ▪ 10. Draft the following relationship at the conceptual level and then resolve it for a logical model: - An ingredient may be part of many drugs. - A drug may have many ingredients. - How is the PK of the new entity called?

72 ▪ 4. INGREDIENT PACK Have Indicated in Have Contraindicated in Part of Defined by Part of Indicated in Part of Contraindicated in CONDITION AGE BAND DRUG INDICATION DRUG C/I GENDER Part of Defined by Part of Defined by Part of Defined by 5. RECOMMENDED DRUG DOSE TIME SIG PRECAUTION +DYH ,QGLFDWHG␣LQ 1HFHVVLWDWH $SSO\␣WR +DYH %H␣SDUW␣RI DOSE +DYH %H␣SDUW␣RI +DYH %H␣SDUW␣RI DOSE TYPE PATIENT PATIENT MEDICATION CLINICIAN PATIENT DRUG STATUS Have Belong to Have Belong to $␣VWDWXV␣IRU ,Q␣WKH␣VWDWH␣RI +DYH %H␣SDUW␣RI +DYH %H␣SDUW␣RI +DYH %H␣SDUW␣RI +DYH %H␣SDUW␣RI QUANTITY +DYH %H␣SDUW␣RI DRUG FORM ROUTE BRAND PACK +DYH %HORQJ␣WR FORM ROUTE +DYH 'HILQH +DYH 8VHG␣E\ +DYH %HORQJ␣WR INGREDIENT STRENGTH +DYH %H␣SDUW␣RI +DYH 'HILQH +DYH %HORQJ␣WR DRUG INGREDIENT STRENGTH +DYH␣ ,QGLFDWHG␣LQ +DYH␣ &RQWUDLQGLFDWHG␣LQ 3DUW␣RI 'HILQHG␣E\ 3DUW␣RI ,QGLFDWHG␣LQ 3DUW␣RI &RQWUDLQGLFDWHG␣LQ CONDITION AGE BAND DRUG INDICATION DRUG C/I GENDER 3DUW␣RI 'HILQHG␣E\ 3DUW␣RI 'HILQHG␣E\ 3DUW␣RI 'HILQHG␣E\ +DYH %H␣SUHVFULEHG␣RUGHUHG␣ GLVSHQVHG␣RU␣DGPLQLVWHUHG +DYH %H␣SDUW␣RI The next three questions refer to Figure 2.12: ▪ 6. Explain why the relationship (near the CONDITION side) between CONDITION and DRUG INDICATION is mandatory while that between CONDITION and DRUG C/I is optional. 7. Explain why the relationships between GENDER and DRUG INDICATION as well as between GENDER and DRUG C/I are optional near the GENDER entity. 8. Explain why the relationships of TIME UNIT, SIG, and QUANTITY UNIT are optional and not mandatory near these entities. 9. The recommended dose of acetaminophen for infants is up to 60 mg/kg/ day (10–12 mg/kg/dose and up to 5 doses/day). The child has a fever and weighs 22 pounds. If the acetaminophen solution is 100 mg/ml for use by mouth and is dispensed as a bottle of 30 ml, – How many milligrams can be given in each dose? – How many milliliters for each dose? – What is the total number of milligrams per day that can be administered safely to this child? – What is the maximal number of milliliters per day that can be given? – How many doses are there in one dispensed bottle? 10. In example 9, indicate the following for the acetaminophen: – Form – Strength – Route – Recommended dose – Frequency – Maximal dose – Dispensed original package

3 Chapter 3: Prescribe/eRx

106 Review Questions Find the errors in the following four diagrams: 1. RECOMMENDED DRUG DOSE PRESCRIBING CLINICIAN /LVW▯RI▯GUXJV 1.3.1 Select New Drug &'6▯'UXJ)RUPXODU\▯GDWD 6HOHFW▯GUXJ /LVW▯RI▯GUXJV▯&'6▯)RUPXODU\ 6HOHFW▯GUXJ▯5HILOO▯0RGLI\▯5HQHZ▯6WRS /LVW▯RI▯GUXJV PATIENT MEDICATION 2. PRESCRIBING CLINICIAN 1.3.1.1 Search by Indication ▪ 3. RECOMMENDED DRUG DOSEPATIENT MEDICATION Select dose, Frequency, Duration, Quantity, Number of refills, SIG Dosing regimens PRESCRIBING CLINICIAN 1.4.2 Select Dose CDS: Dose Rx details Limits on duration or quantity 1.4.1 Review Drug Dose Dosing regimens Dose, Frequency, Duration, Quantity, Number of refills, SIG 4. PBM 1.5.1 Retrieve PBM Data on Patient 3DWLHQW▯,' (OLJLELOLW\▯ 1.5.2 Retrieve PBM Data on Drug 1.5.3 Retrieve PBM Data on Quantity 'UXJ 7LHU▯$OWHUQDWLYHV▯&R▯SD\▯ 'RVH 4XDQWLW\▯'XUDWLRQ▯OLPLWV▯ 3DWLHQW▯,' (OLJLELOLW\▯ 'UXJ 7LHU▯$OWHUQDWLYHV▯&R▯SD\ 'RVH 4XDQWLW\▯'XUDWLRQ▯OLPLWV▯ PBM PLAN DETAIL 3DWLHQW▯,'▯'UXJ 7LHU▯$OWHUQDWLYHV▯&R▯SD\▯4XDQWLW\▯'XUDWLRQ▯OLPLWV

108 5. Explain why CDS does not appear in Figure 3.2. 6. Explain why process 1.5 Consider Formulary does not appear in Figure 3.6. 7. Explain what happened to CLINICAL REPOSITORY: It appears in Figure 3.1 and does not appear in Figure 3.2. 8. Are the processes numbered correctly in the following figure? 6HOHFW▯GUXJ▯ 7LHU▯$OWHUQDWLYHV▯&R▯SD\ 6HOHFW▯GRVH▯ 4XDQWLW\▯RU▯GXUDWLRQ▯OLPLWV PRESCRIBING CLINICIAN PBM 1.3 Select Drug 1.4 Review Patient Data 1.5 Consider formulary 3DWLHQW▯,'▯5[▯GHWDLOV (OLJLELOLW\▯)RUPXODU\▯GDWD 6HOHFWHG▯GUXJ 7LHU▯$OWHUQDWLYHV▯&R▯SD\ 3DWLHQW▯,' (OLJLELOLW\ 1.2 Select Dose 3DWLHQW▯,' (OLJLELOLW\▯ 6HOHFWHG▯GRVH▯DQG▯TXDQWLW\ 4XDQWLW\▯RU▯GXUDWLRQ▯OLPLWV 9. Which table provides the clinician with the patient’s name, address, date of birth, and gender? 10. Which entity or actor provides the clinician with knowledge on safer therapeutic alternatives? Less-expensive alternatives?

4 Chapter 4: Order/CPOE

136 Review Questions 1. Identify the possible sources of information for an order drug and dose. 2. Which workflows or processes may alter, filter, or categorize the information on drugs and doses included in an order? 3. Which processes from Prescribe workflow are good candidates for reuse in Order workflow? 4. What are the main differences between Prescribe and Order workflows? 5. During Order workflow: Who is the source of Formulary information? Which process is responsible for retrieving the Formulary information? 6. What are the main categories of information in a typical Order Set? 7. What are the benefits of using Order Sets? 8. In Figure 4.16, a communication on a drug may be communicated to many destinations, but all with the same priority level. How would you change the model to accept for each destination a possible different priority? For example: Pharmacy—High Priority, Dr. X—Medium Priority, MAR—Low priority. 9. Calculate the number of rows in the table PATIENT MEASUREMENT after 1 year, assuming the following parameters: 100 patients/year, 10 vital signs parameters monitored continuously for each patient, and each value stored every 1 minute. 10. Show in the logical model the following data elements: Patient X, 20110403 22:39:12, Weight 2.4 kg, Source: nurse weighed patient. Patient Y, 20110413 10:03:09, Weight 180 pounds, Height 5 feet 6 inches, Source: patient.

152 9. Institute for Safe Medication Practices. Institute for Safe Medication Practices (ISMP) guidance on the interdisciplinary safe use of automated dispensing cabinets. http://www.ismp.org/tools/guidelines/ADC/default.asp (accessed August 14, 2011). 10. Murray, M.D. 2001. Automated medication dispensing devices. In Making health care safer: A critical analysis of patient safety practices, Evidence Report/ Technology Assessment, No. 43. http://archive.ahrq.gov/clinic/ptsafety/chap11.htm (accessed August 16, 2011). ▪ Review Questions 1. Sketch a DFD with the Patient, Prescribing, Ordering, Dispensing, and Administering Clinicians communication processes 2. Billing as an entity is not shown in DFD 3. Which entity may provide information on drugs dispensed for Billing to capture charges? 3. The pharmacist needs to review the patient's renal and hepatic functions as reflected in the values of a laboratory analysis. Which entity is the source for this information? 4. Where does a pharmacy receive information on a patient's eligibility? 5. A medication is out of stock. Which process checks drug availability? Which entity provides this information? 6. A therapeutic equivalent is being considered. Which process recommends equivalent drugs? Which entity provides the information? 7. A robot is loaded with the wrong drug. Later, the robot dispenses the drug and labels it with a bar code. Which pharmacy process may prevent the wrong drug reach the patient? 8. Which process/entity provides information on limits on a drug quantity or treatment duration? 9. A patient's condition has recently changed. Which process will alert the pharmacist to the new condition, related contraindications, and any precautions that need to be considered? 10. A pharmacy sends a SMS (Short Message Service) text to patients' cellular phones, letting them know their prescription is ready for pickup. Which data store is the source for this information?

6 Chapter 6: Administer/eMAR

184 Review Questions 1. In a fully automated environment and assuming the nurse is already logged in, what is the minimal number of steps to validate the 5 Rights and activate an infusion? 2. Which aspects of DFD 4 pertain to a Personal Health Record (PHR)? 3. How is Pharmacy communicating with the Administering Clinician? 4. While loading a pump with drug A, a nurse erroneously documents drug B in the SMART PUMP system. How can the Medications system prevent this error? 5. If a patient weights 110 pounds and the dose is 1.2 mcg/kg/min, what is the drug rate in milligrams/hour? 6. A neonate weighting 3 kg has an order for an Adrenaline (Epinephrine) infusion at 0.2 mcg/kg/min. The pharmacy has dispensed 1.8 mg Adrenaline in a 100-ml bag. What would be the pump drop rate? 7. What problem should be expected with a drug administered at 0.8 ml/hour with a drop-based pump? 8. Which drugs may a patient document as self-administered? 9. A drug at a strength of 10 mg/ml is administered at 6 drops/min to a patient weighing 100 kg. What is the dose in micrograms/kilogram/minute? 10. A nurse receives a solution of a drug with a strength of 0.001 mcg/ml. Concerned that the diluted solution would cause an overload of fluids, the nurse concentrates the solution so the same quantity can be administered in a smaller volume. Explain the calculations involved.

7 Chapter 7: User Interface

216 11. Tufte, E.R. 2006. Beautiful evidence. Cheshire, CT: Graphics Press. 12. Powsner, S.M., and E.R. Tufte. 1994. Graphical summary of patient status. Lancet 344: 386–389. 13. Wong, D.M. 2010. The Wall Street Journal guide to information graphics: The dos and don’ts of presenting data, facts and figures. New York: Norton. 14. Lidwell, W., K. Holden, and J. Buttler. 2003. Universal principles of design. 125 ways to enhance usability, influence perception, increase appeal, make better design decisions and teach Through design. Minneapolis, MN: Rockport. 15. Tidwell, J. 2006. Designing interfaces. Sebastopol, CA: O’Reilly. 16. Scott, B., and T. Neil. 2009. Designing web interfaces. Sebastopol, CA: O’Reilly. 17. Bertin, J. 1983. Semiology of graphics, 214–215. Redlands, CA: Esri Press. ▪ Review Questions 1. Product A: 88% of trained users select the correct drug and dose in less than a minute using 12 clicks. Product B: 86% of trained users select the correct drug and dose in less than a minute using 4 clicks. Which EHR is more effective, and which one is more efficient? 2. What are the potential UI issues in the following figure? History Physical exam Diagnosis Problems list Medications Prescriptions Labs Imaging Procedures Consultations Reporting Billing CDS Formulary Medications Tab Glucophage 500 mg/tab 1 tab/day Tab Diovan 40 mg/tab 1 tab/day Tab Norvasc 2.5 mg/tab 2 tab/day Inh Combivent 21/120 mcg/puff 4 puffs/day 3. What difficulties should one expect with the UI in the following figure? History Physical exam Diagnosis Problems list Medications Prescriptions Labs Imaging Procedures Consultations Reporting Billing Sodium Result 140 Range 133-143 Unit mmoL/L Potassium Chloride 4.3 103 3.6-5.2 98-112 mmoL/L mmoL/L *Glucose BUN 139 14 70-100 7-18 mg/dL mg/dL *Creatinine Calcium 1.4 10.1 0.6-1.3 8.8-10.5 mg/dL mg/dL *Troponin-I Total bilirubin 1.8 0.5 0.0-0.06 0.0-1.0 ng/mL mg/dL Labs 01/12/2011

218 4. What may interrupt the clinician’s cognitive processes in the following figure? History Physical exam Diagnosis Problems list Medications Prescriptions Labs Imaging Procedures Consultations Reporting Billing *Glucose Result 140 Range 70-100 Unit mg/dL *Glucose *Glucose 488 520 70-100 70-100 mg/dL mg/dL Labs 01/12/2011 History Physical exam Diagnosis Problems list Medications Orders Labs Imaging Procedures Consultations Reporting Billing Orders 01/12/2011 Time 02:22 04:10 05:20 05:20 06:02 07:10 07:10 08:48 09:35 *Glucose *Glucose 378 230 70-100 70-100 mg/dL mg/dL *Glucose *Glucose 192 112 70-100 70-100 mg/dL mg/dL IV Reg Insulin Stop Dr. S 07:35 IV Reg Insulin

4 u/hour Dr. A 07:10 IV Reg Insulin 8 u/hour Dr. A 06:02 IV Reg Insulin 10 u/hour Dr. B 05:29 IV Reg Insulin 6 u/hour Dr. A 04:12 IV KCl 5 meq/hour Dr. B 05:29 Potassium 3.9 3.6-5.2mmoL/L *Potassium 3.4 3.6-5.2mmoL/L 5. Assuming there are 3,600 discrete data elements to be displayed for a specific time frame, how many screens need to be reviewed if the average data density is (a) 120 elements/screen? (b) 1,800 elements/screen? 6. How would you improve the UI in the following figure? 7. Why is the system assuming a rectal suppository should be used in the right eye according to the figure in Question 6? What went wrong? ▪ 8. Tab Diovan comes in strengths of 40, 80, 160, and 320 mg/tab. Why is the Diovan strength drop-down menu showing different values in the figure in question 6? 5HFWDO VXSSRVLWRU\ 9ROWDUHQ PJ VXSS SHU GD\ 7DE 'LRYDQ SHU GD\ 32 PJ WDE PJ WDE PJ WDE PJ WDE PJ WDE PJ WDE PJ WDE PJ WDE PJ WDE PJ WDE 5W (\H What would you suspect is not working correctly? 9. How would you improve the UI of the patient banner in the following figure? 10. “User should be in a good mood when done.” Why wouldn’t a clinician be in a good mood viewing the display in the following figure?

8 Chapter 8: Clinical Decision Support

260 Review Questions 1. What are the consequences of false-positive and false-negative alarms? 2. Consider two populations: healthy and sick. When new laboratory test results are charted, the distributions of the two populations are noted to overlap. How can we minimize the new test's false-positive and false-negative rates? 3. A patient is taking warfarin (an anticoagulant) at home. During admission to the hospital, a physician orders heparin, also an anticoagulant. Which CDS processes are expected to be involved in the decision-making process and alert the physician about the potential danger of the combination? 4. A patient is on eight different medications. The physician considers adding two antibiotics to treat his pneumonia. How many pairs of medications should the drug-drug interaction process consider? 5. A patient is prescribed two types of insulin: regular and slow acting. Which process decides whether the combination is appropriate or should be considered duplicate therapy? 6. A patient with end-stage renal disease needs an antibiotic known to be excreted by the kidney. Which CDS processes should advise the physician prescribing the drug? 7. Green leafy vegetables are known to reduce the effectiveness of the anticoagulant warfarin. Which process would be responsible to inform a physician about this interaction? 8. Acitretin is potentially harmful to the fetus if taken by the mother. The CDS used by the physician fires a critical alert when the doctor tries to prescribe this drug to a 16-year-old male patient. What should be configured differently in the CDS setup? 9. Erythromycin and ketoconazole may interact, with the result being a dangerously high blood level of erythromycin. What are the possible CDS recommendations related to this interaction? What would be considered excessive and thus the CDS should not contemplate? 10. A female patient on promethazine has her positive pregnancy test flagged as suspect and potentially false positive. Which CDS process decided to alert about this possible wrong result?

9 Chapter 9: Report

288 Review Questions 1. What are the main types of health-related reports? 2. What are the financial incentives for reporting under the PQRS and MU initiatives? 3. What are the three main aspects of healthcare quality according to the Donabedian model? 4. What are the IOM-declared goals for the U.S. healthcare system? 5. Consider the following quality measure: PQRS # 170: Coronary Artery Bypass Graft (CABG): Beta-Blockers Administered at Discharge Percentage of patients aged 18 years and older undergoing isolated CABG surgery who were discharged on beta-blockers. Using the following data calculate the quality measure: Total number of patients aged 18 years and older during 2011 = 13,200. Of these, 2,883 were diagnosed with heart disease; 594 patients had CABG surgery: 22 patients also had mitral valve repair, 15 also had tricuspid valve repair, and 5 patients had CABG and mitral and tricuspid valve repair during the same surgery. Eight patients were transferred to another facility, 13 had died, and 31 had a contraindication to beta-blockers. There were 380 patients on betablockers at admission, and 420 patients received this treatment at discharge. 6. List the current quality reporting methods from the most to the least efficient. 7. Why is a data warehouse needed for reporting? 8. What are the three main schemas for dimensional data stores? 9. What are the main types of SQL operations used for reporting purposes? 10. Assume there are 5,000 physicians in an organization, which reports on 240 quality measures using a three-dimensional cube as its data mart. How many cells would accumulate in this cube after 1 year if the quality measures are stored once per month per physician?

10 Chapter 10: Interoperability Standards and Vocabularies

312 Review Questions 1. What levels of interoperability, according to LCIM, must be reached before achieving conceptual interoperability? 2. Follow a message through the layers of the OSI model from sender to receiver. 3. What are the dimensions of an ontology? 4. Assume each entity needs to communicate with all the others in the following list: Emergency Department, Surgery, Internal Medicine, Pediatrics, Lab, Pathology, Imaging, Operating Room, Postanesthesia Care Unit, Intensive Care Unit, Neonatal Intensive Care Unit, Cardiology, Endocrinology, Psychiatry, Dietary Services, Physiotherapy, Billing, Medical Records. How many interfaces are needed? 5. List the qualities of a modern medical terminology according to Cimino. 6. In the following figure, draw the lines between clinical domains and the respective standards and vocabularies as applied in the United States. Diagnosis Problem List History & Physical exam Lab Medications Imaging Procedures Communications Bed side devices Summaries Reporting Eligibility, claims Privacy & Security HL7 ICD LOINC NCPDP RxNorm CPT 4 IEEE/ISO PQRI ASC X.12 HIPAA XML: CCD, CDA, CCR SNOMED NDC UNII UCUM NDF-RT DICOM NIST/FIPS DRG CLINICAL DOMAINS STANDARD/VOCABULARY ▪ 7. Reporting to immunization registries and other quality programs is an expected part of HIE. Explain why reporting issues are not detailed in DFD 7. 8. List the issues HIE participants need to agree on when joining a RHIO. 9. List the tables in the Medications system ERD that may be involved in HIE in the following scenario: New patient data arrive from RHIO just as a patient is being admitted to the hospital: patient name, address, gender, date of birth, height, weight, medications taken at home, drugs and doses just ordered by the admitting physician. 10. List the tables in the ERD that a Drug Knowledge Authority may update periodically.