SECTION III

Nursing Informatics Administrative Applications: Precare and Care Support

Nursing informatics (NI) and information technology (IT) have invaded nursing, and some nurses are happy with the capabilities afforded by this specialty. Others, however, remain convinced that changes wrought by IT are nothing more than a nuisance. In the past, nursing administrators have found the implementation of technology tools to be an expensive venture with minimal rewards. This is likely related to their lack of knowledge about NI, which caused nursing administrators to listen to vendors or other colleagues; in essence, it was decision making based on limited and biased information. There were at least two reasons for the experience of limited rewards. One was the failure to include nurses in the testing and implementation of products designed for nurses and nursing tasks. Second, the new products they purchased had to interface with old, legacy systems and were not at all compatible or seemed compatible until the glitches arose. The glitches caused frustration for clinicians and administrators alike. They purchased tools that should have made the nurses happy, but instead all they did was grumble. The good news is that approaches have changed as a result of the difficult lessons learned from the early forays into technology tools. Nursing personnel are involved both at the agency level and the vendor level, in the decision making and development of new systems and products charged with enhancing the practice of nursing. Older legacy systems are being replaced with newer systems that have more capacity to interface with other systems. Nurses and administrators have become more astute in the realm of NI, but there is still a long way to go. Chapter 12 introduces the system development life cycle, used to make important and appropriate organizational decisions for technology adoption.

Administrators need information systems that facilitate their administrative role, and they particularly need systems that provide financial, risk management, quality assurance, human resources, payroll, patient registration, acuity, communication, and scheduling functions. The administrator must be open to learning about all of the tools available. One of the most important tasks that an administrator can oversee and engage in is data mining, or the extraction of data and information from sizeable data sets collected and warehoused. Data mining helps to identify patterns in aggregate data, gain insights, and ultimately discover and generate knowledge applicable to nursing science. Nursing administrators must become astute informaticists—knowledge workers who harness the information and knowledge at their fingertips to facilitate the practice of their clinicians, improve patient care, and advance the science of nursing. Clinical information systems (CIS) have traditionally been designed for use by one unit or department within an institution. However, because clinicians working in other areas of the organization need access to this information, these data and information are generally used by more than one area. The new initiatives arising with the develop-

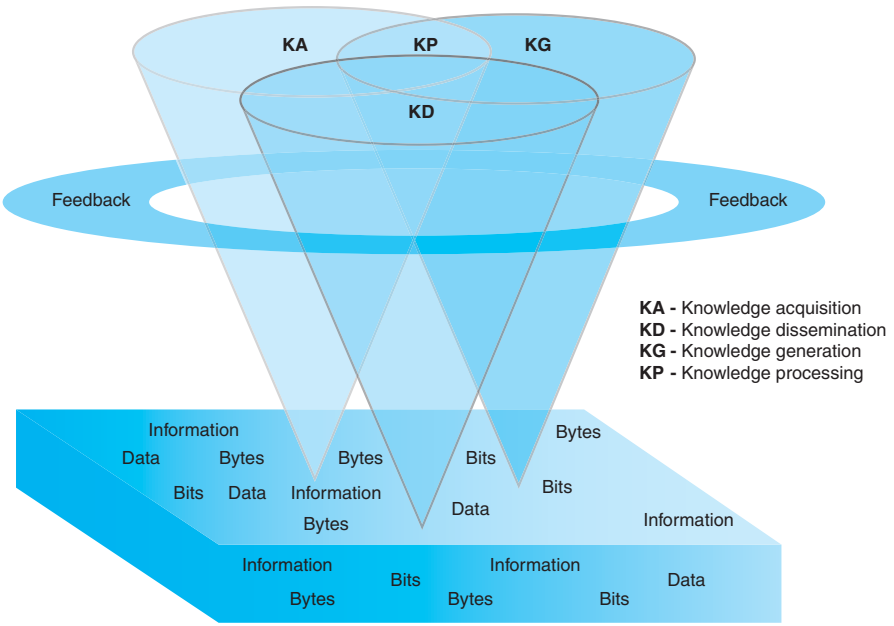
ment of the electronic health record place institutions in the position of striving to manage their CIS through the electronic health record. Currently, there are many CISs, including nursing, laboratory, pharmacy, monitoring, and order entry, plus additional ancillary systems to meet the individual institutions' needs. Chapter 13 provides an overview of administrative information systems and helps the reader to understand powerful data aggregation and data mining tools afforded by these systems.

Chapter 14, related to improving the human–technology interface, discusses the need to improve quality and safety outcomes significantly in this country. Through the use of IT, the designs for human–technology interfaces can be radically improved so that the technology better fits both human and task requirements. A number of useful tools are currently available for the analysis, design, and evaluation phases of development life cycles and should be used routinely by informatics professionals to ensure that technology better fits both task and user requirements. In this chapter, the author stresses that the focus on interface improvement using these tools has had a huge impact on patient safety in one area of health care: anesthesiology. With increased attention from informatics professionals and engineers, the same kinds of improvements should be possible in other areas. This human–technology interface is a crucial area if the theories, architectures, and tools provided by the building block sciences are to be implemented.

Each organization must determine who can access and use their information systems, and provide robust tools for securing information in a networked environment. It is also imperative that nurses understand copyright and fair use rules as they apply to both written and electronic information. Chapter 15 introduces these important safeguards for protecting information. As new technologies designed to enhance patient care are adopted, barriers to implementation and resistance by practitioners to change are frequently encountered. Chapter 16 provides insights into clinical workflow analysis and provides advice on improving efficiency and effectiveness to achieve meaningful use of caring technologies.

Pause to reflect on the Foundation of Knowledge Model (Figure III-1) and its relationship to both personal and organizational knowledge management. Consider that organizational decision making must be driven by appropriate information and knowledge developed in the organization and applied with wisdom. Equally important to adopting technology within an organization is the consideration of the knowledge base and knowledge capabilities of the individuals within that organization. Administrators must use the system development life cycle wisely, and carefully consider organizational workflow as they adopt NI technology for meaningful use.

FIGURE III-1 Foundation of Knowledge model.



SOURCE: Designed by Alicia Mastrian.

The reader of this section is challenged to ask the following questions: (1) How can I apply the knowledge gained from my practice setting to benefit my patients and enhance my practice; (2) How can I help my colleagues and patients understand and use the current technology that is available; and (3) How can I use my wisdom to create the theories, tools, and knowledge of the future?

CHAPTER 12

Systems Development Life Cycle: NI and Organizational Decision Making

Dee McGonigle and Kathleen Mastrian

CASE

Envision two large healthcare facilities that merge resources to better serve their community. This merger is called the Wellness Alliance and its mission is to establish and manage community health programming that addresses the health needs of their rural, underserved populations. They would like to establish pilot clinical sites in five rural areas to promote access and provide health care to these underserved consumers. Each clinical site will have a full-time program manager and three part-time employees (a secretary, a nurse, and a doctor). Each program manager will report to the Wellness Program Coordinator, a newly created position within the Wellness Alliance.

Because you are a community health nurse with extensive experience, you have been appointed as the Wellness Program Coordinator. Your directive is to establish these clinical sites within 3 months and report back in 6 months as to the following: (1) community health programs offered, (2) level of community involvement in outreach health programs and clinical site-based programming, (3) consumer visits made to the clinical site, and (4) personnel performance.

You are excited and challenged, but soon reality sets in; you know that you have five different sites with five different program managers. Therefore, there must be some way to gather this information from each of them in a similar manner so that it is meaningful and useful to you as you develop your reports and evaluate the strengths and weaknesses of the pilot project. You know that you need a system that will handle all of the pilot project's information needs.

Your first stop is the **chief information officer** of the health system, a nurse informaticist. You know her from the **health**

Key Terms

WWW

- Chief information officer
- Computer-aided software engineering
- Dynamic system development method
- End-users
- Free/open source software
- Health management information system
- Hospital information system
- Information technology
- Integration
- Interoperability
- Iteration
- Milestones
- MoSCoW
- Object-oriented systems development
- Open source software
- Prototype
- Rapid application development
- Rapid prototyping
- Repository
- Systems development life cycle
- TELOS strategy
- Waterfall model

management information system mini seminar that she led. After explaining your needs, you also share with her that this system must be in place in 3 months when the sites are up and running. When she begins to ask questions, you realize that you do not know the answers. All you know is that you must be able to report on what community health programs were offered, track the level of community involvement in outreach health programs and clinical site-based programming, monitor consumer visits made to the clinical site, and monitor the performance of site personnel. You do know that you want accessible, real-time tracking but as far as programming and clinical site-related activities, you do not have a precise description of the process and procedures that will be involved in implementing the pilot nor how they will gather and enter data.

The chief information officer requires that you and each program manager remain involved in the development process. She assigns an **information technology** (IT) analyst to work with you and your team in the development of a system that will meet your current needs. After the first meeting, your head is spinning because the IT analyst has challenged your team not only to work out the process for your immediate needs but also to envision what your needs will be in the future. At the next meeting, you tell the analyst that your team does not feel comfortable trying to map everything out at this point. He states that there are several ways to go about building the system and software by using the systems' developmental life cycle (SDLC). Noticing the blank look on everyone's faces, he explains, SDLC is a series of actions used to develop an information system. The SDLC is similar to the nursing process where the nurse must assess, diagnose, plan, implement, evaluate, and revise. If this does not meet the patient's need or if a new problem arises, the nurse either revises and updates the plan or starts anew. Therefore, you will plan, analyze, design, implement, operate, support, and secure the system. The SDLC is an iterative process, a conceptual model that is used in project management describing the phases involved in building or developing an information system from assessing feasibility or project initiation, design analysis, system specification, programming, testing, implementation, maintenance, and destruction, literally from beginning to end. Once again, he saw puzzled looks and quickly stated that even the destruction of the system is planned, how it will be retired, broken down, and replaced with a new system. Even during upgrades, destruction tactics can be invoked to secure the data and even decide if servers are to be disposed of or repurposed. The security people will tell you that this is their phase, where they make sure that any sensitive information is properly handled whether the data is to be securely and safely archived or destroyed.

After reviewing all of the possible methods and helping you to conduct your feasibility and business study, the analyst chose the **dynamic system development method** (DSDM). This SDLC model was chosen because it works well when the

time span is short and the requirements are fluctuating and mainly unknown at the outset. He explains that this model works well on tight schedules and is a highly iterative and incremental approach stressing continuous user input and involvement. As a highly iterative process, this means that the team will revisit and loop through the same development activities numerous times; this repetitive examination provides ever-increasing levels of detail, thus improving accuracy. The analyst explains that you will use a mockup of the **hospital information system** (HIS) and design for what is known and then create your own mini system that will interface with the HIS. Because time is short, the analysis, design, and development phases will occur simultaneously while formulating and revising your specific requirements through the iterative process so that they can be integrated into the system.

The functional model **iteration** phase will be completed in 2 weeks based on what you have given to the analyst. He explains further that, at that time, the **prototype** will be reviewed by the team. He tells you to expect at least two or more iterations of the prototype based on your input. You should end with software that provides some key capabilities. Design and testing will occur in the design and build iteration phase until the system is ready for implementation, the final phase. This DSDM should work well because any previous phase can be revisited and reworked through its iterative process.

One month into the SDLC process, the IT analyst tells the team that he will be leaving his position at Wellness Alliance. He introduces his replacement. She is new to Wellness Alliance and is eager to work with the team. The initial IT analyst will be there 1 more week to help the new analyst with the transition. When he explains that you are working through DSDM, she looks a bit panicky and states that she has never used that approach. She says that she has used the waterfall, prototyping, iterative enhancement, spiral, and object oriented methodologies but never the DSDM. From what she heard, DSDM is new and often runs amok because of the lack of understanding as to how to implement it appropriately. After 1 week, she believes that this was not the best choice. As the leader of this SDLC, she is growing concerned about having a product ready for when the clinical sites open. She might combine another method to create a hybrid approach with which she would be more comfortable; she is thinking out loud and has everyone very nervous.

She reviews the equipment that has arrived for the sites and is excited that the Mac computers were ordered from Apple. They will be powerful and versatile enough for your needs. Two months after the opening of the clinical sites, you as the wellness program coordinator are still tweaking the system with the help of the IT analyst. It is hard to believe how quickly the team was able to get a robust system in place. As you think back on the process it seems so long ago that you

reviewed the HIS for deficiencies and screen shots. You re-examined your requirements and watched them come to life through five prototype iterations and constant security updates. You trained your personnel on its use, tested its performance, and made final adjustments before implementation. Your own standalone system that met your needs was installed and fully operational on the Friday before you opened the clinic doors on Monday, 1 day ahead of schedule. You are continuing to evaluate and modify the system, but that is how the SDLC works: it is never finished, but rather constantly evolving.

INTRODUCTION

This case scenario demonstrates the need to have all of the stakeholders involved from the beginning to the end of the SDLC. Creating the right team to manage the development is a key. Various methodologies have been developed to guide the process. This chapter reviews the following approaches to SDLC: waterfall, **rapid prototyping** or **rapid application development** (RAD), **object-oriented system development** (OOSD), and DSDM. When reading about each approach, think about the case scenario and how important it is to understand the specific situational needs and the various methodologies for bringing a system to life. As in this case, it is generally necessary or beneficial to use a hybrid approach that blends two or more models for a robust development process.

As the case demonstrates, the process of developing systems or SDLC is an ongoing development with a life cycle. The first step in developing a system is to understand the problem or business needs; followed by understanding the solution or how to address those needs; developing a plan; implementing the plan; evaluating the implementation; and finally, maintenance, review, and destruction. If the system needs major upgrading outside of the scope of the maintenance phase, needs to be replaced because of technologic advances, or if the business needs change, a new project is launched, the old system is destroyed, and the life cycle begins anew.

SDLC is a way to deliver efficient and effective information systems that fit with the strategic business plan of an organization. The business plan stems from the mission of the organization. In the world of health care, this includes the needs assessment for the entire organization, which should include outreach linkages (as seen in the case scenario) and partnerships and merged or shared functions. The organization's participating physicians and other ancillary professionals and their offices are included in thorough needs assessments. When developing a strategic plan, the design must take into account the existence of the organization within the larger health care delivery system and also assess those factors outside of the organization itself including technologic, legislative, and environmental issues that impact the organization. The plan must identify the needs of the organization as a whole and solutions to meet the needs or a way to address the issues.

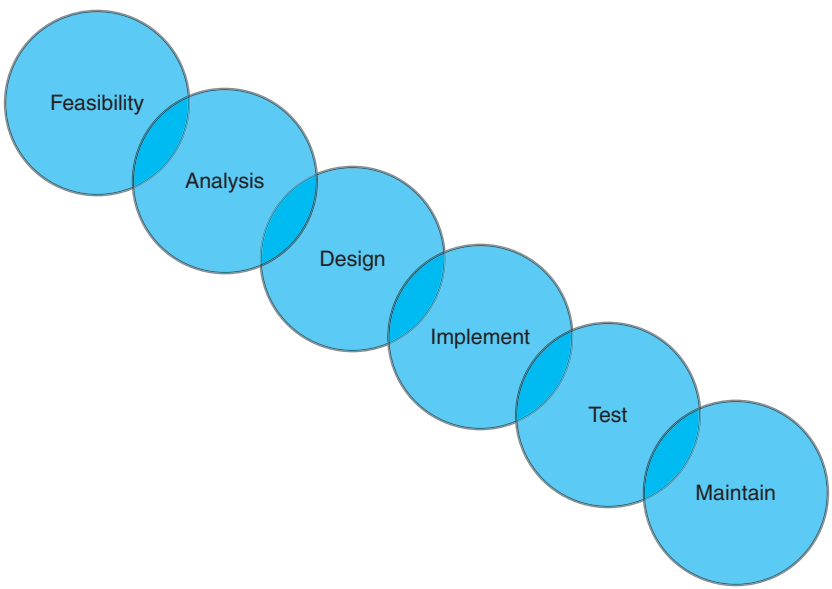
SDLC can occur within an organization, be outsourced, or be a blending of the two. By outsourcing, the team hires an outside organization to carry out all or some of the development. Developing systems that truly meet business needs is not an easy task and is quite complex. Therefore, it is common to run over budget and miss **milestones**. When reading this chapter, reflect on the case scenario and in general the challenges teams face when developing systems.

WATERFALL MODEL

The **waterfall model** is one of the oldest methods and literally depicts a waterfall effect; the output from each previous phase flows into or becomes the initial input for the next phase. This model is a sequential development process because there is one pass through each component activity from conception or feasibility through implementation in a linear order. The deliverables for each phase result from the inputs and any additional information that is gathered. There is minimal or no iterative development where one takes advantage of what was learned during the development of earlier deliverables. Many projects are broken down into six phases (Figure 12-1), especially small- to medium-size projects.

FIGURE 12-1

Waterfall phases.



Feasibility

As the term implies, the feasibility study is used to determine if the project should be initiated and supported. This study should generate a project plan and estimated budget for the SDLC phases. Often, the **TELOS strategy** is followed: *t*echnologic and systems, *e*conomic, *l*egal, *o*perational, and *s*chedule feasibility. Technologic and systems feasibility addresses the issues of technologic capabilities including the expertise and infrastructure to complete the project. Economic feasibility is the cost–benefit analysis, weighing the benefits versus the costs to determine if the project is fiscally possible to do and worth undertaking. Formal assessments should include return on investment. Legal feasibility assesses the legal ramifications of the project including current contractual obligations, legislation, regulatory bodies, and liabilities that could affect the project. Operational feasibility determines how effective the project will be in meeting the needs and expectations of the organization and actually achieving the goals of the project or addressing and solving the business problem. Schedule feasibility assesses the viability of the timeframe, making sure it is a reasonable estimation of the time and resources necessary for the project to be developed in time to attain the benefits and meet constraints. TELOS helps to provide a clear picture of the feasibility of the project.

Analysis

During the analysis phase, the requirements for the system are teased out from a detailed study of the business needs of the organization. This is when work flows and business practices are examined. It may be necessary to consider options for changing the business process.

Design

The design phase focuses on high- and low-level design and interface and data design. At the high-level phase, the team establishes what programs are needed and ascertains how they are going to interact. At the low end phase, they explore how the individual programs are actually going to work. The interface design determines what the look and feel will be or what the interfaces will look like. During data design, the team critically thinks about and verifies what data are required or essential.

The analysis and design phases are vital in the development cycle and great care is taken during these phases to ensure that the software's overall configuration is defined properly. Mockups or prototypes of screen shots, reports, and processes may be generated to clarify the requirements and get the team or stakeholders on the same page, limiting the occurrence of glitches resulting in costly software development resolutions later in the project.

Implement

During this phase, the designs are brought to life through programming code. The right programming language, such as C++, Pascal, Java, and so forth, is chosen based on the application requirements.

Test

The testing is generally broken down into five layers: (1) the individual programming modules, (2) **integration**, (3) volume, (4) the system as a whole, and (5) beta testing. Typically, the programs are developed in a modular fashion and these individual modules are subjected to detailed testing. The separate modules are then synthesized and the interfaces between the modules are tested. The system is evaluated with respect to its platform and expected amount or volume of data. The system is then tested as a complete system by the team. Finally, to determine if the system performs appropriately for the user, it is beta tested. During beta testing, users put the new system through its paces to make sure that it does what they need it to do to perform their jobs.

Maintain

Once the system has been finalized from the testing phase, it must be maintained. This could include user support through actual software changes necessitated through use or time.

The waterfall approach is linear and progresses sequentially. The main lack of iterative development is seen as a major weakness according to Purcell (2007). No projects are static and typically changes occur during the SDLC. As requirements change there is no way formally to address them using the waterfall method after project requirements are developed. The waterfall model should be used for simple projects when the requirements are well-known and stable from the outset.

RAPID PROTOTYPING OR RAPID APPLICATION DEVELOPMENT

As technology advances and faster development is expected, RAD provides a fast way to add functionality through prototyping and user testing. It is easier for users to examine actual prototypes rather than documentation. There is a rapid requirements gathering phase using workshops and focus groups to build a prototype application using real data. This is then beta tested with users and their feedback is used to perfect or add functionality and capabilities to the system. According to Alexandrou (2010), "RAD (rapid application development) proposes that products can be developed faster and of higher quality" (para. 1). The RAD approach uses informal communication, repurposes components, and typically follows a

fast-paced schedule. Object-oriented programming using such languages as C++ and Java promotes software repurposing and reuse.

The major advantage is the speed with which the system can be deployed; a working, usable system can be built within 3 months. The use of prototyping allows the developers to skip steps in the SDLC process in favor of getting a mockup in front of the user. At times, the system is deemed acceptable if it meets a predefined minimum set of requirements rather than all of the identified requirements. This rapid deployment also limits the project's exposure to change elements. The fast pace can also be its biggest disadvantage. Once one is locked into a tight development schedule, the process may be too fast for adequate testing to be in place and completed. The most dangerous lack of testing is in the realm of security.

The RAD approach is chosen because it builds systems quickly through user-driven prototyping and adherence to quick, strict delivery milestones. This approach continues to be refined and honed and other contemporary manifestations of RAD continue to emerge in the agile software development realm.

OBJECT-ORIENTED SYSTEMS DEVELOPMENT

The OOSD model "combines the logic of the **systems development life cycle** with the power of object-oriented modeling and programming" (Stair & Reynolds, 2008, p. 501). Object-oriented modeling makes an effort to represent real world objects; modeling the real world entities or things (e.g., hospital, patient, account, nurse) into abstract computer software objects. Once the system is object oriented, all of the interactions or exchanges take place between or among the objects. The objects are derived from classes and "an object consists of both data and the actions that can be performed on the data" (Stair & Reynolds, p. 501). Class hierarchy allows objects to inherit characteristics or attributes from parent classes and this fosters object reuse resulting in less coding. The object-oriented programming languages, such as C++ and Java, promote software repurposing and reuse. Therefore, the class hierarchy must be clearly and appropriately designed to reap the benefits of this SDLC approach, which uses object-oriented programming to support the interactions of objects.

For example, in the case scenario, a system could be developed for the Wellness Alliance to manage the community health programming for the clinic system being set up for outreach. There could be a class of programs and well-baby care could be an object in the class of programs; programs is a relationship between Wellness Alliance and well-baby care. The program class has attributes, such as clinic site, location address, or attendees or patients. The relationship itself may be considered an object having attributes, such as pediatric programs. The class hi-

erarchy from which all of the system objects are created with resultant object interactions must be clearly defined.

The OOSD model is a highly iterative approach. The process begins by investigating where object-oriented solutions can address business problems or needs, determining user requirements, designing the system, programming or modifying object modeling (class hierarchy and objects), implementing, user testing, modifying, and implementing the system, and ends with the new system being reviewed regularly at established intervals and modifications being made as needed throughout its life.

DYNAMIC SYSTEM DEVELOPMENT METHOD

DSDM is a highly iterative and incremental approach with a high level of user input and involvement. The iterative process requires repetitive examination that enhances detail and improves accuracy. The DSDM has three phases: (1) preproject; (2) project life cycle (feasibility and business studies, functional model iteration, design and build iteration, and implementation); and (3) postproject.

In the preproject phase, buy-in or commitment is established and funding is secured. This helps to identify the stakeholders (administration and **end-users**) and gain support for the project.

In the project life cycle phase, the project's life cycle begins. There are five steps during this phase: (1) feasibility, (2) business studies, (3) functional model iteration, (4) design and build iteration, and (5) implementation.

In steps one and two, the feasibility and business studies are completed. The team ascertains if this project meets the required business needs while identifying the potential risks during the feasibility study. In step one, the deliverables are a feasibility report, project plan, and a risk log. Once the project is deemed feasible step two, the business study, is begun. The business study extends the feasibility report by examining the processes, stakeholders, and their needs. It is important to align the stakeholders with the project and secure their buy-in because it is necessary to have user input and involvement throughout the entire DSDM process. Therefore, bringing them in at the beginning of the project is imperative.

Using the **MoSCoW** approach, the team works with the stakeholders to develop a prioritized requirements list and a development plan. The MoSCoW approach stands for *M*ust have, *S*hould have, *C*ould have, and *W*ould have. The "must have" requirements are needed to meet the business needs and are critical to the success of the project. "Should have" requirements are those that would be great to have if possible, but the success of the project does not depend on them being addressed. The "could have" requirements are those that would be nice to have met, and the "would have" are those requirements that can be put off until

later; these may be undertaken during future developmental iterations. Timeboxing is generally used to develop the project plan. In timeboxing, the project is divided into sections with each having its own fixed budget and dates or milestones for deliverables. The MoSCoW approach is then used to prioritize the requirements within each section; the requirements are the only variables because the schedule and budget are set. If a project is running out of time or money, the team can easily omit the requirements that have been identified as the lowest priority to meet their schedule and budget obligations. This does not mean that the final deliverable, the actual system, would be flawed or incomplete. Instead, it meets the business needs. According to Haughey (2010), the 80/20 rule or Pareto principle can be applied to nearly everything. Using the Pareto principle, 80% of the project comes from 20% of the system requirements; therefore, the 20% of requirements must be the crucial requirements or those with the highest priority. One also must consider the pancake principle: the first pancake is not as good as the rest, and one should know that the first development is not going to be perfect. This is why it is extremely important to clearly identify the "must have" and "should have" requirements.

In the third step, functional model iteration, the deliverables are a functional model and prototype ready for user testing. Once the requirements are identified the next step is to translate them into a functional model with a functioning prototype that can be evaluated by users. This could take several iterations to develop the wanted functionality and incorporate the users' input. At this stage, the team should examine the quality of the product and revise the list requirements and risk log. The requirements are adjusted, the ones that have been realized are deleted, and the remaining requirements are prioritized. The risk log is revised based on the risk analysis completed during and after prototype development.

The design and build iteration step focuses on integrating functional components and identifying the nonfunctional requirements that need to be in the tested system. Testing is crucial; the team will develop a system that the end-users can safely use on a daily basis. The team will garner user feedback and generate user documentation. These efforts provide this step's deliverable, a tested system with documentation for the next and final phase of the development process.

In the final step, implementation, deliverables are the system (ready to use), documentation, and trained users. The requirements list should be satisfied along with the users' needs. Training users and implementing the approved system is the first part of this phase, and the final part consists of a full review. It is important to review the impact of the system on the business processes and determine if it addressed the goals or requirements established at the beginning of the project. This final review determines if the project is completed or if further development is necessary. If further development is needed, preceding phases are revisited. If it

is complete and satisfies the users, then it moves into postproject (maintenance and ongoing development).

The final phase is labeled "postproject." The team verifies that the system is functioning properly. Once verified, the maintenance schedule is begun. Because the DSDM is iterative, this postproject phase is seen as ongoing development and any of the deliverables can be refined. This is what makes the DSDM such an iterative development process.

DSDM is one of an increasing number of agile methodologies, such as Scrum and Extreme Programming. These new approaches address the organizational, managerial, and interpersonal communication issues that bog down SDLC projects. Empowering teams and user involvement enhance the iterative and programming strengths provided in these SDLC models.

COMPUTER-AIDED SOFTWARE ENGINEERING TOOLS

When reviewing SDLC, the **computer-aided software engineering** (CASE) tools must be described. The "CASE tools automate many of the tasks required in a systems development effort and encourage adherence to the SDLC, thus instilling a high degree of rigor and standardization to the entire systems development process" (Stair & Reynolds, 2008, p. 500). These tools help to reduce cost and development time while enriching the quality of the product. The CASE tools contain a **repository** with information about the system: models, data definitions, and references linking models together. They are valuable in their ability to make sure the models follow diagramming rules and are consistent and complete. They can be referred to as upper CASE tools or lower CASE tools. The upper CASE tools support the analysis and design phases, whereas the lower CASE tools support implementation. The tools can also be general or specific in nature with the specific tools being designed for a particular methodology. Two examples of CASE tools are Visible Analyst and Rational Rose. According to Andoh-Baidoo, Kunene, and Walker (n.d.), Visible Analyst "supports structured and object-oriented design (UML)," whereas Rational Rose "supports solely object-oriented design (UML)" (p. 372). They can both "build and reverse database schemas for SQL and Oracle" and "support code generation for pre.NET versions of Visual Basic" (p. 372). Visible Analyst can also support shell code generation for pre.NET versions of C and COBOL, whereas Rational Rose can support complete code for C++ and Java. In addition, Andoh-Baidoo et al. found that Rational Rose "Provides good integration with Java, and incorporates common packages into class diagrams and decompositions through classes" (p. 372). The CASE tools have many advantages including decreasing development time and producing more flexible systems. On the down side, they can be difficult to tailor or customize and use with existing systems.

OPEN SOURCE SOFTWARE AND FREE/OPEN SOURCE SOFTWARE

Another area that must be discussed with SDLC is **open source software** (OSS). An examination of job descriptions or advertisements for candidates shows that many IS and IT professionals need a thorough understanding of SDLC and OSS development tools (e.g., PHP, MySQL, and HTML). This is because, with OSS, any programmer can implement, modify, apply, reconstruct, and restructure the rich libraries of source codes available from proven, well-tested products. Vaulde (2008) offers "Examples of FOSS successes are the Internet, Google, Web 2.0, the GNU/Linux operating system, courseware such as Moodle, and the Veterans Affairs VistA hospital system" (p. 7).

To transform health care it is necessary for clinicians to use information systems that can share patient data (Goulde & Brown, 2006). This all sounds terrific and many people wonder why it has not as yet happened, but the challenges are many. How does one establish the networks necessary to share data between and among all healthcare facilities easily and securely? "Healthcare IT is beginning to adopt open source software to address these challenges" (Goulde & Brown, p. 4). Early attempts at OSS ventures in the healthcare realm failed because of a lack of support or buy-in for sustained effort, technologic lags, authority and credibility, and other such issues. "Spurred by a greater sense of urgency to adopt IT, health industry leaders are showing renewed interest in open source solutions" (Goulde & Brown, p. 5). Health care is realizing the benefits of OSS. Goulde and Brown stated that "other benefits of open source software—low cost, flexibility, opportunities to innovate—are important but independence from vendors is the most relevant for health care" (p. 10).

INTEROPERABILITY

Interoperability, the ability to share information across organizations, will remain paramount under the HITECH Act (see Chapter 10). The ability to share patient data is extremely important, both within an organization and across organizational boundaries. According to HIMSS (2010), few healthcare systems take advantage of the full potential of the current state of the art in computer science and health informatics. The consequences of this situation include a drain on financial resources from the economy, the inability to truly mitigate the occurrence of medical errors, and a lack of national preparedness to respond to natural and man-made epidemics and disasters. HIMSS has created the Integration and Interoperability Steering Committee to guide the industry on allocating resources to develop and implement standards and technology needed to achieve interoperability (para. 2).

As we enter into SDLCs, we must be aware of the impact on our own healthcare organization and the impact on the healthcare delivery system as a whole. In an ideal world, we would all work together to create systems that are integrated within our own organization while having the interoperability to cross organizational boundaries and unite the healthcare delivery system to realize the common goal of improving the quality of care provided to consumers.

CONCLUSION

At times during the SDLC, new information affects the outputs from earlier phases and the development effort may be re-examined or halted until these modifications can be reconciled with the current design and scope of the project. There are times that teams are overwhelmed with new ideas from the iterative SDLC process that result in new capabilities or features that exceed the initial scope of the project. Astute team leaders preserve these ideas or initiatives so they can be considered at a later time. The team should develop a list of recommendations to improve the current software when the project is complete. This iterative and dynamic exchange makes the SDLC robust.

As technology and research continue to advance, new SDLC models are pioneered and introduced to enhance techniques. The interpretation and implementation of any model selected reflects the knowledge and skill of the team applying the model. The success of the project is often directly related to the quality of the organizational decision making throughout the project; how well the plan was followed and documented. United efforts to create systems that are integrated and interoperable will define the future of health care.

THOUGHT-PROVOKING Questions

1. How would you describe cognitive informatics? Reflect on a plan of care that you have developed for a patient. How could cognitive informatics be used to create tools to help with this important work?
2. Think of a clinical setting you are familiar with and envision artificial intelligence tools. Are there any current tools in use? What tools would enhance practice in this setting and why?
3. Reflect on the SDLC in relation to the quality of the organizational decision making throughout the project. What are some of the major stumbling blocks faced by healthcare organizations?

For a full suite of assignments and additional learning activities, use the access code located in the front of your book to visit this exclusive website: **http://go.jblearning.com/mcgonigle.** If you do not have an access code, you can obtain one at the site.

References

Alexandrou, M. (2010). *Rapid application development (RAD) methodology.* Retrieved from http://www.mariosalexandrou.com/methodologies/rapid-application-development.asp

Andoh-Baidoo, F., Kunene, K., & Walker, R. (n.d.). *An evaluation of CASE tools as pedagogical aids in software development courses.* Retrieved from http://www.swdsi.org/swdsi 2009/Papers/9K10.pdf

Brown, P. (2010). *Free software is a matter of liberty, not price.* Retrieved from http://www.fsf.org/about/

Goulde, M., & Brown, E. (2006). *Open source software: A primer for health care leaders.* Retrieved from http://www.chcf.org/~/media/Files/PDF/O/OpenSourcePrimer.pdf

Haughey, D. (2010). *Pareto analysis step by step.* Retrieved from http://www.projectsmart.co .uk/pareto-analysis-step-by-step.html

HIMSS. (2010). *Integration and interoperability.* Retrieved from http://www.himss.org/ASP/topics_integration.asp

Lee, M. (2009). *FSF announces new executive director.* Retrieved from http://www.fsf.org/news/new-executive-director.html

Purcell, J. (2007). *Comparison of software development lifecycle methodologies.* Retrieved from http://www2.giac.org/resources/whitepaper/application/217.pdf

Stair, R., & Reynolds, G. (2008). *Principles of information systems* (8th ed.). Boston, MA: Thomson Course Technology.

Vauldes, I. (2008). *Free and open source software in healthcare 1.0: American Medical Informatics Association. Open Source Working Group White Paper.* Retrieved from Https://Www.Amia.Org/Files/Final-Os-Wg%20white%20paper_11_19_08.Pdf