Objectives

1. Describe the systems development life cycle (SDLC).
2. Explore selected approaches to the SDLC.
3. Assess interoperability and its importance in addressing and meeting the challenges of implementing the HITECH Act in health care.
4. Reflect on the past to move forward into the future to determine how new systems will be developed, integrated, and made interoperable in health care.

Key Terms

- agile
- chief information officer
- computer-aided software engineering (CASE)
- dynamic system development method (DSDM)
- end user
- free/libre open source software (FLOSS)
- free/open source software (F/OSS or FOSS)
- health management information system
- hospital information system (HIS)
- information technology (IT)
- integration
- interoperability
- iteration
- milestone
- MoSCoW
- object-oriented systems development (OOSD)
- open source software (OSS)
- prototype
- rapid application development (RAD)
- rapid prototyping
- repository
- Scrum
- Scrum master
- Scrum team
- systems development life cycle (SDLC)
- TELOS strategy
- total cost of ownership (TCO)
- waterfall model

CHAPTER 9

Systems Development Life Cycle: Nursing Informatics and Organizational Decision-Making

Introduction

The following case scenario describes the development of rural clinics with an integrated data collection and processing system demonstrating the need to have all stakeholders involved from the beginning to the end of a **systems development life cycle (SDLC)**. Creating the right team to manage system development is key. Various methodologies have been developed to guide this process. This chapter reviews the following approaches to the SDLC: waterfall; rapid prototyping, or rapid development (RAD); object-oriented system development (OOSD); dynamic system development method (DSDM); and agile. 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 the case scenario described later, 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 the SDLC, is an ongoing development with a life cycle. The first step in developing a system is to understand the problems, or business needs. This step is followed by understanding the solution, or how to address those needs; developing a plan; implementing the plan; evaluating the implementation; and, finally, performing plan maintenance, plan review, and system destruction. If a system needs major upgrading outside the scope of the maintenance phase or needs to be replaced because of technological advances or the business needs change, a new project is launched, the old system is destroyed, and the life cycle begins anew.

The SDLC is a way to deliver efficient and effective information systems (ISs) that fit with the strategic business plan of the organization, which stems from its mission. In the world of health care, the development of the IS includes a needs assessment of 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 a thorough needs assessment. When developing a strategic plan, the design must consider the existence of the organization within the larger healthcare delivery system and assess the various factors outside of the organization itself, including technological, legislative, and environmental issues, that affect the organization. The plan must identify the needs of the organization as a whole and propose solutions to meet those needs or a way to address the issues.

The SDLC can occur within an organization or be outsourced or be a blend of the two approaches. With 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.

CASE SCENARIO

Wellness Alliance was formed from the merger of two large healthcare facilities to better serve the community. Its mission is to establish and manage community health programming that addresses the health needs of the rural, underserved populations in the area. Wellness Alliance 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 (i.e., a secretary, a nurse, and a doctor). Each program manager will report to the wellness program coordinator, a newly created position within 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 with the following information: (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 sites with five program managers. You need some way to gather vital information from each of them in a similar manner so that the data are 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 management information system** mini-seminar that she led. After explaining your needs, you share with her the constraint that this system must be in place in 3 months, which is when the sites will be up and running, before you make your report. 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 which 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 know that you want accessible, real-time tracking, but as far as programming and clinical site–related activities are concerned, you do not have a precise description of either the process or the procedures that will be involved in implementing the pilot or the means by which the clinical sites will gather and enter data.

The chief information officer requires that you and each program manager remain involved in the development process. They assign 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. The IT analyst has challenged your team not only to work out the process for your immediate needs but also to envision what your future needs will be. At the next meeting, you tell the analyst that your team does not feel comfortable trying to map everything out at this point.

The analyst states that there are several ways to go about building the system and software by using the SDLC. Noticing the blank look on everyone's faces, they explain that the SDLC is a series of actions used to develop an IS. The SDLC is similar to the nursing process in which the nurse must assess, diagnose, plan, implement, evaluate, and revise. If the plan developed in this way does not meet the patient's need or a new problem arises, the nurse either revises and updates the plan or starts anew. Likewise, you will plan, analyze, design, implement, operate, support, and secure the proposed community health system. The SDLC is an iterative process—a conceptual model that is used in project management to describe the phases involved in developing an IS. It moves from assessing feasibility to project initiation, to design analysis, to system specification, to programming, to testing, to implementation, to maintenance, and to destruction—literally from beginning to end.

As the IT analyst describes this process, once again they see puzzled looks. They quickly state that even the destruction of the system is planned—that is, 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 whether 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 and decide whether the data are to be securely and safely archived or destroyed.

After reviewing all the possible methods and helping you to conduct your feasibility and business study, the analyst chooses the DSDM. This SDLC model was selected because it works well when the time span is short and the requirements are fluctuating and mainly unknown at the outset. The IT analyst explains that this model works well on tight schedules and is a highly iterative and incremental approach that stresses continual user input and involvement. As part of this highly iterative process, the team will revisit and loop through the same development activities numerous times; this repetitive examination provides ever-increasing levels of detail, thereby improving accuracy. The analyst explains that you will use a mock-up of the **hospital information system (HIS)** and design for what is known; you will 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 you are 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 the information that you have given the analyst. At that time, the team will review the **prototype**. The IT analyst 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 and continue until the system is ready for implementation, which is 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 they will be leaving their position at Wellness Alliance. They introduce their replacement. The replacement is new to Wellness Alliance and is eager to work with the team. The original IT analyst will be there for one more week to help the new analyst with the transition. When the original analyst explains to the new analyst that you are working through DSDM, they look a bit panicky and state that they have never used this approach. They have used the waterfall, prototyping, iterative enhancement, spiral, and object-oriented methodologies but never the DSDM. From what they have heard, DSDM is new and often runs amok because of the lack of understanding as to how to implement it appropriately. After one week on the project, the new IT analyst believes that this approach was not the best choice. As the leader of this SDLC, they are growing concerned about having a product ready when the clinical sites open.

They might combine another method to create a hybrid approach with which they would be more comfortable; they are thinking aloud and have everyone very nervous.

The IT analyst reviews the equipment that has arrived for the sites and is excited to learn that the 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 screenshots. You reexamined 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 stand-alone system that met your needs was installed and fully operational on the Friday before you opened the clinic doors on Monday, one day ahead of schedule.

At the end of week one, the IT analyst sends out a meeting invitation to discuss the project and lessons learned. In the invitation, they state that the purpose of the meeting is to review the process, what went well and what did not, and any issues that arose and how they were addressed so that this process can be improved going forward. They stress that this is a meeting to review the process and a time to reflect, not a time to cast blame but instead to look for areas of growth; it can be a time to recognize and praise those who went above and beyond.

The lessons-learned meeting was highly collaborative and energizing since everyone came to the meeting prepared to share their reflections and perspectives openly. Communication was one of the areas where issues arose during development. They discussed how this could have been handled better at the time and offered suggestions for improving communications in subsequent developments. Many of the people in attendance were recognized for their work ethic and positive attitudes throughout the entire project. It was a very positive meeting that did address issues and ideas for improving the process going forward.

You are continuing to evaluate and modify the system, but that is how the SDLC works. It is never finished but rather is constantly evolving.

Waterfall Model

The **waterfall model** is one of the oldest methods and literally depicts a waterfall effect; that is, the output from each previous phase flows into or becomes the initial input for the next phase. This model is a sequential development process in that 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 9-1**), especially small to medium-size projects.

Feasibility

As the term implies, the feasibility study is used to determine whether the project should be initiated and supported. This study should generate a project plan and an estimated budget for the SDLC phases. Often, the **TELOS strategy** (i.e., technological and systems, economic, legal, operational, and schedule feasibility) is followed. Technological and systems feasibility addresses the issues of technological capabilities,

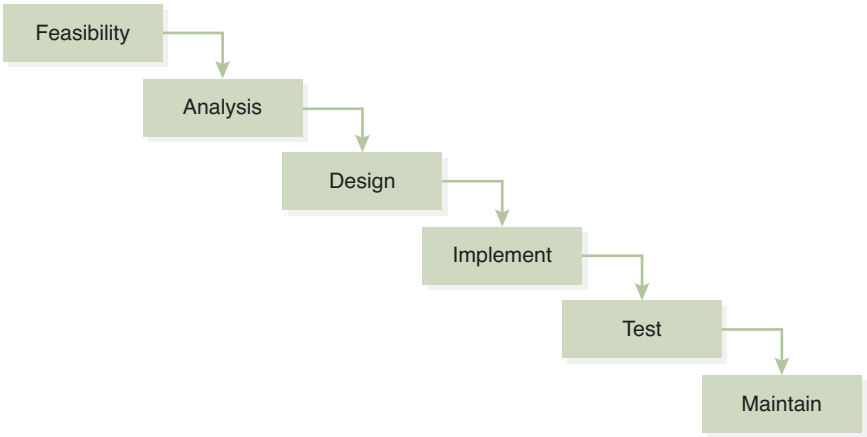

Figure 9-1 Waterfall Phases

including the expertise and infrastructure needed to complete the project. Economic feasibility is determined by performing a cost-benefit analysis, which weighs the benefits versus the costs to determine whether the project is fiscally possible and worth undertaking. Formal assessments should include a 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 time frame, 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. The TELOS strategy 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. As part of this analysis, workflows 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, team members establish which programs are needed and ascertain how they will interact. At the low-level phase, team members explore how the individual programs will actually 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 which 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. Mock-ups or prototypes of screenshots, 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 that might result in costly software development revisions 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, and Java, 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 the individual modules are then subjected to detailed testing. The modules are subsequently synthesized, and the interfaces between the modules are tested. The system is evaluated regarding its platform and the expected volume of data. It is then tested as a complete system by the team. Finally, to determine whether 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.

This model is sequential, with five common stages that must be completed before the next sequential stage begins (Adobe Communications Team, 2022; Isaias & Issa, 2015). This could be considered a disadvantage to system development since no projects are static and changes typically occur during the SDLC. As requirements change, there is no way to address them formally using the waterfall model 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, **rapid prototyping**, also known as **rapid application development (RAD)**, provides a fast way to add functionality through prototyping and user testing. It is easier for users to examine an actual prototype than documentation. A rapid requirements-gathering phase relies on workshops and focus groups to build a prototype application using real data. This prototype is then beta tested with users, and their feedback is used to perfect or add functionality and capabilities to the system (**Figure 9-2**).

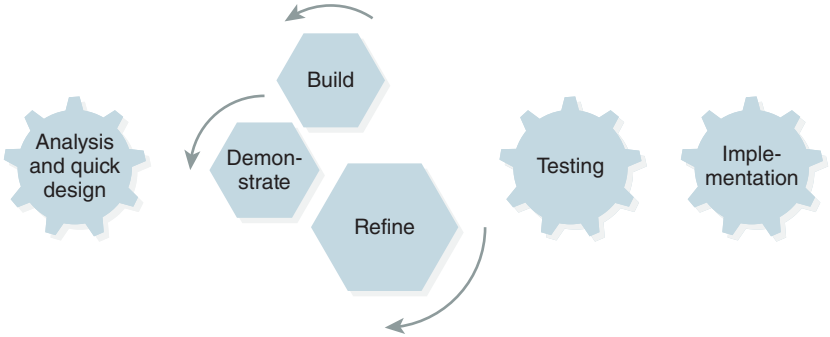

Figure 9-2 Rapid Prototyping, or Rapid Application Development (RAD)

According to Alexandrou (2023), "RAD 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 languages such as C++ and Java, promotes software repurposing and reuse.

The major advantage of RAD 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 mock-up in front of the user. At times, the system may be 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. Unfortunately, the fast pace can be its biggest disadvantage in some cases. Once one is locked into a tight development schedule, the process may be too fast for adequate testing to be put 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 **object-oriented systems development (OOSD)** model blends SDLC logic with object-oriented modeling and programming power to apply object-oriented concepts to all the stages of the SDLC (GeeksforGeeks, 2022; Stair & Reynolds, 2016). It is a method of contemplating and reflecting on problems using models organized around real-world concepts or objects. Object-oriented modeling represents real-world objects by modeling the real-world entities or things (e.g., clinic, patient, account, or nursing or healthcare professional) into abstract computer software objects. Once the system is object oriented, all the interactions or exchanges take place between or among the objects. The objects are derived from classes, and each object comprises data and the actions that can be enacted on that data. Class hierarchy allows objects

to inherit attributes from parent classes, which fosters object reuse and results 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 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 hierarchy from which all 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 (i.e., class hierarchy and objects); and implementing, testing, modifying, and reimplementing 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

The **dynamic system development method (DSDM)** is a very 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 (i.e., feasibility and business studies, functional model iteration, design and build iteration, and implementation), and (3) postproject.

In the preproject phase, buy-in is established and funding is secured. This helps to identify the stakeholders (i.e., administration and **end users**) and gain support for the project. In the second phase, the project's life cycle begins. This phase includes five steps: (1) feasibility, (2) business studies, (3) functional model iteration, (4) design and build iteration, and (5) implementation (**Figure 9-3**).

In steps 1 and 2, the feasibility and business studies are completed. The team ascertains whether this project meets the required business needs while identifying the potential risks during the feasibility study. In step 1, the deliverables are a feasibility report, project plan, and risk log. Once the project is deemed feasible, step 2, the business study, is begun. The business study extends the feasibility report by examining the processes and the stakeholders' 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. **MoSCoW** stands for "must have, should have, could have, and would have." The "must have" requirements are

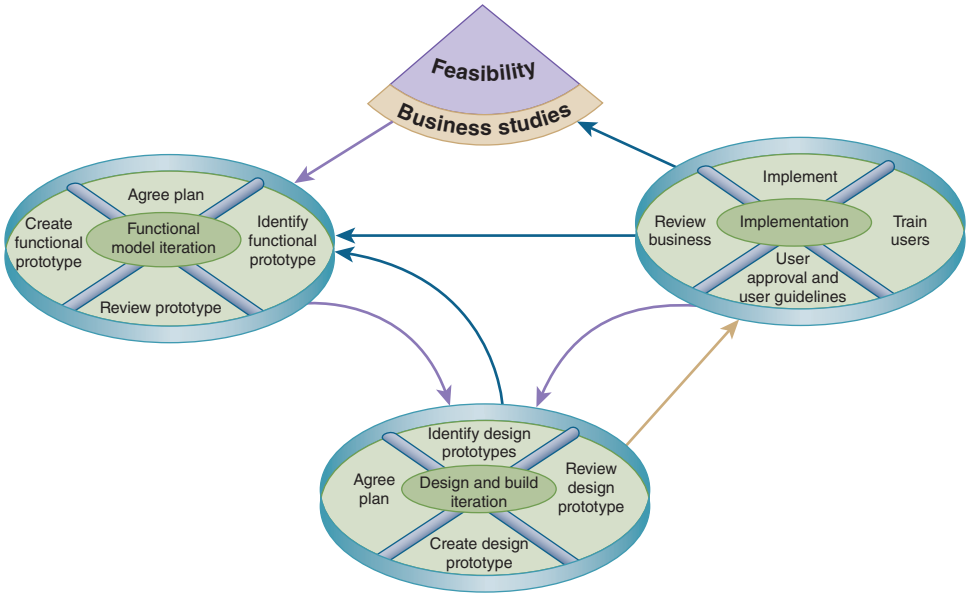

Figure 9-3 Dynamic System Development Method (DSDM)

necessary to meet the business needs and are critical to the success of the project. The "should have" requirements are those that would be great to have if possible, but the success of the project does not depend on their being addressed. The "could have" requirements are those that would be nice to have met, and the "would have" requirements can be put off until later; these requirements may be undertaken during future developmental iterations. Timeboxing is generally used to develop the project plan. In timeboxing, the project is divided into sections, each having its own fixed budget and 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, because the team has already determined the "must have" or "should have" items, it still meets the business needs. According to Haughey (2021), the 80/20 rule, or Pareto principle, can be applied to nearly everything. The Pareto principle states that 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 must also consider the pancake principle: The first pancake is not as good as the rest, and one should know that the first development will not be perfect. This is why it is extremely important to clearly identify the "must have" and "should have" requirements.

In the third step of the project life cycle phase, known as functional model iteration, the deliverables are a functional model and prototype ready for user testing. This could take several iterations to develop the desired 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 of the project life cycle phase, known as 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 are the first part of this phase, and the final part consists of a full review. It is important to review the effect of the system on the business processes and determine whether it addresses the requirements established at the beginning of the project. This final review determines whether the project is completed or if further development is necessary. If further development is needed, the preceding phases are revisited. If the project is complete and satisfies the users, then it moves into maintenance and ongoing development.

The final phase is labeled postproject. In this phase, the team verifies that the system is functioning properly. Once verified, the maintenance schedule begins. 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 being introduced, such as Scrum and Extreme Programming. Refer to the following information on the agile method. These new approaches address the organizational, managerial, and interpersonal communication issues that often bog down SDLC projects. Empowerment of teams and user involvement enhance the iterative and programming strengths provided in these SDLC models.

Agile

The **agile** method is different from other approaches to software development since the focus is on "the people doing the work and how they work together. Solutions evolve through collaboration between self-organizing cross-functional teams utilizing the appropriate practices for their context" (Agile Alliance, n.d., para. 7). The agile software development approach merges iterative and incremental process models whereby the product is broken into small incremental builds on which the teams work in short bursts, such as 2- to 4-week intervals, to accomplish their tasks. Failure is considered positive if you fail fast and fail often to get to your goal. This method

lends itself well to smaller projects and rapidly delivers functional software to the customer; this quick turnaround enhances customer satisfaction.

Scrum is an agile strategy that emphasizes collaboration, team autonomy, or self-management and flexibility to adapt to emerging business realities. This strategy organizes software developers into a team called a **Scrum team** to reach a common goal of creating market-ready products and services using iterative development cycles. As a framework, it can manage complex knowledge work. Scrum defines roles, procedures, and tools to guarantee the delivery of an effective and efficient product. The three foundational concepts of Scrum are adaptation, transparency, and inspection. The **Scrum master** is the agile servant–leader project manager who facilitates the work performed by the Scrum team; this leader is responsible for removing barriers and obstacles so that tasks may be completed for the team to achieve its goals. The Scrum master must support the Scrum team and facilitate its ability to self-manage and self-organize. Since this is a highly collaborative process, the Scrum master must also establish and ensure a positive environment that is open to discussion and sharing of ideas. In addition, the Scrum master must assess and monitor Scrum team interactions with those outside of the team to determine which exchanges with the Scrum team are beneficial and which are not in order to maximize the value generated by the Scrum team (i.e., shielding the Scrum team from distractions and work interruptions). The Scrum master guides, protects, coaches, facilitates, and develops the Scrum team to reach its goals of producing high-quality products and service deliverables.

Computer-Aided Software Engineering Tools

When reviewing the SDLC, the **computer-aided software engineering (CASE)** tools that will be used must be described.

CASE is the methodical application of a set of tools that promote adherence to the SDLC process by automating several required tasks, which provides standardization and thoroughness to the total systems to develop quality software systems that are free of defects (Stair & Reynolds, 2016; Waghmare, 2023; Wikibooks.org, 2018). These tools help to reduce cost and development time while enriching the quality of the product. CASE tools contain a **repository** with information about the system: models, data definitions, and references that link models together. Repositories are valuable in their ability to make sure the models follow diagramming rules and are consistent and complete.

The various types of tools can be referred to as upper-CASE 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 (IBM, 2021). According to Andoh-Baidoo et al. (2009), Visible Analyst "supports structured and object-oriented design (UML)" (p. 372). Rational Rose supports solely object-oriented design (UML), allowing for "side-by-side comparison between the Rose diagrams and the equivalent diagrams in the new Rational UML modeling products" (IBM, 2021, para. 3). Both tools can "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. (2009) found that Rational Rose "[p]rovides good integration with Java, and incorporates common packages into class diagrams and decompositions through classes" (p. 372).

CASE tools have many advantages, including decreasing development time and producing more flexible systems. On the downside, they can be difficult to customize and use with existing systems.

Open Source Software, Free/Libre Open Source Software, and Free/Open Source Software

Another area that must be discussed with the SDLC is **open source software (OSS)**, **free/libre open source software (FLOSS)**, and **free/open source software (F/OSS or FOSS)**. There are two political camps in the free software community: the free software movement and open source. According to Stallman (2021),

> *The free software movement is a campaign for computer users' freedom; we say that a nonfree program is an injustice to its users. The open source camp declines to see the issue as a matter of justice to the users, and bases its arguments on practical benefits only.* (para. 1)

Free software refers to freedom and not cost and, as a nonpolitical statement, is noted as free/libre, creating the FLOSS abbreviation. Stallworth stated that "[o]thers use the term 'FOSS,' which stands for 'Free and Open Source Software.' This is meant to mean the same thing as 'FLOSS,' but it is less clear, since it fails to explain that 'free' refers to *freedom*" (para. 4).

An examination of job descriptions or advertisements for candidates shows that many IS and IT professionals need a thorough understanding of the SDLC and OSS development tools (e.g., PHP, MySQL, and HTML). With OSS, any programmer can implement, modify, apply, reconstruct, and restructure the rich libraries of source codes available from proven, well-tested products.

As Karopka et al. (2014) noted,

> *Free/Libre Open Source Software (FLOSS) has been successfully adopted across a wide range of different areas and has opened new ways of value creation. Today there are hundreds of examples of successful FLOSS projects and products. . . . Especially in times of financial crisis and austerity the adoption of FLOSS principles opens interesting alternatives and options to tremendously lower* **total cost of ownership (TCO)** *and open the way for a continuous user-driven improvement process.* (para. 6)

To transform health care, it is necessary for clinicians to use ISs that can share patient data (Goulde & Brown, 2006; HealthIT.gov, 2022; NORC, 2014). Because this all sounds terrific, many people wonder why it has been slowly evolving, not realizing that the challenges are many. How does one establish the networks necessary to share data between and among all healthcare facilities easily and securely? Early

attempts at OSS ventures in the healthcare realm failed because of a lack of support for sustained effort, technological lags, lack of authority and credibility, and other such issues. Health care in the United States has been applying FLOSS-HC (health care). OSS is developed to work with everything and facilitates the sharing of data and information, making it essential to interoperability. With OSS, numerous people and entities are working to solve the same or similar problems, affording healthcare organizations access to innovative solutions to their major challenges such as scalability and security.

Health care is realizing the benefits of FLOSS, including vendor independence.

Interoperability

Interoperability, which is the ability to share information across organizations, remains paramount under the HITECH Act. The ability to share patient data is extremely important, both within an organization and across organizational boundaries (**Figure 9-4**).

According to the Health Information and Management Systems Society (HIMSS; n.d.), there are standards, implementation specifications, and standards-development organizations. These various types of organizations are working to support the development of systems that can seamlessly exchange and use health data. Few healthcare systems take advantage of the full potential of the current state of the art in

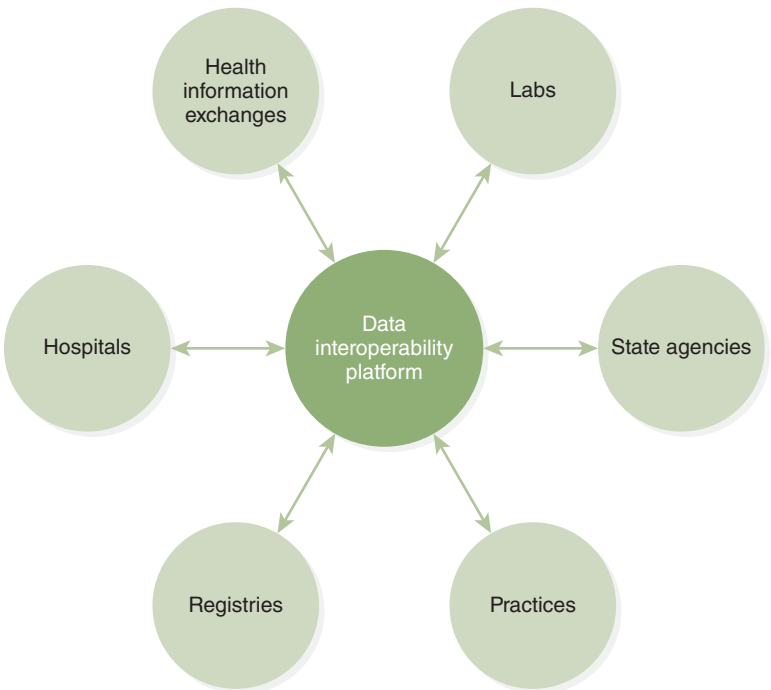

Figure 9-4 Interoperability

computer science and health informatics (HIMSS, n.d.). 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 created the Integration and Interoperability Steering Committee to guide the industry on allocating resources to develop and implement the standards and technology needed to achieve interoperability.

As we enter into the SDLC, we must be aware of how this type of development will affect both our own healthcare organization and 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.

Summary

At times during the SDLC, new information affects the outputs from earlier phases; the development effort may be reexamined or halted until these modifications can be reconciled with the current design and scope of the project. At other times, 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 will preserve these ideas so that 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 being pioneered and revised to enhance development techniques. The interpretation and implementation of any model selected reflect 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—that is, 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. 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?
2. Why is it important for all nurses and healthcare professionals to understand the basics of how ISs are selected and implemented?
3. Describe the major differences between the agile method and other SDLC approaches. Reflect on why you would select the agile method over the waterfall model, and provide a detailed rationale for your choice.

References

Adobe Communications Team. (2022). *Waterfall methodology: A complete guide*. https://business.adobe.com/blog/basics/waterfall

Agile Alliance. (n.d.). *Agile 101*. www.agilealliance.org/agile101

Alexandrou, M. (2023, May 4). Rapid application development (RAD) methodology. *Infolific*. www.infolific.com/technology/methodologies/rapid-application-development

Andoh-Baidoo, F., Kunene, K., & Walker, R. (2009). *An evaluation of CASE tools as pedagogical aids in software development courses*. Southwest Decision Sciences Institute. www.swdsi.org/swdsi2009/Papers/9K10.pdf

GeeksforGeeks. (2022, July 8). *Types of models in object oriented modeling and design*. www.geeksforgeeks.org/types-of-models-in-object-oriented-modeling-and-design

Goulde, M., & Brown, E. (2006, March). *Open source software: A primer for health care leaders*. www.chcf.org/wp-content/uploads/2017/12/PDF-OpenSourcePrimer.pdf

Haughey, D. (2021, October 18). *Pareto analysis step by step*. ProjectSmart. www.projectsmart.co.uk/pareto-analysis-step-by-step.php

Health Information and Management Systems Society. (n.d.). *Interoperability in healthcare*. www.himss.org/resources/interoperability-healthcare

HealthIT.gov. (2022, July 25). *The ONC healthIT playbook: Section 3: Health information exchange*. www.healthit.gov/playbook/health-information-exchange

IBM. (2021, March 2). *Rational Rose model*. www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=migration-rational-rose-model

Isaias, P., & Issa, T. (2015). *High level models and methodologies for information systems*. Springer.

Karopka, T., Schmuhl, H., & Demski, H. (2014). Free/libre open source software in health care: A review. *Healthcare Informatics Research*, *20*(1), 11–22. www.ncbi.nlm.nih.gov/pmc/articles/PMC3950260

NORC. (2014). *Data sharing to enable clinical transformation at the community level: IT takes a village*. Office of the National Coordinator for Health Information Technology. www.healthit.gov/sites/default/files/briefs/beacondatasharingbrief062014.pdf

Stair, R., & Reynolds, G. (2016). *Principles of information systems* (12th ed.). Cengage Learning.

Stallman, R. (2021). *FLOSS and FOSS*. GNU Operating System. www.gnu.org/philosophy/floss-and-foss.en.html

Waghmare, P. (2023, May 17). *Software engineering—CASE tools: Learn about types, components & more!* Testbook. https://testbook.com/software-engineering/software-case-tools

Wikibooks.org. (2018, September 27). *Introduction to software engineering/tools/modelling and case tools*. https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Tools/Modelling_and_Case_Tools