Chapter 1
Introduction to Information Systems Models and Methodologies

1.1 Introduction

As information technology (IT) permeates more and more aspects of human life, information systems (IS) have grown to become an essential component of organizational management. Iivari and Hirschheim (1996) define an information system as a system providing users with information on specified topics within an organizational context, with computers as its main support. Alter (2008), on the other hand, defines an information system as a work system whose activities are centered on the processing of information. IS ultimately provide the support for an organization's networks of information creation, gathering, processing, or storing.

Today, a solid IS, one that is generally accepted by its users and proves to be successful, can determine the success of a business, in a world where competition is ever fiercer. Accordingly with this phenomenon, researchers have grown more and more interested in establishing IS development methodologies and models that can be used across a wide range of contexts, with the purpose of finding ordered, systemic frameworks among the immense variety of techniques and methods that can be found in practice. On the other hand, as IS become more complex, there is a growing need for organizations to have a basis of logical constructs that can provide them with the tools to easily define, control, and integrate all the components of the system (Zachman 1987).

The great variety of existing models for IS development is rooted in the fact that developers of a system will be guided by a number of influences related not only with the object of their work (the goal of the system they are developing), but also with the very nature of their organization, and how it affects expectations. As the developer absorbs these influences, so does the system being built (Hirschheim and Klein 1989), leading to a large number of possible variables, which in turn complicates bringing out a unified view of the problem.

Research has not only focused on the creation and development of IS, but also on what happens to the system beyond its implementation stage, particularly

P. Isaias and T. Issa, *High Level Models and Methodologies for Information Systems*,
DOI 10.1007/978-1-4614-9254-2_1

regarding its acceptance or not within the context of the organization and the user base. The interest in defining what can "make or break" a new system has also lead researchers to focus on building models that can help an organization or project manager determine and measure the system's success.

In this book, we will discuss the major methodologies that have been established in existing literature related to systems development and acceptance, as well as the more prominent models that are rooted in each methodological approach. This will allow us to identify how specific methodologies and models are fit for specific types of IS development projects, underlining the usefulness of such theoretical frameworks for practitioners that want to identify which methods are best for their specific projects.

This book is organized into the following chapters:

Chapter 1—Introduction to IS Models and Methodologies (the current chapter);
Chapter 2—IS Development Life Cycle Models;
Chapter 3—IS Development Methodologies;
Chapter 4—Web Site Development Methodologies;
Chapter 5—Usability Evaluation Models;
Chapter 6—Quality Evaluation Models;
Chapter 7—IS Models for Success Assessment.

Each of the chapters, from 2 to 7, will be briefly introduced in the next pages of this Chap. 1 and detailed in the remaining book.

1.2 Systems Development Paradigms

The vast body of research that relates to IS development has led some researchers to attempt to group different methods into a set of simple categories, based on common principles and similarities. These categories, or paradigms, are essentially formed by the underlying philosophies, goals, guiding principles, and fundamental concepts that justify the choice of a given approach to IS development (Iivari et al. 1998).

According to the seminal work of Hirschheim and Klein (1989), there are four paradigms of IS development, which, in turn, are based on paradigms of systems analysis (see Fig. 1.1).

The *functionalist paradigm* focuses on the context, social order, consensus, needs, and rational choices. IS are developed by application of formal concepts, through methodical and planned intervention and based on rational principles. The *social relativist paradigm* focuses on individual subjectivity and the personal frame of reference of the social actor. IS development takes into account the subjective and cultural context of the developer. The *radical structuralist paradigm* advocates the need to transcend existing limitations born out of social and organizational structures. IS development is built by an awareness of necessities and limits and what can be done to improve the system beyond that border. Finally, the

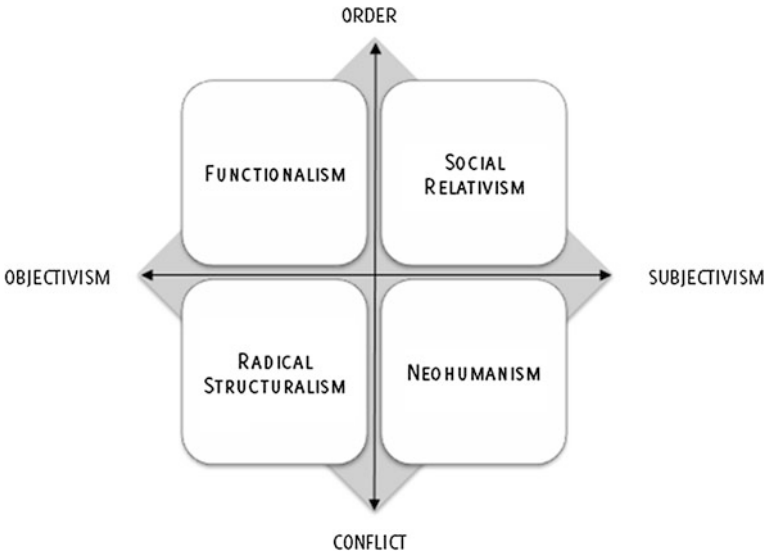

Fig. 1.1 Four paradigms of IS development (adapted from Hirschheim and Klein 1989)

neohumanist paradigm emphasizes the role of different social and organizational forces in exercising change. IS development is shaped by the rationality of human action (Hirschheim and Klein 1989).

Iivari and Hirschheim (1996) build on this concept to define three major aspects that shape the modeling of IS and can be used to determine different underlying paradigms: the organizational context and user base (host organization), the topic of interest to the users (universe of discourse), and computers (technology). They are common across the board of IS methodologies; however, there is great variety in how each information system is conceived at each level (Iivari and Hirschheim 1996). An approach that focuses on the technical level, for example, will have its emphasis placed on methodic planning and design, and prototyping.

Iivari et al. (1998) eventually expanded the four paradigms into a set of new five approaches. The *interactionist approach* focuses on the social use of IS and defines IS as institutions, with complex and overlapping interactions and negotiations between actors. The *speech act-based approach* focuses on communications and communicative action and perceives the IS as a communication system that mediates speech acts or a formalization of professional language. The *soft systems methodology approach* focuses on the learning methodology and the IS as a support system for human activity. The *trade unionist approach* focuses on the worker and perceives computers as tools, and IS as support systems for working relationships, built with collective participation. The *professional work practice approach* aims at combining performance and management principles and perceives that IS development requires a balance between methodological and practical approaches (Iivari et al. 1998).

The discussion of paradigms and approaches is important because it allows to determine a broader context for different IS development practices and provides them with a position within the frameworks of systems analysis and general social sciences. On the other hand, this also allows for a better understanding of how principles of general scientific paradigms can improve systems development (Iivari et al. 1998).

Paradigms established through research are intimately connected with systems development in practice. A paradigm does not constitute a methodology for practical interpretation. However, existing examples in practice are the fundamental drive behind the definition and further research of these paradigms. An existing system becomes part of a body of knowledge that can further fuel the body of research. But that system can also gain from the existing body of research, by adopting certain of its principles. Therefore, for some authors, systems development provides researchers with the necessary component of experience that can further the advancement of research (Nunamaker et al. 1991). Thus, it can be asserted that paradigms are useful tools that can aid in the process of systems development, by providing simple frameworks that can be identified with the organization's culture and goals.

1.3 IS Development Life Cycles

A system development life cycle (SDLC) is a framework oriented toward the description of the sequence of activities or stages that a given product goes through between its conception and its implementation or acceptance. Generally, all projects go through these stages, but there are numerous different models of SDLC that are more or less appropriate to particular types of project. The developers have to pinpoint the characteristics of their project and figure which of the SDLC models is more useful for their situation (Massey and Satao 2012).

The concept of SDLC emerged as a framework for software development in the late 1960s, particularly oriented toward large-scale developments under traditional methodologies. However, it has since then evolved to become a general concept for systems development of any kind, including IS (Patterson 2004). Some life cycle models have also attempted to break from the rigid structure of initial concepts and approach the more flexible agile methodology.

SDLC can be divided into two generic types. First, there are the waterfall-type models, thus named due to the seminal work of Royce (1970) who outlined an SDLC model of successive stages sequenced downward like the flow of a waterfall (see Fig. 1.2). This model essentially presented the ideal strategy for a development project, by outlining some principles of good practices, such as design before coding, rigorous documentation of each stage, and appropriate planning (Munassar and Govardhan 2010). It described the development project in a sequence that can be summarized in five steps: analysis, design, coding, testing, and implementation (Balaji and Murugaiyan 2012). It is, essentially, a description of a product's

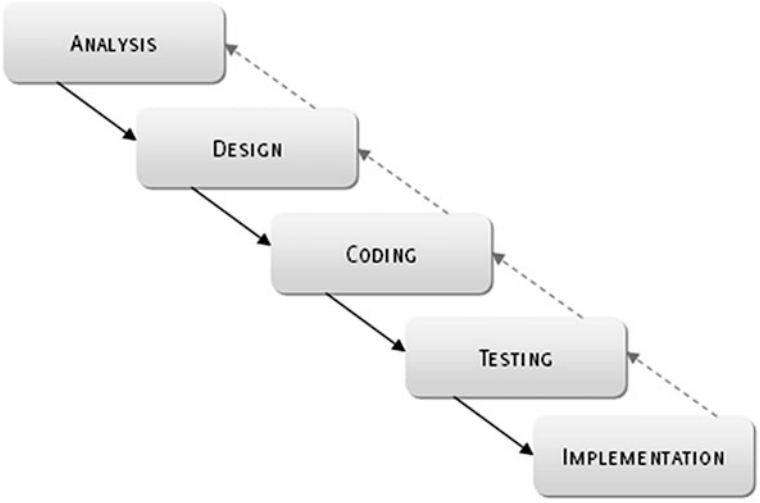

Fig. 1.2 Waterfall model (adapted from Balaji and Murugaiyan 2012)

development under the perspective of traditional methodologies, with an emphasis on the process, rigorous documentation, and self-contained stages. It was the first approach to SDLC in research.

The second type of SDLC comprises the incremental-type models. The incremental model contradicts the waterfall principle of developing a system in a single-pass process, with rigorous documentation and an extensive testing stage, to produce a final, fully usable product in the end. Incremental models instead propose developing a system in successive builds or increments. With each build, the system is designed and developed, and a working version or prototype is implemented. Users can then test it actively, within working contexts, and provide valuable feedback. This feedback will then be used as a starting point for the next build. With each successive build, the system becomes more complete, more functional, and closer to what the users intend (Massey and Satao 2012).

Most models of SDLC can be considered variants of the waterfall model, the increment model, or a combination of both. By introducing innovative concepts within the structure of the original models, or by bringing together the strong points of each one, researchers have attempted to build ideal models of SDLC for many years, resulting in a great variety of different approaches.

The V-model was an adaptation of the waterfall model that attempted to emphasize the testing stage, by proposing that each stage of the process entails a certain type of testing activity. It was presented in the shape of a V. The first sequence of events moves downward like the waterfall model, from analysis of requirements, to high- and low-level design, and coding. Once coding is complete, and a new sequence of actions moves upward, comprising all the different testing phases that should be followed: unit testing, integration testing, system testing, and acceptance testing (Balaji and Murugaiyan 2012).

The spiral SDLC model (Boehm 1988) proposed a much more complex approach to the incremental model, where development of the system is built in successive waves, much like the growing arms of a spiral, while also introducing the concept of risk analysis in the process.

The rapid application development model, or RAD, was an adaptation of the incremental model for projects that had very restricted time limits, as it was based on the concept of establishing time boxes for the development of each build, in an attempt to bring together IS development and the business goals of the organization (Gottesdiener 1995).

SDLCs can be seen as context-specific applications of the principles of the various system development methodologies. The dichotomy between traditional and agile has a parallel in waterfall versus incremental, albeit not an exact one. While methodologies allow for the organization to position the desired information system within the larger context of the project's needs and goals, development life cycles describe the system's development process in detail, from conception to deployment. Pinpointing the appropriate SDLC for a given project can provide developers with a valuable tool for organization and management.

1.4 IS Development Methodologies

An IS development methodology (ISDM) can be defined as a "system of procedures, techniques, tools, and documentation aids, usually based on some philosophical view, which help the system developers in their efforts to implement a new information system" (Avison and Fitzgerald 1995, cited by Avison and Taylor 1997). Iivari et al. (2001) define IDSM as a set of specific instructions or procedures, constituting a model or general guideline for the goals, tools, and steps necessary to build a system.

Toward the end of the twentieth century, most ISDM that were in practical use by organizations and companies were either *structural* or *object* methodologies (Tumbas and Matkovic 2006). Essentially, structural methodologies were characterized by rigid, step-by-step descriptions of the flow of activities that constitute the development process, from the analysis of the system's requirements to the design and eventual implementation and maintenance of the final product. Each step is rigidly determined, and there are no overlaps. Object methodologies focused on the dynamic aspect of the process of development and perceived each stage in the process as part of an evolutionary chain of events, leading to the notion of iterative or incremental development, where the system is released in a preliminary version, and subsequent versions improve and complete it.

Both structural and object methodologies are now commonly referred to as *traditional methodologies*. In essence, traditional development advocates single-pass development through successive stages, based on extensive documentation and a rigid perception of requirements. Methods outlined under the traditional scope aim at being as simple as possible, because the goal is often to make them adaptable

to as many different projects as possible. This led some researchers and developers alike to find such methods inadequate for the fluid nature of development projects (Hardy et al. 1995).

As IT and IS became more complex, developing projects were increasingly constrained by external factors such as budgetary and time limits, unstable user requirements, and the constant evolution of available technology (Tumbas and Matkovic 2006). Toward the end of the 1990s, a new category of ISDM has surfaced that is commonly referred to as agile development, and its increasing popularity has reshaped the research on ISDM during the last decade. The most popular form of agile development in recent years is the scrum methodology (VersionOne 2013), which is particularly flexible and can account for requirement changes at any point of the process, making it ideal for commercial projects (Fig. 1.3).

Avison and Taylor (1997) classify the different ISDM according to five different types, which are ultimately based on the scope of the problem situation that the system aims at resolving. We have summarized these findings in Table 1.1.

The first class consists of well-defined problems, with clear requirements and objectives. This class encompasses the more traditional methodologies, which divide the development process into a given number of stages, starting typically with analysis of requirements and ending with the product's final release and maintenance, with no overlapping between stages (Avison and Taylor 1997). An example is the structured systems analysis and design methodology (SSADM). This methodology follows a set structure of eight stages, starting with strategic planning and feasibility studies, and ending with production, maintenance, and review of the final product (Goodland and Riha 1999). Although later alterations can be made, it is not an incremental methodology, as the product is only released when it is

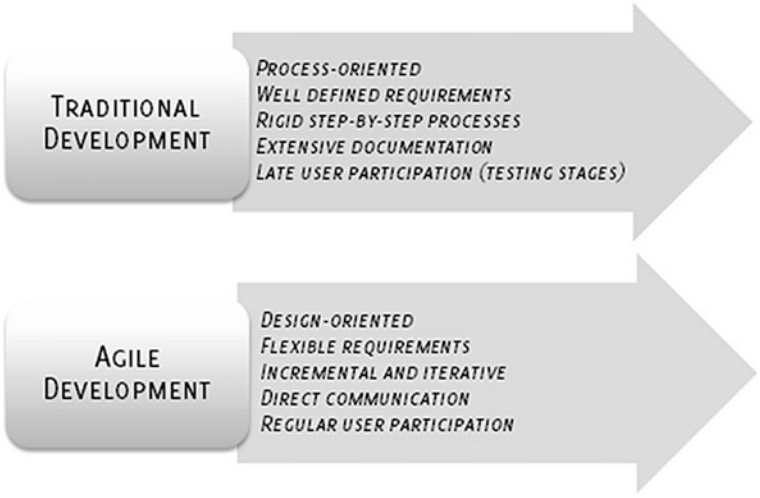

Fig. 1.3 Traditional versus agile development

Table 1.1 Different types of ISDM, based on Avison and Taylor (1997)

Problem situation	Requirements	Methodologies
Well defined	Clear	Technical, rigid, hard approaches. Ex.: SSADM
Well defined	Unstable	Technical, rigid, problem-oriented and focused on tools rather that stages. Ex.: STRADIS
Unstructured	Unstable	Soft approaches, context and user-based. Ex.: SSM
N/A	Unstable	User-centric and focused on subjectivity. Ex.: ETHICS
Complex	Unstable	Contingency models, hybrid approaches. Ex.: Multiview

complete. SSADM was originally intended for use by government entities and large projects, so it is ideal for stable requirements and is heavily reliant on documentation (Schumacher 2001).

The second class of ISDM includes all methodologies that are applicable to well-structured problem situations, where the ultimate goals are clear, but where user requirements are likely to change along the process (Avison and Taylor 1997). Structured analysis and design of information systems (STRADIS) is an example of this class of ISDM. It is essentially a traditional methodology; however, it focuses heavily on the tools necessary to solve specific problems, instead of attempting to outline a generic set of stages that should be followed for all situations, therefore making it a much more problem-oriented solution (Britts 2011).

The third class of ISDM is comprised of methodologies which are applicable to unstructured problem situations, where objectives and requirements are unclear and most likely unstable (Avison and Taylor 1997). Such situations call for an approach that focuses on the wider context of the project, and the subjective views of the users and developers, thus these methodologies are commonly known as "soft" approaches (as opposed to "hard" approaches that emphasize the technical processes and tools). The prime example is the soft systems methodology (SSM) which was precisely intended to bridge the gap between the different (and often conflicting) views of the stakeholders involved in the development project. To achieve this, SSM relies on the building of conceptual models that synthesize the problem situation, facilitating its simplification (Sánchez and Mejía 2008).

The fourth class of ISDM consists of methodologies that are applicable to situations where user interaction is very high and/or where user acceptance is a major factor, such as in highly commercial projects. An example is the effective technical and human implementation and computer-based systems, or ETHICS methodology, an approach which is heavily focused on user participation and the impact of the system on the working environment of the users (Avison and Taylor 1997).

The fifth and final class of ISDM comprises situations where the problem situation is too complex, requiring contingency solutions to the system development (Avison and Taylor 1997). Such situations are usually met by resorting to hybrid methodologies that pick aspects from various others, in order to reach a solution that is appropriate for the particular situation at hand. The Multiview methodology is an example of this hybrid approach.

Essentially, ISD methodologies are specific theoretical constructions of what exactly is necessary to build a system. The great variety of existing methodologies is rooted on the reality that each system has a particular context—not only organizational, but social and technological as well—and the methodology to build that system will be influenced by what particular goals and philosophies the stakeholders are trying to promote or focus on. Thus, methodologies determine the tools and techniques that will be used to create or improve a system and are more specific and practice-oriented constructs of IS research than the previously discussed paradigms.

1.5 Web Site Development Methodologies

Traditionally, projects that involved the creation and development of Web applications and sites were managed much in the same way as any other software development projects, and the corresponding methodologies were used. However, even during the first years of widespread commercial use of the internet, researchers have pointed out that there are very particular aspects to Web development which give rise to particular needs, when it comes to developing a new product or system.

Developers had been faced with this reality, but the solution was often to implement ad hoc strategies, without the systematic, methodical, and rigorous approach that characterized traditional software development. This issue was further emphasized by the rapid growth of the Internet and the perceived need by many companies and organizations to quickly "be on the Web," leading to rushed development processes (Murugesan et al. 2001).

In 1998, a group of researchers and developers attempted to address this issue in the first Workshop on Web Engineering, where Web Engineering was presented as a new discipline of software engineering, focusing on the inherent aspects of Web development that require appropriate solutions. A set of guidelines was determined, essentially adapting key constructs from software development methodologies to the reality of the Web. Their ultimate goal was to establish "sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality web-based systems and applications" (Murugesan et al. 2001).

In Web development, there is more emphasis on design as a process stage, because Web developers cannot control the environment in which potential users are going to use the product. A wide variety of user preferences, as well as the awareness of existing competition, create a prominent need to make the Web site or application immediately distinctive and usable, thus making design a fundamental aspect, and introducing a component of esthetic creativity that is not present in traditional software development.

The object-oriented hypermedia design methodology (OOHDM), proposed in 1995 by Schwabe and Rossi, breaks down the design process into three dimensions: conceptual design, navigational design, and abstract interface design, after which

follows the stage of implementation of the product. Conceptual design involves the creation of a conceptual model of the Web site that produces a set of classes, subsystems, and their relationships. Navigational design implies the description and visualizations of the navigational structure of the Web site, through varied navigational classes such as nodes, links, indexes, and tours. The abstract interface design then interprets the conceptual model and the navigational structure into interface classes—text fields, buttons, etc. Throughout the entire design process, OOHDM uses object-oriented modeling as its main tool, hence its name (Schwabe et al. 1999). It is ultimately a methodology that aims at helping developers and designers create single-user hypermedia environments, but researchers have observed that it is not adequate to projects that want to embed authoring functions in the Web site or application, permitting users to edit and add content (Schümmer et al. 1999).

Similarly, the relationship management methodology (RMM) focused on hypermedia applications, as the vehicle for the relationships between objects. Developed by Isakowitz et al. (1995), it is a structured, step-by-step methodology. The process starts with rigorous analysis of the Web site's objectives, the market, and the user base, as well as information sources, permissions, distribution channels, and other business-related principles. Then, much like the OODHM, the design process is broken apart, in this case, in six stages related to different dimensions of design, as outlined in Fig. 1.4.

While OOHDM and RMM are adaptations of traditional, rigid IS development methodologies, other methodologies have attempted to bring a more holistic approach to Web development, in accordance with the large scope of goals and needs of Web projects. The Web information system development methodology (WISDM) was developed by Vidgen et al. (2002) in an attempt to combine essential principles of the Multiview IS development methodology with the specific characteristics of Web projects. Multiview is a contingent, goal-oriented solution to the development of IS projects with complex and diffuse needs and requirements. Likewise, WISDM posits that a unified approach that brings together the different levels of the development project, proposing a socio-technical approach. The development process is broken apart into a four-stage framework. The analysis stage is divided into organizational analysis (where goals of the Web project are integrated into the organization's general strategy) and information analysis (where

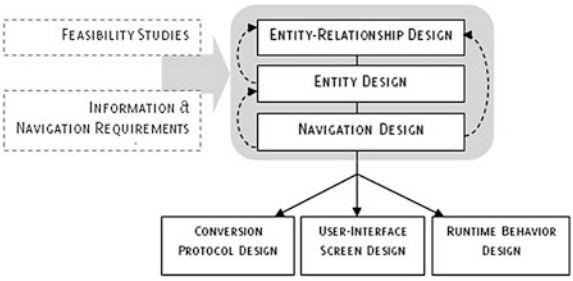

Fig. 1.4 Design processes of the RMM (adapted from Isakowitz et al. 1995)

requirements are specified). The design stage is also divided into two processes: work design (where the characteristics of the Web project are developed in line with user/customer needs) and technical design (where the project is physically developed through programming), while user-interface design bridges the two processes. This methodology stands out due to its heavy emphasis on the creative aspects of Web development, while more traditional methodologies are too reliant on IS-specific terminology and principles.

Much like IS development methodologies, all Web development methodologies ultimately aim at aiding in the creation of products that are efficient and appropriate not just to the organization's goals, but to the users. However, Web methodologies forcibly need to take new aspects into account, namely an exceedingly diffuse user base which cannot be contacted directly for the most part, and the need to differentiate the product at an esthetic level, so as to permit users having a first contact with the Web site or application to immediately feel a positive relationship with the content. This has introduced specific characteristics to the Web development methodologies, namely a great emphasis on design processes.

1.6 Usability Evaluation Models

In IS development research, one question in particular has generated a considerable amount of attention: How can developers and managers effectively determine whether given IS are being successful in accomplishing the goals they were developed for? How to assess the degree to which the system is improving the general working principles of its users?

The issue of usability is of key importance in this field. Usability essentially refers to the degree to which a system is easily learned and used by its users. Some researchers have focused on the study of cognitive processes as a way to define usability principles that are directly inferred from those processes, hence more appropriately matched to the way users behave and think.

According to Norman (1993), there are two dimensions to human cognitive processes. The experiential mode refers to perceptions, actions, and reactions, while the reflective mode implies thinking, reasoning, comparing, and making logical decisions. It is argued that specific modes of cognitive experience require different technologies and systems. The proposed field of cognitive engineering specifically focuses on the development of systems that support users' cognitive processes, in an attempt to facilitate the adjustment to the system, and reduce the difficulty and complexity of the system, using human–computer interaction (HCI) principles.

Similarly, to this approach, researchers have attempted to define models so as to aid developers in determining the adequacy of their system to their respective users, during the testing and evaluation stages of the development process. Nielsen (1994) observed a number of different methods of evaluating usability, summarized as follows:

- Heuristic evaluation—informal methods where usability experts evaluate HCI dialogues according to established principles (heuristics), specific to the project;
- Cognitive walkthroughs—detailed procedures where a user's problem solving process is simulated, and it is analyzed whether the process will lead to the correct, expected actions or not;
- Formal usability inspections—rigid procedures that follow well-defined roles and combine heuristic evaluations with simplified forms of cognitive walkthroughs;
- Pluralistic walkthroughs—meetings where users, developers, and other stakeholders discuss scenarios and dialogue elements;
- Feature inspection—a thorough inspection of features, sequences, processes, and all aspects that users can eventually come across, pinpointing what aspects are exceedingly unnatural or require excessive experience/knowledge;
- Consistency inspection—the designers inspect and compare interface features from multiple projects;
- Standards inspection—an expert on a specific interface standard inspects the project for compliance.

There have been other methods and methodologies established for the better evaluation of usability. Card et al. (1983) proposed the GOMS model, where four essential constructs are emphasized—goals, operators, methods, and selection rules, giving the model its acronym. Goals are the specification of user needs and objectives. Operators are the specific objects that will physically describe the HCI. Methods are programs built from the operators, designed to facilitate the accomplishment of the goals. Selection rules then help predicting which method will be more appropriate for specific situations. The ultimate goal of this methodology is to bridge the gap between the psychological level, where the users' cognitive processes develop, and the concrete, physical level, where the system acts.

Pirolli and Card (1999) in turn describe an adaptive control of thought in information foraging model (ACT-IF) which is essentially derived from the theories of evolutionary psychology. The process by which users search and gather information is illustratively compared to the process of food foraging, and it is asserted that users will follow "scents," which, in the context of IS, are the perceptions of value, cost, accessibility, obtained from instinctive cues such as citations, links, and icons. The stronger and more evident these cues are, the more likely the user is to make correct choices that fulfill his/her needs. Thus, developers need to focus on methods to appropriately direct users to the information they need.

Usability evaluation models are always interrelated with psychological concepts, particularly in the field of cognitive theory, and research in one field accompanies research on the other field. Resorting to essential principles and theories on how the human mind seeks and absorbs new information and new knowledge, researchers on IS usability have attempted to use those principles to establish good practices of development, where developers of new systems take into account the basics of human psychology to build systems that adequately adjust to the psychological framework of its users. This is a means to ensure that the system is successfully accepted.

1.7 Quality Evaluation Models

Technology acceptance has been a very active subject of research, not just for the field of IS, but for marketing as well. For developers and managers alike, it is crucial to evaluate by which processes will users or customers adopt and successfully accept a given system or technology, or reject it altogether. In order to determine this, the stage of implementation, as well as any other stages following that, is fundamental. It is also of key importance to understand the constitution of the user base, its contextual background, their needs, objectives, and obstacles. Finally, researchers have also borrowed concepts from behavioral psychology, going to the deeper level of human behavior to understand the processes by which people make their choices to use or discard tools.

A pioneering approach on this issue was the theory of reasoned action (TRA), developed by Fishbein and Ajzen (1975). It asserts that there are four different variables that influence behavioral action: beliefs, attitudes, intentions, and behaviors. The model describes the relationships between these factors. Essentially, beliefs and evaluations shape the user's attitude toward behavior; normative beliefs and the user's motivation to comply with them shape the subjective norm. Beliefs and subjective norm will then shape the user's behavioral intention, leading to a result of an actual behavior. This premise was later adjusted by Ajzen (1991) in his theory of planned behavior (TPB), where the relationships and variables involved in the process are analyzed in more depth. According to the TPB model, beside behavioral and normative beliefs, there is a third factor that will influence the user's intentions: control beliefs, related to the user's perception of whether he/she can effectively use the new system. Both TRA and TPB models are essentially behavioral theory models that can be adapted to the context of IS acceptance.

However, one of the most popular approaches on this field was the technology acceptance model (TAM), proposed by Davis (1986). It describes the means by which subjective elements, such as a user's perception of the system's usefulness, will influence objective elements, such as system use. Once key design features are implemented and also considering other external influences (such as personal context, organizational structure, and socioeconomic background), users will form a cognitive response based on their perception of the new system's functionality and usability (perceived usefulness). This will generate an affective response, translated in their attitude toward use of the system, and eventually a behavioral response, which is the actual use of the system (or its rejection). This model thus establishes a causal relationship between user's perceptions of the system and their choice to use it (see Fig. 1.5).

TAM is an exceedingly simple model, which has led it to be a very popular option for researchers, because it can easily be adjusted to a variety of contexts. On the other hand, it has also been the subject of frequent criticism, namely due to the vague characterization of its core constructs and relationships. For this reason, there have been attempts at building more consistent and complex models on this simple premise. Venkatesh and Davis (2000) proposed the TAM 2, whose ultimate goal

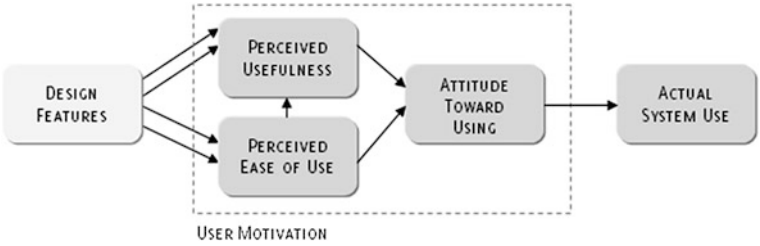

Fig. 1.5 Technology acceptance model (adapted from Davis 1986)

was to provide a description of the specific mechanisms by which perceived usefulness is formed, considering that it was the most fundamental factor in the original TAM model.

Essentially, all quality evaluation models have attempted to bring together the key aspects that form or influence user's cognitive processes and behavioral decisions. Venkatesh et al. (2003) combined eight existing models found in previous literature to create what they described as the unified theory of acceptance and use of technology (UTAUT). They started by outlining a list of constructs used in the existing models and pinpointed which constructs appeared to more useful and significant in empirical research. From there, they determined that the more important factors of user acceptance could be summarized in four variables: performance expectancy, effort expectancy, social influence, and facilitating conditions. External factors, such as gender, age, experience, and voluntariness of use, acted as moderating elements over those variables. The differing levels of impact resulted in particular behavioral intentions, and use behaviors.

These and other models of technology and IS acceptance have in common the importance of individual perceptions, although different theories consider different factors to be of influence in shaping those perceptions. These models are particularly useful for developers and designers, allowing them to adjust the models to the project, and determine what factors will most likely determine the user's acceptance of the final released product.

1.8 IS Models for Success Assessment

As we have seen, the concept of IS success has been closely interrelated with the concept of user acceptance, in accordance with behavioral theories. The pioneering work of DeLone and McLean (1992) established the basics for the creation of a model of IS success assessment, centered on the premise that use of the system is intimately related with user satisfaction. It attempted to describe the acceptance of a system through a causal–explanatory approach, where use and user satisfaction, constantly feeding on each other, directly influence individual impact, which eventually reflects on organizational impact (Iivari 2002). This model was later

Fig. 1.6 D&M model of IS success (adapted from DeLone and McLean 2003)

adapted by the authors to include a more comprehensive perspective of system quality and a more encompassing concept of organizational impact (described as net benefits of the system) (see Fig. 1.6).

The authors argued that for a model of IS success to be truly useful, it had to have as few variables as possible, so as to make it suitable for the great variety of different realities and systems that exist in practice (DeLone and McLean 2003), and this principle justifies the model's simplicity, which has made it one of the most popular—and scrutinized—approaches to IS success in research.

Seddon (1997) attempted to break down the simple concepts of the D&M model by offering a slightly different perspective, particularly on the idea of use/user satisfaction. The subsequent model, named the Seddon model, substituted the concept of use by that of perceived usefulness, thus introducing expectations as key variables in the process. Expectations about the net benefits of future use of the system will lead to use of the system (Seddon 1997). Use, in itself, is not a measure of success but a behavior. User satisfaction, on the other hand, is influenced by a great number of factors, including system quality, information quality, perceived usefulness, individual net benefits, organizational net benefits, and societal net benefits. Later adjustments of the Seddon model introduced the concepts of group impact and external impact, to account for the influences that the user can be subjected to from his/her peers or from his/her social context (Kurian et al. 2000).

Other authors have equally attempted to build on the D&M model, expanding or breaking apart some of its essential concepts, particularly user satisfaction.

The 3D model (Ballantine et al. 1996) analyzed the concept of IS success as a three-dimensional construct related to three different stages of IS development: development, deployment, and delivery. Development pertains to the actual creation of the system (design, coding, etc.). For the system to be successfully deployed, it has to cross a barrier called the implementation filter, comprised mainly of factors relating to the user's expectations, involvement, experience, and possibility of choice. After the system's been deployed—used by its users—there is an integration filter, where factors such as strategy, organizational culture, and organizational structure will determine the degree to which the system fits in with the existing organization. Finally, for the system to be successfully delivered, it has to pass the environmental filter, where competitor movements and economic and political contexts exert their influence (Ballantine et al. 1996).

The IS impact measurement model, on the other hand, focused on two fundamental aspects of success measurement: impact and quality (Gable et al. 2008). It describes IS success as a result of a combination of different factors: quality (system and information quality), satisfaction, and impact (individual and organizational). Instead of perceiving these factors as elements within a causal process, all factors are independent and exert their influence through various degrees, with one common output, IS success. Notably, this model does not consider use of the system as a significant factor, because there are various instances where system use does not depend on other variables and is mandatory regardless of user perceptions, leading the authors to exclude it (Gable et al. 2008).

The success of a given information system within its organization is a difficult aspect to describe with precision, because it is subjected to numerous influences. Different researchers are focused on different variables with more or less emphasis, leading to the creation of various models whose adequacy to describe IS success will depend on the purpose of this assessment. Simpler models such as D&M are ideal for broader considerations. But more specific, quantitative approaches will require more complex models, such as the 3D model.

1.9 Conclusions

We have analyzed the key aspects and contributions to the body of research on IS development and success measurement. Paradigms, methodologies, SDLC models, and success evaluation models are all theoretical constructs that aim at systemically describing the complex reality of IS, in a way that simplifies not just future research, but also the work of developers and managers in determining the principles and methods of their projects.

There are three degrees for the theoretical approach to IS, illustrated in Fig. 1.7. Paradigms offer the broadest perspective, ultimately consisting on the insertion of different approaches to IS development within the context of a particular philosophy or global view on goals and requirements.

At the development level, IS development methodologies, systems development life cycles, and Web development methodologies propose varied systematic approaches to the development process, describing sets of stages, activities, and roles necessary to achieve successful and efficient development.

Finally, usability models, quality evaluation models, and success assessment models allow managers and developers to determine the degree to which the system is adequate to the goals, needs, and intentions of the users.

As we have seen, the characteristics of the project determine what model or methodology should be adopted in order to facilitate the development process. In that sense, a comprehensive study of the different approaches and methods of IS development can be a valuable tool for developers. We have determined that there are two principles of IS development: traditional, structured, rigid methods and agile, incremental, flexible methods. The first category is suitable for large projects,

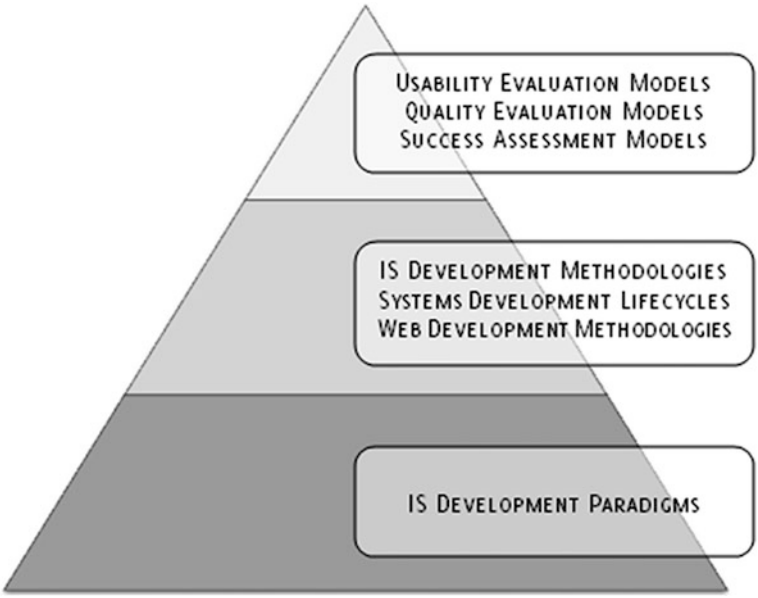

Fig. 1.7 Different levels of IS research

where requirements are well established, face-to-face communication is not efficient (as opposed to documentation), and user participation is not necessary at all times. A good example is government projects, where methodical organization and rigor are essential. The second category is suitable for medium- to small-sized projects, heavily user-centered, where requirements are likely to change and there is constant feedback between developers and users alike. This is the ideal approach for many commercial software projects.

In regard to evaluation of the project, usability models are appropriate primarily to determine how the system can closely interrelate with the user's cognitive and learning processes, thus facilitating their adaptation to it. Quality evaluation models allow developers to determine what will shape the user's acceptance of the new system, and success evaluation models will help developers in measuring the implementation of the system, providing valuable metrics and feedback for future updates and/or systems.

References

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes, 50*(2), 179–211.

Alter, S. (2008). Defining Information Systems as work systems: Implications for the IS field. *Business Analytics and Information Systems,* Paper 22.