SCHOOL OF SCIENCE AND ENGINEERING

VLADIMIR P. OLIVA
UD39360SIN48025
PhD. in IT program
(Candidate)

Core Program: CP1
Systems Design and Implementation --

“A tool for
IT Systems and Analysis in Business Application Development”

Atlantic International University
Honolulu, Hawaii
June 2016
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>3</td>
</tr>
<tr>
<td>2. DESCRIPTION</td>
<td>5</td>
</tr>
<tr>
<td>3. GENERAL ANALYSIS</td>
<td>6</td>
</tr>
<tr>
<td>4. CONCEPTUAL REVIEW</td>
<td>9</td>
</tr>
<tr>
<td>4.1. Object-Oriented Design Process</td>
<td>9</td>
</tr>
<tr>
<td>4.2. Other popular methodologies</td>
<td></td>
</tr>
<tr>
<td>4.2.1 Waterfall model</td>
<td>11</td>
</tr>
<tr>
<td>4.2.2. Software prototyping</td>
<td>12</td>
</tr>
<tr>
<td>4.2.3. Iterative and Incremental</td>
<td>13</td>
</tr>
<tr>
<td>4.2.4. Spiral model</td>
<td>14</td>
</tr>
<tr>
<td>4.2.5. Rapid application development</td>
<td>16</td>
</tr>
<tr>
<td>4.2.6. Extreme programming (XP)</td>
<td>17</td>
</tr>
<tr>
<td>4.2.7. Agile software development</td>
<td>18</td>
</tr>
<tr>
<td>4.2.8. Scrum Methodology</td>
<td>20</td>
</tr>
<tr>
<td>5. PROS AND CONS FOR SYSTEM DESIGN AND IMPLEMENTATION</td>
<td>24</td>
</tr>
<tr>
<td>5.1 Pros</td>
<td>24</td>
</tr>
<tr>
<td>5.2 Cons</td>
<td>25</td>
</tr>
<tr>
<td>7. Conclusion</td>
<td>29</td>
</tr>
<tr>
<td>References</td>
<td>32</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

System’s design and implementation is a concept in the software engineering process or systems’ software development by which an application in business or organization is executed to perform efficiency. The activities in the system are characterized by the methodology of the developer and sometimes depending on the difficulty of the software to be constructed. In a broader sense of the term, it illustrates all the conceptualization of the desired system until the final deployment of the software, ideally in a planned and structured manner.

Consequently, the system’s design will require creative activity to elaborate the optimum solution, identify important components and their relationships, and discuss the final outputs for the customer’s requirements and developer’s mutual understanding.

The case of implementation, it include the extent of outcomes and the effectivity using the desired results needed by the clientele. In general, it is the process of realizing the design as a program/ system. [1]

Business understanding of the system is that it is developed to their preference and not the intent of the developer. Here are some goals why a system is developed for a specific clientele: to meet the requirements of set of potential users; to provide real time information of the business/ operation; or for personal use (Ulrich, et.al. 2000). Those goals are considered part of the development phases to execute/make the research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products.

Present systems, the use of communication technology is a good avenue for levelling up new application that converge mobile devices, office computers, network devices and influence the decision making of the business operation (Zuppo,n.d.). The
design may assist in security and data manipulation and emphasized extent to provide data privacy.

On this digital communication age, the Web is a powerful area of information exchange and it is also the place where data might be corrupted and makes the business inconsistent with its objectives. The design will protect data from the front-end to back-end process.

“Everyone designs who devises courses of action aimed at changing existing situations into preferred ones. The intellectual activity that produces material artifacts is no different fundamentally from the one that prescribes remedies for a sick patient or the one that devises a new sales plan for a company or a social welfare policy for a state.” - Herbert Simon (Nobel Prize Winner & Carnegie Mellon professor).
2. DESCRIPTION

The concept of design and implementation as a tool for business application warrants the so many avenues of paradigms and models depending on the requirements of the business clientele. Some studies shows that products of business software uses new models (agile models) for small businesses. The word “agile” refers to software methods that introduce agility in the process of development and determine a well-established practices. [3]

As mentioned by Fowler (2005), “This new movement grew out of the efforts of various people who dealt with software process in the 1990s, found them wanting, and looked for a new approach to software process. Most of the ideas were not new, indeed many people believed that much successful software had been built that way for a long time.” It is imperative to note that the complexity of the software development process also changes to conform the development of IT in business.

Looking at the finish products, some buyers look at the software as a tool for business efficiency and want it used immediately. They assume that models used are objective enough to satisfy their need. Small businesses uses this assumption. However, the big businesses use finish software only as a starter but tends to use customized software for their special need and data security. As an experienced programmer for ten 10 years, many business clientele wanted to use “buy” software at the soonest but tends to change buy requesting customization of scalable system that conforms with time. “Building your own software that is specifically tailored to your company’s needs, as well as focused on scalability and efficiency, can help mean the difference between offering a commoditized service and offering a highly differentiated one at a better price”, Chuck Cohn.
3. GENERAL ANALYSIS

3.1. How Communication Technology Services help in Business Operation?

This part of conceptualization is inclusive in the design since communication technology demands integration and sharing of data.

As mentioned by Amanda C. Kooser (2009), Demand Media, that “communications are instantaneous, huge amounts of information move through email and the Internet and powerful tools are in the hands of owners and employees. Innovations in technology have improved operations at companies of all sizes and helped turn small local businesses into global businesses”. This view is an elaboration of communication technology that changes not only our life but our belief. It gives us realtime information anywhere. The concepts that support these changes are summarized below.

**Business**

“Business communication is the sharing of information between people within and outside the organization that is performed for the commercial benefit of the organization. It can also be defined as relaying of information within a business by its people” (Wikipedia, n.d.). For effective business communication using technology, users may apply face-to-face, emails, social networking, and telephone (voice messaging). These things becomes part of their lifestyle.

Mobile devices have become tool for business owners/employees to keep in touch with their business. This can be done at realtime including the feedback and decision making. Intranet might be functional in this manner as the communication in the company becomes available to its clientele.
Marketing

“Marketing communications uses different marketing channels and tools in combination: Marketing communication channels focuses on any way a business communicates a message to its desired market, or the market in general. A marketing communication tool can be anything from: advertising, personal selling, direct marketing, sponsorship, communication, promotion and public relations” (Wikipedia, n.d.).

This is an option in contrast with the media prints where reaching the clients/ customer becomes fast and informative. Mobile devices include this technology wherever you are using small application that is fun and entertaining.

Productivity

According to Amanda Kooser, “Small businesses need to wring every ounce of productivity out of their operations and technology tools help employees get tasks done more quickly”. This tells us that with the right logistics and system, the employees and their supervisors will maximize time and produce more output at a negligible cost.

The productivity using technology platform in business creates an atmosphere of transparency and true business success. The system ensures the business to be guided with tools in their day to day decision making. “Increased business productivity can be traced to the automation of processes allowing for faster communication of strategy, increased time spent on strategic priorities and greater project completion rates ” (Success factors, 2016).

Customer Service

This is the provision of service to customers before, during and after a purchase. “Customer service concerns the priority an organization assigns to customer service relative to components such as product innovation and pricing.
In this sense, an organization that values good customer service may spend more money in training employees than the average organization, or may proactively interview customers for feedback” (Wikipedia,n.d.).

Customer is a very important entity in business. The needs and feedback may require absolute attention from the business sector that are timely delivered. The business should be closer to the client and vice-versa using technology. Call centers have this service as its frontline. An application system may consider the services to be more proactive.

Commercial Software/ Technology

A wide range of domains have provided the availability of commercial off-the-shelf systems (COTS) that can be adopted and tailored to the user’s requirements. For instance, if you want to implement a financial management system, you can buy a package that is already used in accounting firms. This might be cheaper at the first glance as compared to developing a system in a traditional and longer developing stages. [5]

The management must evaluate their requirements at a given time and may require I.T. experts to validate Pros and Cons for “buying” or “developing” the system to be used by the organization.

Commercial software is sometimes easy to use as they come with many threads and suggest better option as you test it. The flexibility makes it friendly to user. However, security might be compromised due to its “openness” and so many variables/ parameters might be a source of confusion. In general, it is still dependent on the company’s immediate need at that time.
4. CONCEPTUAL REVIEW

4.1 Object-Oriented Design Process

The OODP involves a number of different system models. It requires a lot of effort for development and maintenance of these models and for small systems that may not be cost-effective. However for large systems developed by different groups design models are an important communication mechanism.

Small Systems

This is sometimes confused with segmented program. A small system might be connected to a larger system and part of the extended user interface. The small system only have a limited number of maintenance data that keeps on repeating but not necessarily increasing. It might be thread to some that runs at the background. This is usually used in mobile devices where it runs fast if it is installed in the user-side.

Large Systems

By its name it is large and maintains huge amount of data. It is usually stationed in the server-side since it needs a lot of computing power and resources.

Using technology mixed with integrated application, the concept of browser-side and server-side is used to make the application more flexible to the business and hides its complexity to the user.

Example of Process Stages

There are a variety of different object-oriented design process that depend on the organization using the method. We have here an example of activities in the OODP (Weisfeld, 2009):

Identify Classes & Objects

This is the Object Oriented Approach to define the problem, identify what is and what is not part of our problem domain (abstraction).

The output of this stage is a data dictionary containing all the key elements of the problem. Data dictionaries have advantages over just maintaining a list of things that will are in the problem domain. These are:
• common and consistent vocabulary
• supports project migration (good for new team members)
• gives a global view of the project

Some sort of technique (CRC cards, drawing tool, CASE tool) is helpful here to represent each thing in the data dictionary (Armstrong, 2006).

**Class & Object Semantics**

We proceed with our list of things from the first step and develop the semantics (what our abstractions encapsulate and what they can do) of our classes and objects.

This stage is where the "two heads are better than one" truism is seen most clearly. Debate sharpens the semantics of classes and improves them. The best ideas survive, the weak ones are reformulated.

Debates about class semantics and names can become heated. According to Matt Weisfeld “since OO involves significant anthropomorphism, designers become more attached to their points-of-view”.

**Class & Object Relationships**

We formalize in our diagrams the relationships between classes and objects.

Associations are between classes. We look at groups of related classes and work on the diagrams to represent how they associate with each other.

Collaborations are more the concern of the design phase and deal with object diagrams and how objects collaborate.

**Implementing Classes & Objects**

Initially (in analysis) we do just enough of this to help show gaps and lacks with the design. Eventually the implementation becomes more and more "real". Iterations at this phase are consistent with the idea of working complex systems evolving from working simple systems.
4.2. Other popular methodologies

4.2.1 The **waterfall model** is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall, Fig. 4.2.1.) through the phases of conception, initiation, analysis, design, construction, testing, production/implementation and maintenance. [4]

The waterfall development model started in the business of manufacturing and construction supplies: highly structured physical environments where inventory requires real time updates as sales/purchases have been applied at the entry phase. Because no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development in business operation.

![Waterfall Model Diagram](image)

**Fig.4.2.1. Waterfall model of the software process (Royce, 1970)**
In Royce's original waterfall model, the following phases are followed in order:

1. System and software requirements: captured in a product requirements document
2. Analysis: resulting in models, schema, and business rules
3. Design: resulting in the software architecture
4. Coding: the development, proving, and integration of software
5. Testing: the systematic discovery and debugging of defects
6. Operations: the installation, migration, support, and maintenance of complete systems

With this, each phase is being reviewed and satisfied before moving to next phase.

4.2.2. **Software prototyping** is the activity of creating prototypes of software applications, i.e., incomplete versions of the software program being developed. It is an activity that can occur in software development and is comparable to prototyping as known from other fields, such as mechanical engineering or manufacturing (Wikipedia, n.d.).

A prototype typically simulates only a few aspects of development, and may be completely different from, the final product. It has several benefits including the following (Thayer, 2002):

The software designer and implementer can get valuable feedback from the users early in the project. The client and the contractor can compare if the software made matches the software specification, according to which the software program is built. It also allows the software engineer some insight into the accuracy of initial project estimates and whether the deadlines and milestones proposed can be successfully met.

The original purpose of a prototype is to allow users of the software to evaluate developers' proposals for the design of the eventual product by actually trying them out, rather than having to interpret and evaluate the design based on descriptions. Prototyping can also be used by end users to describe and prove requirements that have
not been considered, and that can be a key factor in the commercial relationship between developers and their clients.

Prototyping can also avoid the great expense and difficulty of changing a finished software product.

The process of prototyping involves the following steps

1. Identify basic requirements
   Determine basic requirements including the input and output information desired. Details, such as security, can typically be ignored.

2. Develop Initial Prototype
   The initial prototype is developed that includes only user interfaces

3. Review
   The customers, including end-users, examine the prototype and provide feedback on additions or changes.

4. Revise and Enhance the Prototype
   Using the feedback both the specifications and the prototype can be improved. Negotiation about what is within the scope of the contract/product may be necessary.

4.2.3. **Iterative and Incremental development** is any combination of both iterative design or iterative method and incremental build model for software development. The combination is of long standing and has been widely suggested for large development efforts. The relationship between iterations and increments is determined by the overall software development methodology and software development process (Wikipedia, n.d.). The exact number and nature of the particular incremental builds and what is iterated will be specific to each individual development effort (Fig. 4.2.3.).
Incremental development slices the system functionality into increments (portions). In each increment, a slice of functionality is delivered through cross-discipline work, from the requirements to the deployment. The Unified Process groups increments/iterations into phases: inception, elaboration, construction, and transition.

- Inception identifies project scope, requirements (functional and non-functional) and risks at a high level but in enough detail that work can be estimated.
- Elaboration delivers a working architecture that mitigates the top risks and fulfills the non-functional requirements.
- Construction incrementally fills in the architecture with production-ready code produced from analysis, design, implementation, and testing of the functional requirements.
- Transition delivers the system into the production operating environment.

Each of the phases may be divided into 1 or more iterations, which are usually time-boxed rather than feature-boxed. Architects and analysts work one iteration ahead of developers and testers to keep their work-product backlog full.

4.2.4. The spiral model is a risk-driven process model generator for software projects. Based on the unique risk patterns of a given project, the spiral model (Fig. 4.2.4.) guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping (Wikipedia, n.d.).
Sequentially defining the key artifacts for a project often lowers the possibility of developing a system that meets stakeholder “win conditions” (objectives and constraints).

This invariant excludes “hazardous spiral look-alike” processes that use a sequence of incremental waterfall passes in settings where the underlying assumptions of the waterfall model do not apply. Boehm lists these assumptions as follows:

1. The requirements are known in advance of implementation.
2. The requirements have no unresolved, high-risk implications, such as risks due to cost, schedule, performance, safety, security, user interfaces, organizational impacts, etc.
3. The nature of the requirements will not change very much during development or evolution.
4. The requirements are compatible with all the key system stakeholders’ expectations, including users, customer, developers, maintainers, and investors.
5. The right architecture for implementing the requirements is well understood.
6. There is enough calendar time to proceed sequentially.

In situations where these assumptions do apply, it is a project risk not to specify the requirements and proceed sequentially. The waterfall model thus becomes a risk-driven special case of the spiral model.
4.2.5. **Rapid application development (RAD)** is both a general term used to refer to alternatives to the conventional **waterfall model** of software development as well as the name for James Martin's approach to rapid development. In general, RAD approaches to software development put less emphasis on planning tasks and more emphasis on development. In contrast to the waterfall model, which emphasizes rigorous specification and planning, RAD approaches emphasize the necessity of adjusting requirements in reaction to knowledge gained as the project progresses. This causes RAD to use prototypes in addition to or even sometimes in place of design specifications. RAD approaches also emphasize a flexible process that can adapt as the project evolves rather than rigorously defining specifications and plans correctly from the start (Wikipedia, n.d.).

The James Martin approach to RAD divides the process into four distinct phases (Fig. 4.2.5):

1. **Requirements planning phase** – combines elements of the system planning and systems analysis phases of the Systems Development Life Cycle (SDLC). Users, managers, and IT staff members discuss and agree on business needs, project scope, constraints, and system requirements. It ends when the team agrees on the key issues and obtains management authorization to continue.
2. **User design phase** – during this phase, users interact with systems analysts and develop models and prototypes that represent all system processes, inputs, and outputs. The RAD groups or subgroups typically use a combination of Joint Application Development (JAD) techniques and CASE tools to translate user needs into working models. User Design is a continuous interactive process that allows users to understand, modify, and eventually approve a working model of the system that meets their needs.
3. **Construction phase** – focuses on program and application development task similar to the SDLC. In RAD, however, users continue to participate and can still suggest changes or improvements as actual screens or reports are developed. Its tasks are programming and application development, coding, unit-integration and system testing.
4. **Cutover phase** – resembles the final tasks in the SDLC implementation phase, including data conversion, testing, changeover to the new system, and user training. Compared with
traditional methods, the entire process is compressed. As a result, the new system is built, delivered, and placed in operation much sooner.

![RAD Model](image)

**Fig 4.2.5. RAD model (Martin, 1991)**

### 4.2.6. Extreme programming (XP)

Extreme programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development (Wikipedia, n.d.). It advocates frequent "releases" in short development cycles (Fig. 4.2.6.), which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.

Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, simplicity and clarity in code, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels. As an example, code reviews are considered a beneficial practice; taken to the extreme, code can be reviewed *continuously*, i.e. the practice of pair programming.
Extreme Programming Explained describes extreme programming as a software-development discipline that organizes people to produce higher-quality software more productively.\footnote{6}

XP attempts to reduce the cost of changes in requirements by having multiple short development cycles, rather than a long one. In this doctrine, changes are a natural, inescapable and desirable aspect of software-development projects, and should be planned for, instead of attempting to define a stable set of requirements (Beck, n.d.).

Extreme programming also introduces a number of basic values, principles and practices on top of the agile programming framework.

4.2.7. **Agile software development** is a set of principles for software development in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change (Wikipedia, n.d.).
The Agile Manifesto

In February 17, 2001 software developers met at the Snowbird resort in Utah to discuss lightweight development methods. They published the Manifesto for Agile Software Development, in which they said that by "uncovering better ways of developing software by doing it and helping others do it," they have come to value individuals and interactions over Processes and tools, Working software over Comprehensive documentation, Customer collaboration over Contract negotiation, and Responding to change over Following a plan.

- **Individuals and interactions**: self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software**: working software is more useful and welcome than just presenting documents to clients in meetings.
- **Customer collaboration**: requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.
- **Responding to change**: agile methods are focused on quick responses to change and continuous development.

Some of the authors formed the Agile Alliance, a non-profit organization that promotes software development according to the manifesto's values and principles. Introducing the manifesto on behalf of the Agile Alliance, Jim Highsmith said,

The Agile movement is not anti-methodology, in fact many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment. Those who would brand proponents of XP or SCRUM or any of the other Agile Methodologies as "hackers" are ignorant of both the methodologies and the original definition of the term hacker. [7]

The Agile Manifesto is based on twelve principles:[8]

1. Customer satisfaction by early and continuous delivery of valuable software
2. Welcome changing requirements, even in late development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the principal measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

4.2.8. Scrum Methodology. Scrum is an iterative and incremental agile software development framework for managing product development (Fig. 4.2.8.). Agile is a response to the failure of the dominant software development project management paradigms (including waterfall) and borrows many principles from lean manufacturing. In 2001, 17 pioneers of similar methods met at the Snowbird Ski Resort in Utah and wrote the Agile Manifesto, a declaration of four values and twelve principles. The Agile Manifesto placed a new emphasis on communication and collaboration, functioning software, team self organization, and the flexibility to adapt to emerging business realities.[9]

How Does Scrum Fit With Agile?

The Agile Manifesto doesn’t provide concrete steps. Organizations usually seek more specific methods within the Agile movement. These include Crystal Clear, Extreme Programming, Feature Driven Development, Dynamic Systems Development Method (DSDM), Scrum, and others. While I like all the Agile approaches, for my own team Scrum was the one that enabled our initial breakthroughs. Scrum’s simple definitions gave our team the autonomy we needed to do our best work while helping our boss (who became our Product Owner) get the business results he wanted. Scrum opened our door to other
useful Agile practices such as test-driven development (TDD). Since then we’ve helped businesses around the world use Scrum to become more agile. A truly agile enterprise would not have a “business side” and a “technical side.” (Verheyen, 2016).

![Fig. 4.2.8. Scrum an Agile Framework](image)

There are three core roles in the scrum framework. They represent the scrum team. Although other roles may be encountered in real projects, scrum does not define any team roles other than those described below.\[13\]

**Product owner**

The product owner represents the stakeholders and is the voice of the customer, who is accountable for ensuring that the team delivers value to the business. The product owner writes customer-centric items (typically user stories), ranks and prioritizes them, and adds them to the product backlog. Scrum teams should have one product owner, this role should not be combined with that of the scrum...
master. The product owner should focus on the business side of the project and spend majority of time to liaise with stakeholders and should not interfere or interact with team members on the technical solution aspects of the development task. This role is equivalent to the customer representative role in some other agile frameworks such as extreme programming (XP).

Communication is a main function of the product owner. The ability to convey priorities and empathize with team members and stakeholders is vital to steer the project in the right direction. Product owners bridge the communication gap between the team and its stakeholders.

**Development team**

The development team is responsible for delivering potentially shippable increments (PSIs) of product at the end of each sprint (the sprint goal). A team is made up of 3–9 individuals who do the actual work (analyse, design, develop, test, technical communication, document, etc.). Development teams are cross-functional, with all of the skills as a team necessary to create a product increment. The development team in scrum is self-organizing, even though there may be some level of interface with project management offices (PMOs).

**Scrum master**

Scrum is facilitated by a scrum master, who is accountable for removing impediments to the ability of the team to deliver the product goals and deliverables. The scrum master is not a traditional team lead or project manager, but acts as a buffer between the team and any distracting influences. The scrum master ensures that the scrum process is used as intended. The scrum master helps ensure the team follows the agreed scrum processes, often facilitates key sessions, and encourages the team to improve.

The core responsibilities of a scrum master include (but are not limited to):
- Helping the product owner maintain the product backlog in a way that ensures the needed work is well understood so the team can continually make forward progress
- Helping the team to determine the definition of done for the product, with input from key stakeholders
- Coaching the team, within the scrum principles, in order to deliver high-quality features for its product
- Promoting self-organization within the team
- Helping the scrum team to avoid or remove impediments to its progress, whether internal or external to the team
- Facilitating team events to ensure regular progress
- Educating key stakeholders in the product on scrum principles

One of the ways the scrum master role differs from a project manager is that the latter may have people management responsibilities and the scrum master does not. Scrum does not formally recognise the role of project manager, as traditional command and control tendencies would cause difficulties.

A sprint (or iteration) is the basic unit of development in scrum. The sprint is a timeboxed effort; that is, it is restricted to a specific duration. The duration is fixed in advance for each sprint and is normally between one week and one month, with two weeks being the most common.

Each sprint starts with a sprint planning event that aims to define a sprint backlog, identify the work for the sprint, and make an estimated commitment for the sprint goal. Each sprint ends with a sprint review and sprint retrospective, that reviews progress to show to stakeholders and identify lessons and improvements for the next sprints.

Scrum emphasizes working product at the end of the sprint that is really done. In the case of software, this likely includes that the software has been integrated, fully tested, end-user documented, and is potentially shippable.
5. PROS AND CONS FOR SYSTEM DESIGN AND IMPLEMENTATION

5.1. Pros of systems design

The system design/modelling is a tool for managing the relationship and interaction of component across implementation phases. It allows the illustration of formal operations and the analysis of transaction as it produces results.

Most recognized benefit of using formal design is to reduce technology and cost, and allows the understanding of the system by the management and the frontline users (Levin, 2015).

According to Javed Nehal these two advantages are considered advantages “Reduced time and costs: Prototyping can improve the quality of requirements and specifications provided to developers and Improved the user involvement. Prototyping requires user involvement and allows them to see and interact more complete feedback and specifications”. The changes in a system may be necessary anytime and cost follows to implement updates. But with a good system design, it will have a follow through changes that is less or negligible cost (Nehal, 2009).

Benefits for project owner

- openness to change using technology;
- provision for design option and operation that reduce cost;
- understand the impact of operational scenarios on their own operations and grid obligations;
- minimize technology risks and accelerate adoption of commercial-scale system;
- quality and quantify trade-offs with other stakeholders; and
- analyze investment options and other risk for the system.

Benefits for vendors

- to develop and optimize their own processes using tailor-made design models;
- to incorporate experimental data into models using simulation technology;
- to analyze operation in conjunction with other elements of the proposed; and
• as a system design tool, to demonstrate capabilities to prospective clients.

Benefits for equipment suppliers
• to test their hardware under performance effectiveness
• as a tool to balance system resource and system efficiency

5.2. Disadvantages of no appropriate system models

In any system to be developed, there is a requirement for orderliness or some method of organization. In the absence of these might leads to a cycle of solutions with no directions. The simple things might be more complex as the business progresses. At some point, it might not be fitted to the expected outcomes. The following are more detailed discussion as observed by Javed Nehal:

Insufficient analysis: The focus on a limited prototype can distract developers from properly analyzing the complete project. This can lead to overlooking better solutions, preparation of incomplete specifications or the conversion of limited prototypes into poorly engineered final projects that are hard to maintain. To some extent, developers with poor system analysis will make the system more complicated.

User confusion of prototype and finished system: Users can begin to think that a prototype, intended to be thrown away, is actually a final system that merely needs to be finished or polished. The user is a very important factor in the design phase since he might be the source and user of the system.

Developer misunderstanding of user objectives: Developers may assume that users share their objectives (e.g. to deliver core functionality on time and within budget), without understanding wider commercial issues. Minimizing
management and analyst relationship in the model introduces inconsistencies in the process.

**Developer attachment to prototype:** Developers can also become attached to prototypes they have spent a great deal of effort producing. This can lead to problems like attempting to convert a limited prototype into a final system when it does not have an appropriate underlying architecture.

**Excessive development time of the prototype:** A key property to prototyping is the fact that it is supposed to be done quickly. If the developers lose sight of this fact, they very well may try to develop a prototype that is too complex. When the prototype is thrown away the precisely developed requirements that it provides may not yield a sufficient increase in productivity to make up for the time spent developing the prototype. Users can become stuck in debates over details of the prototype, holding up the development team and delaying the final product.

**Expense of implementing prototyping:** The start up costs for building a development team focused on prototyping may be high. Many companies have development methodologies in place, and changing them can mean retraining, retooling, or both. Many business companies tend to just jump into the prototyping without bothering to retrain their workers as much as they should.

You might ask yourself “how agile are you”. If you check and review the Web you might be amazed how “agile” developers are. This means that many software developers are doing agile in most of their business application. In the case of Scrum you may have freedom to adopt your own style in the manner of following the basics.

The following are some of the consolidated best practices:\[14\]

1. **Progress review iteratively**
   - Critical risks are determined before making enormous investments
   - Early iterations enable quick user feedback
   - Testing and updates are consistent with the phases of the method
   - Progress milestones provide short term attention

2. **Monitor Requirements**
   - Requirements may change, so expect revisions along the way
   - User’s own understanding should be reviewed to conform with the design
   - Settle agreement with the user on how the system works
   - Validate forward and backward requirements

3. **Use Modular Based Architecture**
   - Using modular permits reuse
   - Choice of thousands of free/commercially plug-ins
   - Improved system maintenance and extent of application
   - Promotes team work and mutual directions among developers
4. Visually Graphical Model Software

- Visual modeling improves our ability to understand software complexity
- Capture the design structure and behavior of components
- Hide or expose details as appropriate for the process
- Promote consistent communication visuals

5. Verify the Quality of Software

- What is quality? - The characteristic of producing a product or services which meets or exceeds agreed requirements by some rules.
- Software problems are 100 to 1000 times more expensive to enhance and update changes.
- Manage modular runs for each iteration and test for -
  - Functionality
  - Reliability
  - Performance

6. Control Changes to Software

- Without explicit control parallel development may result to confusion or inconsistencies
- Decompose the system into subsystems/segments and assign responsibility of each segment to a team. Establish uniform workspaces for each team i.e. each team is isolated from changes made in other workspaces.
- Establish an enforceable change control mechanism where
  - Change requests are prioritized
  - Impact of the change request is assessed
  - Plan put in place to introduce change in a particular iteration
7. CONCLUSION

A system designer utilizes the information he/she can get from the business system requirements. A change in the organization of data and the delivery of services can be done in parallel. This means that if the company has an orderly data management, it follows that the services might improve. The way to formalize some steps and the use of technology will reduce problems along some areas in the organization.

New systems design and finish product may be evident, but that is only the beginning. This means that a proper methodology is encourage to meet challenging requirements in the future. It is expected that changes is necessary soon as the company evolve and the demand of efficiency is necessary.

The work of an analysts in the systems design is very important in the development process that include [15]

- Consulting with management and users to determine the needs of the system
- Designing a system to meet the business goals
- Specifying inputs and formatting outputs to meet users' needs
- Using techniques such as sampling, model building and structured analysis, along with accounting principles, to ensure the solution is efficient, cost-effective and financially feasible
- Developing specifications, diagrams and flowcharts for programmers to follow
- Overseeing implementation, coordinating tests and observing initiation of the system to validate performance

Being a business analyst is not a requirement for the system designer, however bridging the gap of this two entities will help the system design to be more efficient. Their main focus is identifying opportunities for improving a business’s processes and using technology to eliminate problems that affect productivity, output, distribution and ultimately, the outcomes.
These professionals require a high degree of specialized skills in order to solve business problems through a variety of typical job duties, including:

- Analyzing the business processes in an organization for inefficiencies.
- Making recommendations for solutions or improvements that can be accomplished through new technology or alternative uses of existing technology.
- Acting as liaison between business stakeholders, such as management, customers or end users, and the software development or information technology team.
- Analyzing and communicating stakeholder needs by translating business requirements into software requirements.
- Documenting and evaluating required data and information.
- Using modeling, testing and data models to improve the flow of information through an organization to enhance project success.

**Skills necessary for both Business Analysts and Systems Designers**

Strong problem-solving and analytical skills, communication and interpersonal skills, and the ability to focus with close attention to detail are required in both the business analyst[16] and systems analyst professions. A business analyst needs a broad base of business knowledge and sharply honed essential skills, while the systems analyst’s skill set is more technology-specific.

While there are some common skills and knowledge requirements between systems analysts and business analysts, the business analyst profession requires an entirely different set of core specialty skills involving eliciting, analyzing, communicating, testing and verifying requirements, plus the ability to identify opportunities to solve business problems and improve processes. Business analysts
are functional experts who work for change and improvement, helping organizations reach their strategic goals through continual, successful technology improvements.
References


"What is Agile Software Development?". Agile Alliance. 8 June 2013. Retrieved: 2016-04-16


Boehm B. "A Spiral Model of Software Development and Enhancement", IEEE Computer


Gerber, Aurora et.al. (2007). "Implications of Rapid Development Methodologies"

Armstrong, Deborah J. (February 2006). "The Quarks of Object-Oriented Development". Retrieved 2016-06-08


