A Framework for Agile Methodologies for Development of Bioinformatics

SYED Ahsan, Abad SHAH

R & D Center of Computer Science
University of Engineering and Technology
Lahore, Pakistan
Corresponding author: Syed Ahsan
Email: ahsancs@hotmail.com; abad_shah@hotmail.com
Tel.: 92-322-4379469

Abstract: The realm of Bioinformatics has entered into the post genomic era where the emphasis is not on collection of the data but it has shifted to analyzing the collected data meaningfully through computational and analysis tools specifically developed for Biologist and Scientists. Conventional software engineering frameworks and methodologies are unsuitable for the development of biological applications due to their peculiar characteristics. The agile methods/methodologies have been favored by the software engineering practitioners for developing applications of the evolutionary domain such as Bioinformatics. These methods suffer the shortcomings of non-iterative analysis process and also the temporal information of the development process is not recorded. In this paper, we identify a set of characteristics which are used in identifying the domains in which the agile practices should be adapted. We also propose a framework that can be used to develop software development methodologies for developing agile software systems/applications. [The Journal of American Science. 2008;4(1):20-32]. (ISSN: 1545-1003).

Keywords: Evolutionary systems, agile methodologies, bioinformatics, temporal history, Prototype based methodology, software engineering

1 Introduction

The explosion of bioinformatics as an emerging field during the last few years has brought both opportunities and challenges for the researchers in the areas such as computer science, biomedicine, agriculture, pharmaceutical industry etc. A huge volume of genomic data is being produced at the exponential rate by the corresponding accelerated experimentation and analysis. These novel types of data are originated from a variety of experimental techniques among which many are further capable of producing data at the different levels such as entire cells, organs, organisms, and even populations. The main driving force behind this expansion of the bioinformatics domain is emergence of new, efficient experimental techniques primarily deoxyribonucleic acid (DNA) sequencing that has led to an exponential growth of linear descriptions of protein, DNA and ribonucleic acid (RNA) molecules (Alberts, B., 2002). Other new data producing techniques work as massively parallel versions of the traditional experimental methodologies. Consequently, the computational support in the experiment design, processing of the results, and their analysis and interpretations has become important (Bornberg-Bauer, E. & Paton, N.W., 2002).

The developments and advancements in the bioinformatics domain have extended scope of the domain and the emphasis has progressively switched from the data acquisition and accumulation to its interpretation. The interpretation of this huge volume and diversified data depend on our ability to develop new methods/methodologies for the development sophisticated software that can handle and analyze this new type of data.

Currently available software tools and techniques are not meeting the requirements of the researchers and not providing much help in the research. The main reasons of their inability are: i) they were not develop using the principles of software engineering, ii) there is no formal framework and software development methodologies for this new domain of applications.

We identify those characteristics of the biological data and information which make it different and difficult to handle from the conventional (or non-biological) data. Consequently, these characteristics also makes different and difficult to develop software for the handling and maintaining of this type of data. In biology, the data and its inter-relationships have a profound effect on the system of which they are part of therefore it is important and useful to identify the characteristics of both data and the information of their

relationships in a biological system (Bornberg-Bauer, E. & Paton, N.W., 2002). The characteristics are listed and described as follows.

(i) Biological data and its information are highly evolutionary, uncertain and incomplete. The latest research invalidates the established facts as they are completely changed or modified (Ostell, JM., S.J. Wheelan & J.A. Kans. 2001).

(ii) This is unprecedented type of data (Marijke Keet., 2003). For example, a molecule, such as a bacteriocin, can be coded 'mostly' on plasmids and transposons, though 'rarely' on chromosomal DNA, plus a transposon can insert itself into a plasmid: should one classify the gene location as transposon or plasmid, or both.

(iii) Depending on the environment, the behavior of an object/entity of a biological system can vary (Alberts, B., 2002).

(iv) Both functional requirements and information about the data of bioinformatics need to be studied and analyzed simultaneously because they are closely related to each other and interdependent (Shah, A, 2000).

(v) Same data item (biological object) can have different structure in the different environments (Baldi, Brunak, 2004).

(vi) Data in bioinformatics is semi-structured (Alberts, B., 2002).

(vii) Even accurately studied and analyzed a biological system can become erroneous and liable to be discarded due to data curation, interpretations, tinkering and experimentation at some later stage (Baldi, Brunak, 2004).

(viii) A biological data object can increase its size in the incremental fashion with the availability of new information about the object (Marijke Keet., 2003).

(ix) The usage of bioinformatics is very broad, with different perceptions and objectives.

(x) Determining the functional specifications is a difficult task because they are always imprecise and incomplete. For example, some genes may survive in a particular environment and useless in a laboratory environment (Ostell, JM., S.J. Wheelan, & J.A. Kans. 2001).

(xi) Biological data is explorative and iterative in nature as it depends upon scientific inquiry.

The above mentioned characteristics of this domain especially the characteristics of bioinformatics make both the existing software development frameworks such as the Waterfall and Spiral life-cycle models, and the software development methodologies such as Structured Analysis and Design, Structured Analysis and Design Technique (SADT), Jackson Structured Design (JSD), Fusion, Object Modeling Technique (OMT), Object-Oriented Design Methodology (OODM) (Embely D., Jackson R., & Woodfield, S, 1995; Shah, A., 2001; Fatouhi, F., Shah, A., Ahmed, I., & Grosky,1994; Shah, A., 2001) unsuitable to use for the development of applications of this domain. Main reasons for their unsuitability are the unprecedented functional requirements and type of data of this application domain. Moreover, the existing agile software development methodologies are not supported by any framework which gives a basis to the methodologies. In this paper, we address this problem and propose a framework for a new class of software development methodologies. This proposed framework can be used a basis in the development of the new class of software development methodologies for the new domain applications including bioinformatics.

The remainder of the paper is organized as follows. In Section 2, we give the related work. In section 3, we discuss how the Prototype based framework as proposed by Shah(Shah, 2001) provides the necessary foundations for our proposed framework. The proposed framework is presented in Section 4. In Section 5, we give concluding remarks and future directions of this work.

2. Related Work

In this section, we present review of the bioinformatics discipline and describe the problems and challenges of this area. We also present an analytical review of software development life-cycle models.

2.1 Bioinformatics

Information science has been applied to biological sciences for producing the field called Bioinformatics. It is the application of computer technology to the management of biological information. Computers are used to gather, store, analyze and integrate biological and genetic information which can then be applied to gene-based drug discovery and development. The need for bioinformatics capabilities

has been precipitated by the explosion of publicly available genomic information resulting from the Human Genome Project (Altschul S.F., et al 1997).

The simplest tasks used in bioinformatics concern the creation and maintenance of databases of biological information. Nucleic acid sequences (and the protein sequences derived from them) comprise the majority of such databases. While the storage and organization of millions of nucleotides is far from trivial, designing a database and developing an interface where by researchers can both access existing information and submit new entries is only the beginning. The most pressing tasks in bioinformatics involve the analysis of sequence information. Computational Biology is the name given to this process.

The greatest achievement of bioinformatics methods, the Human Genome Project, is complete now. Because of this the nature and priorities of bioinformatics research and applications are changing (Ostell, JM., S.J. Wheelan, J.A. Kans., 2001). We are now well in to "post genomic era" era. This has affected bioinformatics in several ways:

i) Now that we possess multiple whole genomes we can look for differences and similarities between all the genes of multiple species. From such studies we can draw particular conclusions about species and general ones about evolution. This kind of science is often referred to as comparative genomics.

ii) There are now technologies designed to measure the relative number of copies of a genetic message (levels of gene expression) at different stages in development or disease or in different tissues. Such technologies, such as DNA micro arrays have grown in importance.

iii) Other, more direct, large-scale ways of identifying gene functions and associations (for example yeast two-hybrid methods) have grown in significance and with them the accompanying bioinformatics of functional genomics.

iv) There is a general shift in emphasis (of sequence analysis especially) from genes themselves to gene products. This has lead to:

a) Attempts to catalogue the activities and characterize interactions between all gene products (in humans): proteomics.

b) Attempts to crystallize and or predict the structures of all proteins (in humans): structural genomics.

v) What some people refer to as research or medical informatics, the management of all biomedical experimental data associated with particular molecules or patients---from mass spectroscopy, to in vitro assays to clinical side-effects---has moved from the concern of those working in drug company and hospital I.T. (information technology) into the mainstream of cell and molecular biology and migrate from the commercial and clinical to academic sectors (H. V. Jagadesh., 2004).

Hence, the importance of developing and devising new methods and approaches in Computer Science specially in the areas related to Application Development, such as Software Engineering and Systems Analysis and Design, specifically tailored for Bioinformatics in post genomic era has grown.

2.2 Class-based and Prototype-based techniques.

The two object modeling techniques which are called class-based and prototype-based techniques form the basis of two types of software development methodologies, which are referred to as the *class-based methodologies* and *prototype-based methodologies*. These two types of methodologies mainly differ from each other because they use the two different object-modeling techniques. We list the main features of the prototype-based technique and methodologies to magnify the differences between these two types of object modeling techniques and methodologies. Note that further details of these features can be seen in (Shah, A., 2001).

(i) The prototype-based methodologies are more implementation-oriented than the class-based, because their emphasis is more on design and run-time (or dynamic) aspects of objects of a system.

(ii) The prototype-based methodologies do not organize objects of an application into a hierarchical structure as a class-lattice; rather they place objects of a system at the same level.

(iii) The knowledge-sharing pattern among the objects of the system is fixed at the run-time.

(iv) Objects of a system which are developed using some prototype-based methodology, pass only once through the analysis process (or phase) in their life-span when the system is developed for the first time.

(v) After the system development, each update to the objects of the developed system is only processed by the design and implementation phases. It means that objects of a system are

processed only once by the analysis phase and many times by the design and implementation phases of a prototype-based methodology

These above listed features will provide a basis to our proposed framework for the agile software development.

2.3 Evolutionary Applications Domain

More recently, agile software development methodologies have generated a lot of interest in software engineering community owning to their purported suitability for evolutionary, iterative and volatile domains such as bioinformatics and web based applications.

However, it is no necessary that a certain agile method suits all settings or individuals. In their comprehensive comparison of agile methodologies (Abrahamssons, P., Salo, O., Ronkainen, J., Warsta., J., 2002), it is pointed out that little emphasis has been placed on analyzing for which situations agile methodologies are more suitable than others. Contrary to their findings, a new domain of applications is identified in (Shah, A. 2001) by specifying characteristics of the domain for which agile development methodologies are more suitable. The applications such as Computer-Aided Construction (CAC), and the Web-based applications belong to this class of applications or domain. One typical characteristic of the objects of this class of applications is that they frequently change their structure (instance-variables and methods), state (or data values), or both.

For modeling the objects of this domain that is identified above, it has been suggested that the prototype-based object modeling technique is more suitable than the class-based object modeling technique (for more details and justifications see (Shah, A., 2001; Fotouhi, F., Shah, A. et al, 1994; Borning, A.H., 1986; Lieberman, L. 1986). Here, we list the main characteristics of the application domain as identified by (Shah, A. 2001). More details can be seen in (Shah, A. 2001). We also add a few more characteristics which are endemic to Bioinformatics, thus broadening the scope of the applications conforming to these characteristics. .

i) Applications without hierarchical structure
ii) Rapid prototype development
iii) Incremental growth of objects
iv) Desirability to trace back changes to a specific object
v) Where the grouping of objects is not important
vi) Simultaneous capturing of changes to both parameters (i.e., structure and state) of objects
vii) Polymorphic behavior of objects.
viii) Evolutionary Behavior of the Objects.
ix) Vague Functional Characteristics(This results from imprecise and incomplete data and information about the application domain)
x) The requirements are emergent i.e., the activity of developing, delivering and using the software itself yields more requirements thorough better understanding of the problem.

These characteristics can form the basis of identifying the situations where Agile software development approaches are more suited as compared to traditional approaches.

It has been concluded in (Shah, A. 2001) that the traditional and class-based object-oriented methodologies are unsuitable to use for the development of this new application domain because they lack in the capturing and modeling the characteristics of the application domain. Most of the characteristics listed in Section 1.2.1 and those listed above are common. These common characteristics suggest that bioinformatics also belong to same class of applications for which prototype-based technique has been suggested by (Shah, A. 2001).

2.5 Limitations of Existing Agile Methodologies

In our opinion, existing agile software development methodologies, though used for the development of computer applications relating to evolutionary domain, have their limitations because as mentioned above, they are not supported by a suitable frame work and also owing to the following characteristics of the evolutionary domain.

i) Explorative and iterative nature as mentioned in section 1.1.
ii) Difficulty in specifying functional requirements (Refer section 1.1)

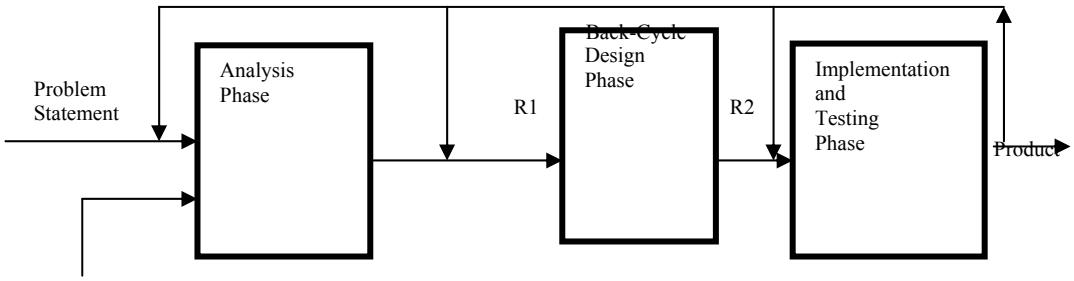
iii) Emergent requirements (Refer section 2.3).

The above characteristics necessitate an iterative analysis process .Moreover as mentioned in section 1.1 and section 2.3, the desirability to trace back changes and recording the historical information of a system adds a temporal aspect to software development. As information in evolutionary domains evolve overtime, it is not desirable to discard the old information. For example, in scientific experiments and research pertaining to some specific project, different alternative methodologies and experimental procedures may be tried to reach a specific goal and the best one selected later This may also be done to come up with some eclectic method by taking in to account many individual experiments and their mutual trade offs (Shah, A. 2001). For this reason it is important to associate temporal information with all the phases of an agile development methodology.

To the best of our knowledge, the existing Agile software development methodologies such as XP, Scrum, Crystal methodologies, Feature Driven Development and The Rational Unified Process do not have an iterative analysis process nor a mechanism to store temporal information of a system. Our proposed framework which we develop and present in next two sections have both an iterative analysis process and a mechanism for storing temporal information.

3. Towards Modification of Classical Water-Fall Model

The classical software development life-cycle Water-Fall model and its main development phases are shown in Figure 1. Note that in the figure, the maintenance phase is not shown. The function-oriented and class-based methodologies mainly follow the general guidelines of the classical water-Fall life-cycle model except for the following difference. The principle of *aggregation* in the *function-oriented methodologies* groups together functions of a system that are constituents of a higher level function implementation. In other words, the main emphasis of the function-oriented methodologies is on the system functionality, and the methodologies are referred to as the *function-oriented methodologies*. In the class-based (object-oriented) methodologies the principle of aggregation groups together functions (or methods) that operate on the same set of data. The main emphasis in the class-based methodologies is on designing object classes of a system and their organization (Wirfs-Brock, R., Wilkerson, B. & Wiener, L, 1991). Due to emphasis on different aspects of problem statement of a system in these two types of methodologies, the class-based methodologies spend more time and effort in the design phase than the analysis phase as compared to the function-oriented methodologies (for details see Wirfs-Brock, R., Wilkerson, B. and Wiener, L ,1991). In other words, the difference in the object-modeling techniques of the function-oriented methodologies and the class-based methodologies has affected the processing of the two phases (the analysis phase and design phase) of the classical life-cycle model. The difference is accommodated through a modification which recommends that for the class-based methodologies the design phase of the classical life-cycle model spend more time and effort than its analysis phase. We conclude that the difference between two classes of methodologies may pursue modifications to the classical life-cycle model to accommodate the difference. Note that in Figure 1, an *additional knowledge* represents an update to an already developed system.



Additional Knowledge
(or New System Requirements and/or updates to the existing objects)

Legend:
R1: Analysis Report - output of Analysis Phase
R2: Design Report - output of Design Phase
Product: A Developed System

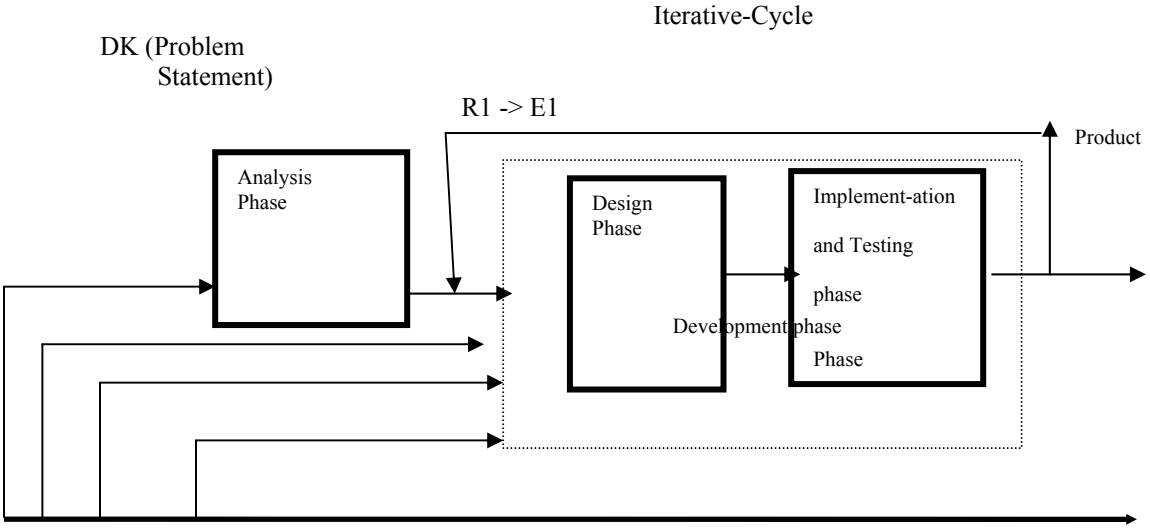
Figure 1: A common representation of classical Water-Fall life-cycle model

The weaknesses and deficiencies which further influence both the class-based and function-oriented methodologies as identified in (Shah, A., 2001) are as follows:

(i) The phases of the model work sequentially in both cases: the development of a new system, and to update an already developed system with additional knowledge. Referred to as *Static Order*.

(ii) During the development of a system, the only meta-data knowledge (i.e., functional requirements) of the system is considered and acquired. The data knowledge (data instances) of the system is neither considered nor acquired due to absence of guide-lines from the classical Water-Fall model.

(iii) The classical Water-Fall model does not provide explicit guidelines for the development of new and different types of application domains such as web-based applications and hypermedia systems (Nestorov, S., & Abiteroul, S., and Motwan, R., 1998). In these application domains, both data knowledge and meta-data knowledge are available before system development.



Legend:
DK: Default Knowledge (problem statement) of system at time t_0
AK_1: Additional Knowledge of system available at time t_1
AK_2: Additional Knowledge of system available at time t_2 **. . .**
AK_n: Additional Knowledge of system available at time t_n
R1, R2 and Product: hold the same meaning as they hold in Figure 1

Figure 2: The modified classical life-cycle model

These weaknesses and deficiencies in the classical Water-fall model and the characteristics of the new type of applications as mentioned in section 2.4 and identified by Shah (Shah,A.,2001) (i.e., web-based applications, hypermedia systems), led to the modifications in the classical Water-Fall model (Shah,A.,2001).

Figure 2 shows overall structure of the modified version of the classical Water Fall model, referred to as the *modified Water-Fall model* (Shah,A,.2001*)*. For the first case, when a new system is going to be developed from scratch both the modified Water-Fall model and the classical Water-Fall model function in the same fashion. But for the second case, when an additional knowledge is being incorporated into an already developed system, then both models differ in their working (compare Figure 1 and Figure2). These two cases are shown as two different types of events in Figure 2, which occur at different time instances on *Time-Line*. The time instance t_0 on Time-Line represents the occurrence of the first type of event, when a system is developed from scratch The set of time instances $\{t_1, t_2, \ldots, t_n\}$ represents the occurrences of the

second type of event on the Time-Line, when the instances of additional knowledge $AK_1, AK_2, \ldots AK_n$, respectively update an already developed system. At these time instances, the modified in Water-Fall model works differently from the classical Water-Fall model. After having the capability of incorporation the second type of event, the modified Water-Fall model becomes capable to provide a suitable framework for the application domains, which is described earlier.

It is already mentioned that both the design phase and implementation and testing phase of the modified Water-Fall model works iteratively and closely at occurrence of each time instance when an update is incorporated in an already developed system. An iteration means the processing of the two phases. Due to this reason both of these two phases are considered as a single phase, and it is refer it to as the *development phase* (see Figure 2).

The iterative property of the development phase is shown by the *Iterative-Cycles* in Figure 2. The Back-Cycles (in Figure 1) and the Iterative-Cycles (in Figure 2) differ in their objectives and functions. An Iterative-Cycle represents the incorporating process of additional knowledge into an already developed system, whereas, a Back-Cycle represents the incorporating Process of a revision to an under-development system. The first difference between them is that a Back-Cycle represents a revision to an under-development system and an Iterative-Cycle represents an update to an already developed system. The second difference between them is that a Back-Cycle can be initialed from any phase of the classical Water-Fall model, except for Analysis phase, whereas an Iterative-Cycle can only be initiated for the development phase (see Figure 2). Note that a *product* (in Figure 2) means an already developed system, and also after completion of each iteration of the development phase.

In the next two sections we propose modifications in the working guidelines of Analysis and Development phases of this modified model to provide us the basis for proposing a new framework suitable for the application domains such as Bioinformatics.

4. The Proposed Framework

Our proposed framework is further extension of the previous framework proposed by Shah that is described in Section 3 (Shah, A., 2001). This new framework takes an evolutionary, iterative and incremental approach both during the analysis and the development phases/process. This framework has been proposed keeping in view the characteristics of the domain applications given in Section 1. This proposed framework consists of three (3) phases, i.e., analysis phase design phase and implementation and testing phase. The design phase and implementation and testing phase are considered as a single phase which we have referred to as the development phase in section 3(Also see Fig. 2). The framework holds an *iterative* property which means that a system may be processed by the three phases more than once in its life-span.

Main feature of the agile systems is that they are inherently *adaptive*. Usually, functional requirements of an agile system is breakable into small *pieces/increments* that can be developed in pieces over periods of time, or iterations when each increment is available as proposed by Shah in (Shah A.,2001) and shown in Figure 2. Each iteration which is named as *Iterative Cycle*/*Back Cycle* in the figure can trigger analysis, design, or coding, and testing phase. The following types of changes can occur to a system.

Type I: If a change causes no loss or addition of information in an existing analysis or design artifact, then we can start the processing of the next phase.

Type II: This type of change causes no loss or addition of information in the analysis artifact but the design artifact changes due to this change. In this case, we can start the processing from Design Phase

Type III: If a change occurs in functional requirements of a system, then the next iteration starts from Analysis Phase.

Type I and Type II changes pertain to the development phase and will be discussed in Section 4.2. Type III change invokes the analysis phase and is discussed in detail in the next section.

We add the temporal component with all three (3) phases of the proposed framework (see Figure 2). By attaching time dimension with the analysis report, the design report and the resulting prototype system, we can maintain the history of changes in the development of the system. Each report and the resulting prototype system is identified by a tuple of the form $(T_s, Tc, A/D/P, T sc)$, where *Ts* and *Tc* represent start time and current time, respectively, and *A/D/P* represents analysis, design or prototype artifact. The time interval *(Ts, Tc) that is* denoted by *Tsc*, is referred to as the *time span* of the artifact. A pair of time-span and an artifact identity reduces the search space for a particular prototype associated with a particular design/analysis of a artifact. This might be used to establish the validity of an artifact during a

particular time span. By examining a sequence of tuples associated with each underdevelopment system, we can also measure the degree of evolution of the system.

The working of the proposed framework is described in the next sections.

4.1 Analysis Phase

The functional requirements of a system are gathered in the same manner as in the classical life-cycle model for the first iteration, or in the case the system is being developed from the scratch. As it has been mentioned in Section 3 and in Section 4 above, we modify Shah's framework (Shah, A., 2001) to provide a rationale for developing methodologies for the new application domain including bioinformatics. The main difference between Shah's framework and this framework is in the analysis phase. In this framework, the analysis phase is iterative.

There are some preliminary processing activities of the phase include comprehensive study of a problem statement, and identification and analysis of the system's functional requirements. The first iteration of this phase can be imitated after completing the preliminarily processing activities. This phase suggests to pick-up an initial list of prototypes referred to as candidate prototypes (or objects) of the system as mentioned in (Shah, A., 2001). This list of the candidate prototypes, can be prepared using similar criteria that are used by the class-based methodologies such as OMT (Pressman, R., 1992, Shah, A., Fotouhi, F., Grosky, W.,. Al-Dehlan, A., and Vashishta, A, 1993). The criterion that is commonly used by the class-based methodologies suggests for picking the *nouns* from a problem statement and considers them as candidate prototypes. After preparing this list, it is further processed, which includes prioritizing and discarding redundant and vague prototypes from the list of candidate prototypes, which might be included in the list during the subsequent iterations, and merging identical and closely related prototypes. This processing of the candidate list gives a *final list of prototypes/objects*. The output of the analysis phase is an Analysis Report/Requirement Document that is referred to as Requirements here, which is denoted as *R1* in Figure 4, and it mainly contains the final list of prototypes and other necessary information about the system such as relationships among the prototypes. This Analysis Report/Requirement Document is prioritized and added to a priority stack Figure 3).

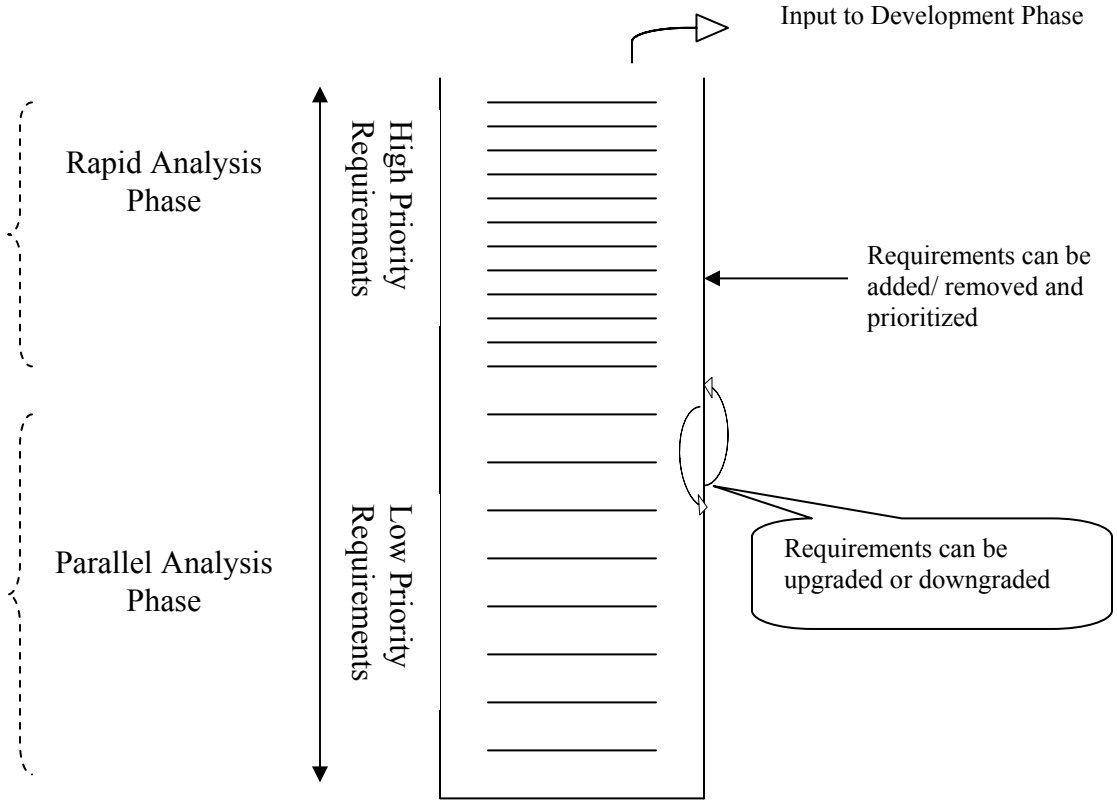
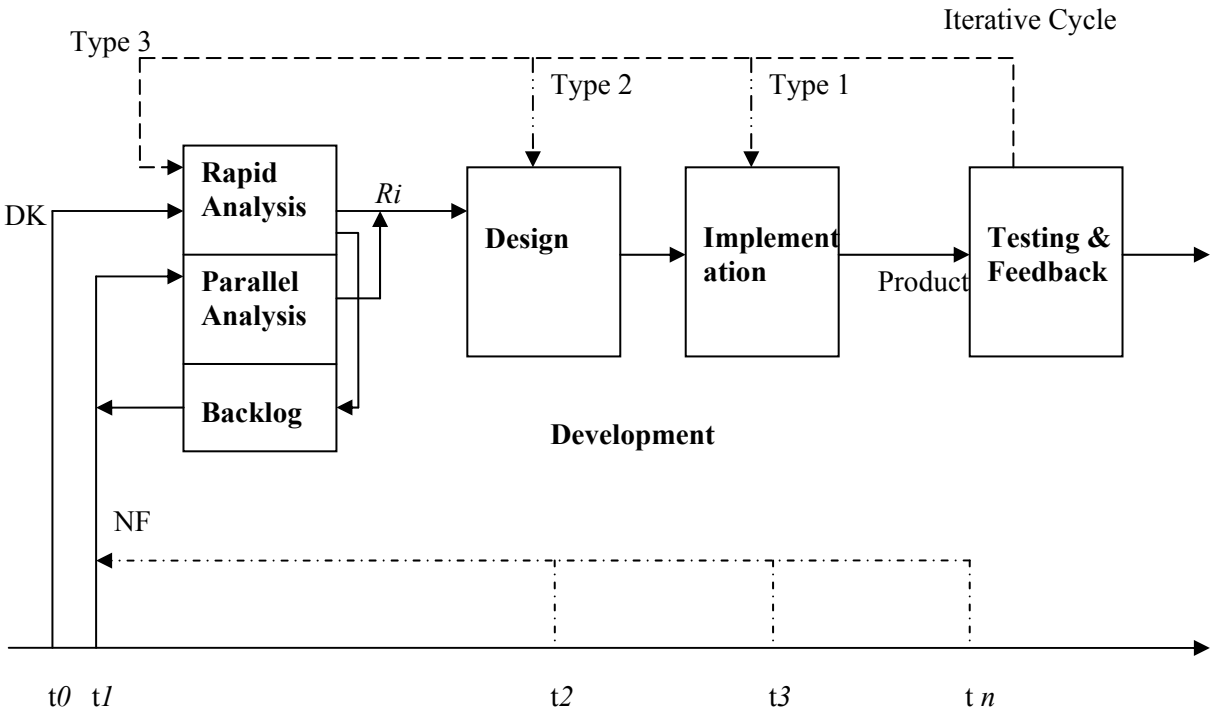


Fig 3: Priority Stack



Legend:

DK: Default Knowledge (problem statement) of system at time t_0
AK_1: Additional Knowledge of system available at time t_1
AK_2: Additional Knowledge of system available at time t_2 . . .
AK_n: Additional Knowledge of system available at time t_n
NF: New Facts about system discovered over time
R1 and Product: hold the same meaning as they hold in Figure 2
Ri: Analysis report(s) at later stages of development at any instant i

Figure 4: The modified Prototype based model for Bioinformatics domain

In the development of some systems like Bioinformatics may pass through this phase many times due to their typical characteristics which have been described earlier. Also, the feedback to this phase can come either from development phase through product/prototype testing or the feed back may be from Analysis Phase itself as new facts become known about the system. The working of Development Phase is described in the next section.

This proposed feedback mechanism makes Analysis Phase highly iterative, and iteration may be caused by the incorporation of some rapid changes occurred to a system. If the change affects the functional requirements, it is referred to as Type III change. In this case the next iteration starts from

Analysis Phase (see Figure 4). To incorporate this type of change/iterative, we suggest modification in the framework which has been proposed by Shah in (Shah A., 2001) and also given in Figure 2. The modified framework is shown in Figure 4. New information (or changes) occur to a system is incorporated in the system, and we refer it to as *New Facts* (NF) and is denoted by an arrow as shown in Figure 4.

The analysis phase in our proposed model is divided into two stages, i.e., Rapid Analysis Phase and Parallel Analysis Phase (see Figure 4). Rapid Analysis Phase processes those requirements which are high priority and temporally stable. Parallel Analysis Phase processes those requirements which are low priority and temporally unstable and they keep on evolving thought-out the life-span pf the system The requirements form Parallel Analysis Phase are passed to Rapid Analysis Phase as they are promoted to high priority and become temporally stable. It is the responsibility of the project/system stakeholders to provide, clarify, specify, and prioritize functional requirements. The priority can be either high or low depending on when they are input to Development Phase. Parallel Analysis Phase processes those functional requirements which evolve throughout life-span of a system until they can be upgraded to high priority as they become temporally stable. Functional requirements can appear throughout most of the project/system prioritized and added to the priority stack shown in Fig 3. In the figure, the requirements may be reprioritized at any time (upgraded/downgraded), or they may be removed from the priority stack (see Figure 3). The highest priority requirements from the top of the stack are input to Development Phase so that they can be implemented within the current iteration.

In the traditional Waterfall and Spiral lifecycle models the system stakeholders/customers specify their potential requirements early in the project, including ones they might need but really aren't sure about at that moment. They feel that it will be difficult to get them added later because of some change management/prevention process, which will be put in place once the requirements document is finalized. In our proposed framework, stakeholders elicit those requirements which they immediately need and are high on their priority list (It is unlikely that they will miss any of the high priority requirements). In conventional methods (Water Fall and Spiral) the majority of requirements efforts are performed at the beginning of the system development. For application domains such as Bioinformatics, it is unlikely that all of these requirements are still valid till the completion of the system. That is because application domains like Bioinformatics are evolutionary and dynamic as they are technology research driven. The iterative nature of analysis phase in our proposed framework suits the evolutionary domain such as bioinformatics because addition and modification of functional requirements is a continuing process.

4.2. Development Phase

Development Phase further consists of two phases, i.e., Design Phase and Implementation Phase. These two phases work closely to each other. Initially, Development Phase is triggered by the output (Analysis Report) from the Rapid Analysis Phase, R1 (see Figure 4). Once an executable prototype(product) is delivered , it goes through the iterative cycle and incrementally developed by using input both from Rapid and Parallel Analysis Phase (Rp). Development Phase can also be triggered by occurrence of the events *t1, t2, …tn*. These events are updates to an already developed system using additional recently acquired knowledge. In Figure 4, *n* number of triggers t are shown When a trigger *t* occurs, Development Phase takes both the product (an already developed system) and additional acquired knowledge as input, and processes them to incorporate the additional knowledge to the product. A continuous development or update is an essential characteristic of the agile applications and they need updates whenever requirements change or new information/knowledge about the application domain becomes known.

After the initial development of a system as we have discussed above, the system can experience different types of changes and the incorporation of these changes is a necessary and continuous task in its life-span. In Section 3.1, we have categorized three (3) changes. Now we propose their incorporation process to an agile system.

i) Type I: This type of change is the change that occurs only to the data parameter of the prototype. This type of change can easily be incorporated without affecting other parameters and to incorporate it, we do not need to do any intensive development effort. The existing implementation report can be used and the change can be incorporated directly in the implementation phase. Te existing analysis and design reports can also be consulted. In the figure, the two inputs to the implementation phase, the existing implementation report and the new information/data (change) are shown. After incorporating the change, it is necessary to update the implementation report for the future reference.

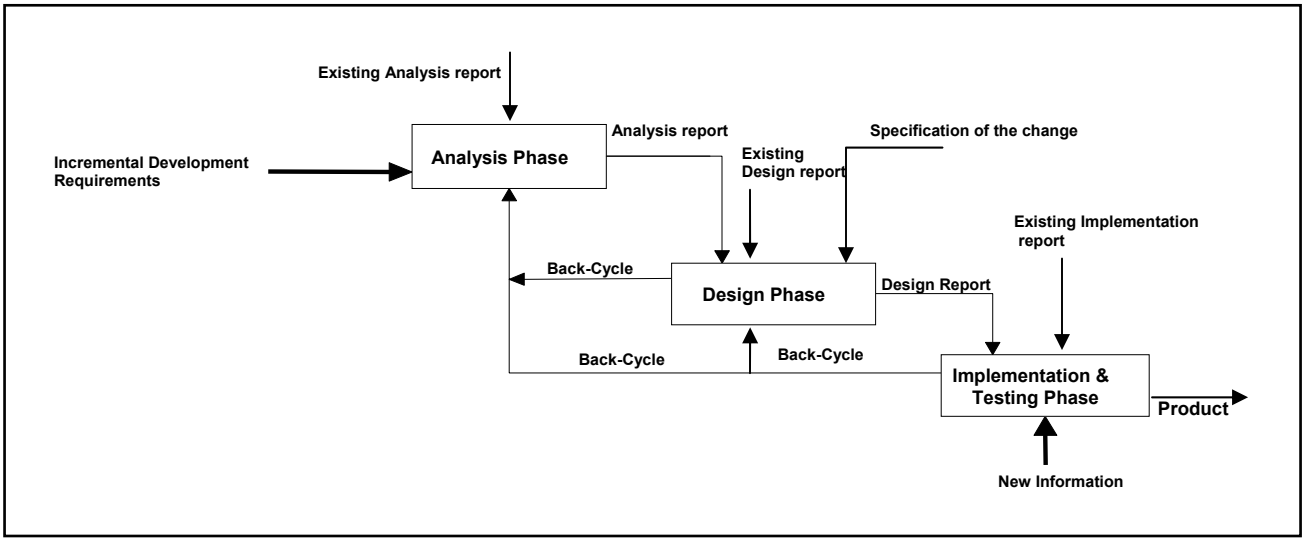


Figure 6: Development phase to incorporate Type I, Type II and Type III changes

ii) Type II change.

A change is a Type II change if it occurs only to the structure of the prototype. For incorporating this type of change, the inputs to the design phase are the existing design report and the specification of the change and the output is the new updated design report. This output is used as input to the implementation phase to physically incorporate the change in the application.

iii) Type III change.

If a change is both to the data and structure parameters, it is referred to as Type III change. A Type III change can significantly change the overall structure of an agile application. For incorporating such a change, we propose efforts in all the three phases as shown in Figure 6.In the figure the change is shown as incremental requirement development that are input to the analysis phase which results in a new analysis report for design phase and a subsequent new design report to be implemented in the implementation phase.

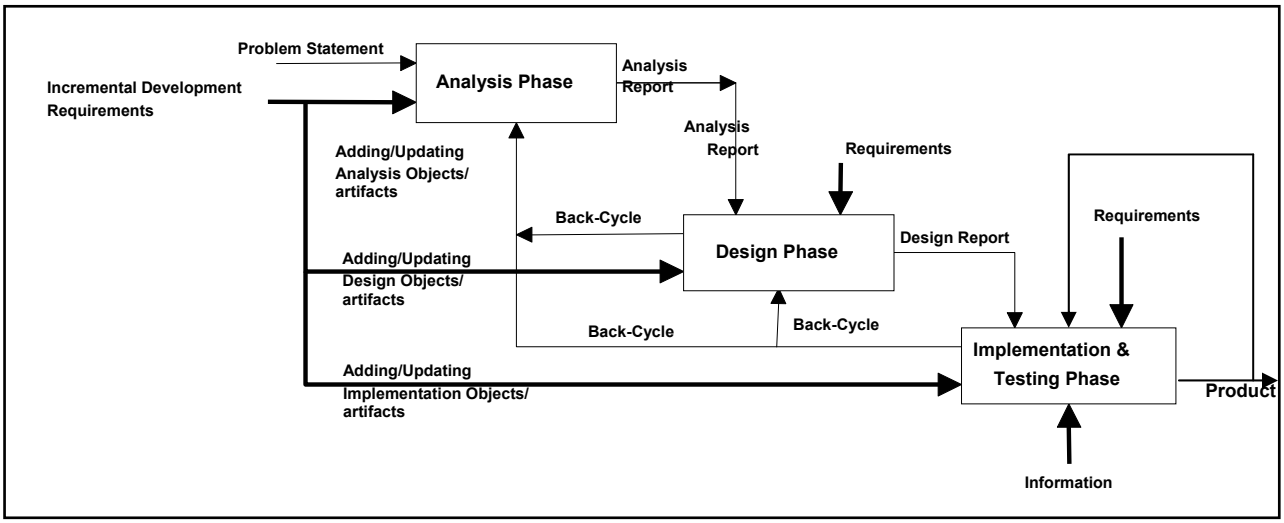


Figure 7: Detailed and Complete development phase for agile application development

Figure 7 illustrates that type 3 changes are input to the analysis phase where it results in a new analysis report which is subsequently input to the design phase to result either in the new design report or a back cycle if further evolutionary changes are reported. The new design report results in the executable prototype which is then tested. The testing phase may also output type 2 or type 3 changes which are then in out to the design or analysis phase through back cycles. The implementation and testing phase can also accommodate type 1 changes which may be reported after the testing of executable prototype or when new information about the system is known.

5 Conclusion and Future Works

The importance of computational methods for analyzing biological data has grown in the post genomic era. In this paper, we concluded that bioinformatics belong to a class of applications for which conventional models of software development are not suitable. This is due to peculiar characteristics of biological data and also because of wide breadth of biological data user base, in which there is no agreement, either vertically (within an organization) or horizontally(across various organization), on what is expected out of a biological data computational tool. This increases the complexity for a computer scientist who must come up with better tools and software engineering models to cater for this increasingly important and complex domain. Agile methods are well suited to the exploratory and iterative nature of Bioinformatics. But most of the agile methods do not prescribe a specific framework, but rater a philosophy for approaching software development. In this work we developed a framework which can form a basis of evolving a methodology for development of Bioinformatics. The frame work can provide the building blocks for a methodology to develop which can include the best practices being used in various stages of most used Agile practices such as XP, SCRUM or Crystal Methodologies.

Corresponding author:
Syed Ahsan
R & D Center of Computer Science
University of Engineering and Technology
Lahore, Pakistan
Email: ahsancs@hotmail.com
Tel.: 92-322-4379469

Contact Author:
Prof. Dr. Abad Ali Shah
Email: abadshah@uet.edu.pk, abad_shah@hotmail.com
Tel: 923004782122, 92-42-6829499, 92-42-5302222

Received: 12/16/2007

References

1. Abrahamssons, P., (2002) Commitment nets in software process improvement. Annals of Software Engineering.
2. Abrahamssons, P., Salo, O., Ronkainen, J., Warsta., J (2002). Agile Software development methods, Review and Analysis. VTT Publications.
3. Amber, S., (2002). Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. New York. John Wiley & Sons, Inc.
4. Bob Hunter, Martin Fowler, & Gregor Hohpe, accessed February 12, 2006. http://www.thoughtworks.com/us/library/agileEAIMethods.pdf
5. Alberts, B. (2002) .Molecular Biology of the Cell. Garland Science Publishers , New York
6. Bornberg-Bauer, E. and Paton, N.W., (2002),'Conceptual Data Modeling for Bioinformatics, *Briefings in Bioinformatics.*.
7. Dahl, O.J. & Nygaard, K., (1966) .SIMULA - An ALGOL-Based Simulation Language.
8. *Combinations of the ACM, 9(9),* September, , pp. 671-678

9. Embely, D.,Jackson, R., and Woodfield, S. (1995) . Object-Oriented Systems Analysis: Is It or Isn't It?. IEEE Software Journal, July, 1995.
10. .Fatouhi, F., Shah, A., Ahmed, I., and Grosky, (1994) .TOS: A Temporal Object –Oriented System. *the Journal of Database Management, Volume.5, No.4*, pp 3-14.
11. H. V. Jagadesh. (2004). Data Management for Life Sciences Research. *SIGMOD RECORD*, June.
12. Marijke Keet. (2003). Biological Data and Conceptual Modeling Methods. *the Journal of Conceptual Modeling, Issue: 29* .
13. Nestorov, S., & Abiteroul, S., and Motwan, R., (1998) .Extracting Schema from Semistructures. *ACM-SIGMOD Record, Vol. 27, No. 2*.
14. Shah, A., Fotouhi, F., and Grosky, W., (2004) .Operators of the Temporal Object-Oriented System and their Implementation. *the Journal of Information Sciences (INS),* Elsevier Publisher, the Netherlands, 158(2004) Pp 37-68.
15. Shah, A., (2001) .A Framework for Life-Cycle of the Prototype-Based Software Development Methodologies. *the Journal of King Saud University*, *Vol. 13*, Pp. 105-125,.
16. Ostell, JM., S.J. Wheelan, J.A. Kans. 2001. The NCBI Data Model in Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins 2ne Edition. John Wiley & Sons Publishing. ISBN: 0471383910 pp. 19-44.
17. Elmasri, R.A. and Navathe, S.B. 2000. Fundamentals of Database Systems 3rd Edition. Addision-Wesley Pubishing. ISBN: 0805317554
18. Genetic Sequence Data Bank April 15, 2001 NCBI-GenBank Flat File Release 123.0 Distribution Release Notes ftp://ncbi.nlm.nih.gov/genbank/gbrel.txt
19. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J.,Zhang, Z., Miller, W. and Lipman, D.J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* 25, 3389-3402, 1997.
20. Pressman, R., "Software Engineering-A Practitioner's Approach," McGraw-Hill, 1992.
21. Rumbaugh et al, "Object-Oriented Modeling and Design," Prentice Hall, Englewood Cliffs, New Jersey, 1991.
22. Shah, A., Fotouhi, F., Grosky, W.,. Al-Dehlan, A., and Vashishta, A., "A Temporal Object System For a Construction Environment," *Proceedings of the XIII Conference of the Brazilian Computer Society (SEMISH-90)*, September, 1993, pp 211-225.
23. Wirfs-Brock, R., Wilkerson, B. and Wiener, L.,, "Designing Object-Oriented Software,".Prentice Hall, Englewood Cliffs, New Jersey, 1991