ABSTRACT – One of the problems we face today for software development is how we can comply with the speed of change in the business environment. To address the issue, we present a new paradigm called Progressive Development Model that gradually releases the software components according to the current business needs. In this paper, we also present a design pattern of software architecture optimized for Progress Development Model. Under the design pattern, system functionalities are identified into two components: Core and user functions. With a standardized interface called AIL, staged user functions can be attached onto the core without breaking or deforming the overall architecture. The development model is especially suitable for the distributed software applications development that is getting its popularity recently.

Keywords - Software process, process modeling, Software architecture design, rapid development.

I. INTRODUCTION

In the new era of electronic business environment, the software development paradigms have been evolved into non-conventional, rapid scenarios of distributed developments. In the past few years, new paradigms have been proposed to accommodate the change. Some researchers were looking into shorten the development cycle with re-use technique, such as Component-Based [4], Rule-Based [3] development. Others were just to modify the current development processes, such as conventional Branch-By-Release Model [2], Branch-By-Purpose Model [13] and packaged software development [5]. In addition, some software engineering researchers have looked into, not the content of the processes, the process management and improvement (PM/I). Hope, by doing PM/I, software organization can improve their processes to meet the new business environment. CMMI [6] is the best known example of PM/I.

Nonetheless, these remedies only solve part of the problem, simple because they did not change the software development scheme appropriately to fit into the E-business behaviors. And the PM/I can be negatively impacted by human factors [8, 9]. That is, the business needs change so rapidly that the software development process cannot keep up with it.

For the problem we are studying here, one remedy is to use a new paradigm that gradually releases the software components according to the current business needs, instead of the overall system functionalities.

In this paper, we present a process model, called Progressive Development Model (PDM). In the PDM model, a business-operable core system is initially released into production for users to start their business operation, and the rest of the software components are released based on a schedule constructed upon the business needs.
This paper is organized as follows. Section 2 describes the details of the PDM. Section 3 presents the PDM release management process. Architecture design is shown in Section 4, while Section 5 performs a comparison of process models. Section 6 discusses the software applications developed under PDM model followed by an industry experience that utilized the PDM as its development process.

II. THE PROGRESSIVE DEVELOPMENT MODEL

The basic concept is to distinguish the most needed core functions of the software system from the other function modules. The business-operable core functions are released as a basis of the production system in a shortest possible development cycle. Those additional function modules are then released with a schedule based on the priority constructed upon the business needs. When business needs change, the priority changes, and the development effort can be transited into the needed function module in a timely fashion.

The process model was firstly introduced in [10]. It starts with a tight relationship between the software development team and the sponsor. In order to define the core system, the software development team needs to work closely with the sponsor to derive a concise list of business-operable core function of the software system.

Once the core system is identified, other desire function modules can be defined as subsequent drop releases according to the current view of business needs. The process flow of the PDM is shown in Figure 1.

Under PDM process model, a software project planning team will create a Technical Proposal (TP) for sponsor review. After granting an approval from the sponsor, the software project planning team should generate a Preliminary Requirements. This Preliminary Requirements will be reviewed, by both sponsor and software team, to collect basic system information from technical and business perspectives.

Once the sponsor and software team have agreed upon with the Preliminary Requirements, the software project planning team will then finalize an Operation Technical Plan (OTP) as the technical and business guidelines of the project. In the OTP, the core system and other function modules should be identified. In addition, the release sequence and schedule opportunity of the function modules should be defined in the OTP as per business views at that point. This will give a road map to the project team and provide necessary information for project planning and tracking.

The sponsor reviews the OTP and makes a “Go/No Go” decision on the scope of the OTP. At the moment when it is a “Go”, the software project is formally established. If it is a “No Go”, the review process for OTP will be repeated until a “Go” decision is reached.

Now the software team can start its normal software development process: (1) Detailed software requirements are written. (2) Software design completes for the core system. (3) Software developers write codes per the requirements. And (4) The core software system is tested.

Figure 1. The Progressive development Model
During this development cycle, the sponsor will provide inputs for any business updates. The core system development process may be iterated until the requirements are stable and baselined. Nonetheless, the core system should be released as a business-operable production system in a timely fashion, such that the sponsor can start its business operation based on the market opportunity.

The development process for function modules is much more dynamic and flexible. The requirements for the function module may be updated any time during the development phase. This update could be an incremental delta or changes to what were specified in the function module.

While the software team develops and tests a function module according to the schedule in OTP, the sponsor may request to deliver a package (bundle or drop) that contains features needed for current business opportunity. These features may be part of the current function module that are already developed, or in the future functional module(s) that need to make every effort to develop them immediately.

For such case, the development efforts will be transited into the package development cycle, and the development of the current function module is paused. The development cycle of the current function module will be resumed when the package is release and there is no urgent need for other package.

III. PLUS PROCESS FOR PDM RELEASE MANAGEMENT

The problems with the conventional release management process is the lack of efficient communication between development organization and the sponsor and is not capable of releasing partial functional module(s) as a package into production when there are business needs.

Furthermore, when a Software Configuration Management (SCM) tool system is used to track for work products, it makes conventional release management process even more difficult to manage the scope changes. Because conventional release management process will starts only at the end of development cycle, it cannot retrieve any useful information from SCM regarding what should and should not be included in an emergency package.

A process for Release Management, called PLUS [11], has been developed for the Progressive Development Model. For the release of the core system, the release management process may be not much different from a conventional process that can be found in any Software Configuration Management (SCM). On the other hands, the release management process for function modules is quite a different picture. This is due to the facts that the work products in every function module usually have local attributes, and are frequently updated. Moreover, they sometimes are moved between function modules. Detailed discussion of the PLUS process is presented in [11].

IV. ARCHITECTURE DESIGN FOR PDM

Software development organizations today are aggressively making their software applications easy for reuse and/or recycle by employing COTS components or code library. However, this makes their software applications hard to maintain because they may carry the defects from the COTS and difficult for new developers to understand the code library.

It is quite common a phenomenon that the architecture designed for a software system reaches a breaking point when so many additional application functions (modules) are added into the initial system. When the software development passes that point, its performance degrades and system flaws easily. This is because the software architecture has been deformed and some of its design advantages are no longer existed. This situation is even worse for the case of software reuse and recycles.

To make the PDM works efficiently and effectively, its software architecture must be designed properly such that staged user application functions can be easily added into the system without deforming or breaking the overall structure. It is especially an important factor when software reuse and recycle is a desired design.

Figure 2 depicts the core system process of the architecture for PDM. Core system is where the core functions resided. All users related Functional Modules are then attached onto the Core system sequentially. A standardized interface is established as the communication channel between the core system and the attached Functional Modules.

The core system consists of two components: Core Engine and AIL. The Core Engine is the main software functional module in the software system. Its functions may include fundamental design, database transactions, parameters or languages translation, Presentation (GUI), specific business rules, basic data elements, etc.
A standardized interface called Attributes Interface Layer, or AIL, sits between the Core Engine and Functional Modules. It converts application specific objects sent by the Functional Modules into core object to be processed by the Core Engine. The application specific objects can be defined locally by the Functional Modules and may not be recognized by the Core Engine. While the core objects are globally defined within the core system process, they are used for processing in Core Engine. Therefore, the AIL translates the objects according to a built-in relation between the application specific objects and the core objects. Any newly attached Functional Module has to establish the relation by adopting any existing core objects, creating new core objects, or mapping new application specific objects into existing core objects.

The Attribute Interface Layer (AIL) plays a major role in making the PDM architecture extremely flexible and expandable. It functions not only as an interface but also as a liaison between the Core Engine and the Functional Modules. Its existence adds the flexibility of implementation for functional modules while assures the data and system integrity.

Details of the architecture design can be found in [14].

V. COMPARISON OF PROCESS MODELS

Figure 3 depicts a conventional software development schedule with Features 1, 2, 3, 4 and 5 being included in Release 1, while Release 2 consisting of Features 6, 7 and 8. Along with the natures of the conventional software development, all staffs are concentrated in developing and testing the feature set of Release 1. Therefore, Release 2 will be not able to be started until Release 1 is officially passed test or deployed into production.

This prolonged life cycle of Release 1 prevents Release 2 from starting any sooner than the finish of Feature 4. Conventional release process can do nothing but to package the software when Feature 4 is completely passed testing and loading the package into production.

In the Progressive Development Model, we can re-arrange the feature set into packages with consent from the sponsor. Figure 4 shows that, by re-arranging the feature set into four packages, the sponsor not only can start its business operation earlier (with Feature 1, 3 and 5), but also will receive all of the features in a shorter schedule. This is because the staffs that are working on the first package can start the development for Package 3 and 4 right after the completion of Package 1. That is a much efficient way of resource utilization, hence, resulting in a better schedule.

The PLUS works efficiently under PDM as shown in this section. PLUS process may place Feature 2 (Package 2) and Feature 4 (package 3) simultaneously into testing environment without causing any conflict and later release them according to their schedule. The same can be applied with Feature 6 and 8 (package 3) and Feature 7 (package 4).

On of the advantages of employing PDM into software development process is that it makes software reuse and recycle extremely efficient and effective.

Nonetheless, either reuse or recycle of the existing functional modules will accelerate the development cycle of software. Again, instead of release the software at once as a complete system, the software core and functional modules will be released according to the new sponsor’s timeline to meet its business needs.

VI. SOFTWARE APPLICATIONS DEVELOPED UNDER PDM MODEL

In this Section, we will discuss a software application as an example of adopting the PDM model and show some industry results from applying the PDM model into the software development.

Here we discuss a software application for telecommunication network management that adopts the PDM as the software development model for more than 3 years.

The software application is designed for managing telecommunication networks, including inventory of network elements and designs of service circuits. The main concerns for the design of the software application are the network provisioning in each metropolitan service area (MSA) and the services that will be provided to the customer of the network service provider.
Although there are many required features to carry the operation for network provisioning, it is not necessary for every MSA to have all of them. In addition, the network elements and services in each MSA may be different. Moreover, the network service provider planned to sequentially introduce network services to its customers in MSAs. The sequence of MSAs was revealed to the software team.

Therefore, it was determined that the function modules should be defined as the feature set needed in every MSA, while the core system was defined as the GUI, database design, interfaces with other systems and other basic operations needed for the initial production.

In this case, as shown in Figure 5, by adopting the process model, it achieves productivity gain in-year by decreasing Network Trunk Engineering & Provisioning AWT (in minutes) from 295 to 231 and by increasing number of orders per day per engineer from 3.4 to 6.8.

VII. CONCLUSION

In this paper, we present a process model, called Progressive Development Model (PDM), and its associated release management process, called PLUS. In addition, we also present the software architecture for the Progressive Development Model (PDM), as a remedy for the architecture design problem of rapid development in the E-era.

The architecture of the PDM model has taken into account all the factors of today’s business environment and is considered to be an efficient and effective design. As a remedy for the problem of distributed development, the PDM model has been utilized in some large-scale software projects. It is proven to be an efficient and effective process for today’s business environment.

Overall, the PDM is different from the other process models in the senses that (1) Requirements volatility does not significantly impact on the process as it does to the other processes. (2) The process model reacts properly to the change of business environment. (3) It is easier for project management to estimate, commit and track the schedule by using the process model. (4) It makes software reuse and recycle very efficient and effective. (5) The PDM architecture is extremely flexible and expandable. The architecture will be not easily deformed by the over growing of software system due to the business demands.

REFERENCE