Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2013, Article ID 796505, 13 pages
http://dx.doi.org/10.1155/2013/796505

Research Article

A New Software Development Methodology for Clinical Trial Systems

Li-Min Liu

Department of Applied Mathematics, Chung Yuan Christian University, 200 Chung Pei Road, Chung Li 32023, Taiwan

Correspondence should be addressed to Li-Min Liu; lmliu@math.cycu.edu.tw

Received 29 October 2012; Revised 14 January 2013; Accepted 25 February 2013

Academic Editor: Pekka Abrahamsson

Clinical trials are crucial to modern healthcare industries, and information technologies have been employed to improve the quality of data collected in trials and reduce the overall cost of data processing. While developing software for clinical trials, one needs to take into account the similar patterns shared by all clinical trial software. Such patterns exist because of the unique properties of clinical trials and the rigorous regulations imposed by the government for the reasons of subject safety. Among the existing software development methodologies, none, unfortunately, was built specifically upon these properties and patterns and therefore works sufficiently well. In this paper, the process of clinical trials is reviewed, and the unique properties of clinical trial system development are explained thoroughly. Based on the properties, a new software development methodology is then proposed specifically for developing electronic clinical trial systems. A case study shows that, by adopting the proposed methodology, high-quality software products can be delivered on schedule within budget. With such high-quality software, data collection, management, and analysis can be more efficient, accurate, and inexpensive, which in turn will improve the overall quality of clinical trials.

1. Introduction

Clinical trials (CTs) are measures to evaluate the safety and efficacy of a medical device or drug. CTs are crucial to modern healthcare industry and usually take substantial resources and time to complete with four major parties involved: the sponsor, the government agency, the participating hospitals (or sites), and the patients (or subjects) [1]. The clinical trial is usually supported financially by a sponsor that may be a pharmaceutical company, a medical device company, or even a government agency. Sponsors also define the scopes, procedures, protocols, and goal of their CTs. The definition of "protocol" in clinical trials is very different from the one commonly used in the computer engineering (please refer to the US Food and Drug Administration (FDA) for more information [2]). Data are collected on sites from subjects and recorded on well-defined forms, that is, case report forms (CRFs), to guarantee the quality of the collected data. For the safety of subjects, CTs should always follow predefined protocols and government regulations.

Today, almost all fields, including the healthcare industry, are benefited by high-speed computers and information technologies [3, 4]. High-speed internet accessibility is one of the prime examples. For computerizing CTs, several concepts were introduced over the past decade, including electronic health record (eHR) [5, 6], electronic data capture (eDC), and electronic CRF (eCRF). Although eDC and eCRF are sometimes used interchangeably [7], they can be formally defined as the following:

> *"eDC: synonymously used with remote data entry; collecting data in (permanent) electronic form by systems… with or without a human interface…"*
>
> *"eCRF: data may be recorded either from source documents or the eCRF may be used as the primary source document…" [8].*

More precisely, eCRFs are considered the replacement for paper-based CRFs while eDCs focus on two things: (1) data are captured remotely and (2) the whole lifecycle of data is always in electronic forms. It means that each site must have an eHR system to build eDC [9–13]. Unfortunately, study shows that only a small portion of the trial data (less than 40%) is available in hospitals' eHR [14]. Other tasks such as CT management, workflow control, and embedding controlled terminologies, standards and regulations (UMLS, SNOMED, CDISC, HL7, and 21 CFR Part 11) into software for CTs are also investigated [4, 15–22].

In this paper, a "Clinical Trial System" (CTS) is defined as a software system to handle trial data; therefore, a CTS includes data-related software, for example, eDC or eCRF, but excludes the rest, for example, CT management or workflow control. The three core components of a CTS are the data collection module, the data exportation module, and the backend database system. A CTS, eventually, exports data for statistical analysis by statisticians with professional statistics software, which is excluded in the current definition of CTS because it requires expertise completely different from software development. From the software developers' point of view, the main complexity of developing a CTS comes from the data collection system rather than data analysis system which requires skills from statisticians.

To deliver a quality software product as complicated as a CTS on time within budget, a predefined software development methodology (SDM, or so-called process model) should be followed. However, adopting existing purpose SDMs to develop a CTS requires the development team to disregard some fundamental guidelines of the SDM because all CTS share some unique properties (described in details in the following section).

The rest of the paper is organized as follows. Section 2 illustrates the CTS/SDMs, the process and properties of CTs and discusses the differences between using generic tools and domain-specific tools to develop CTS and reviews existing SDMs for CTS development. Section 3 describes the proposed SDM in details followed by the case study and discussions in Section 4. Finally, the conclusions are presented in Section 5.

2. Development of CTSs

2.1. Background. Modern CTS works not only as a vehicle at front end for collecting data but also as a complete system which can be as complex as any commercial software. To maintain data integrity, a CTS usually applies constraints such as simple value range check and complex cross-form value derivation. For reasons like data auditing, CRFs may also include nonnumerical fields like hand-drawn pictures (e.g., the position of ulcer), photographs (e.g., photographs before/after operations), or texts (e.g., doctor's comments). In addition, complicated flow branches can be defined in the protocol (e.g., subjects with different conditions/test results may require different flow path to fill additional/different CFRs).

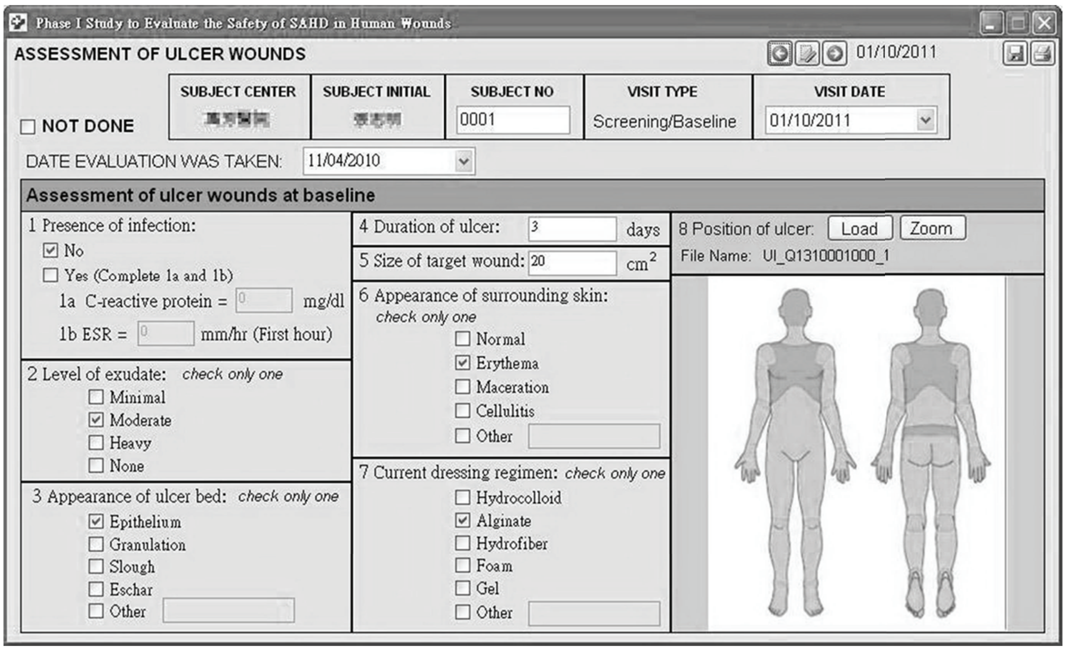
Figure 1 shows an eCRF in the data collection module of a CTS implemented with functional buttons, selection boxes, a picture with hand-drawn marks, and several implemented constraints. With such complexity, most CTSs need to be custom made on a case-by-case basis and demand substantial resources from the sponsors.

Figure 2 shows three commonly used CTS architectures (dashed box indicates the data collection system with databases). Figure 2(a) illustrates an architecture where the trial data are directly retrieved from hospital eHR by performing queries from the CTS front-end module and then saved into the backend databases. CTS front end and backend can be merged into one system as a single server if necessary. This architecture is convenient for CTs with a small number of sites with fully supported and identical eHR. Since data are not entered manually, this architecture provides the highest level of data integrity while eCRFs can serve as a data viewer. However, engineers need to develop a CTS front-end module for any site with a different eHR, which requires significant resources and cooperation from hospitals. This increases the risk of project failure. In addition, most hospitals do not have extra resources to work with every CT conducted in house.

Figure 2(b) indicates an architecture which requires a CTS user interface (UI) to collect trial data. In most cases, the CTS UI is implemented the same way as the CRFs developed in the protocol to ease the learning curve of the users. Data are manually entered from transcribed data or local eHR on sites. Since data are input manually, data integrity is lower than that in the previous architecture. On the other hand, this architecture has many advantages. For instance, the CTS UI can serve as the data viewer and be used at all sites. Today, most of the CTS UI is developed as a web-based application using a web browser for data input. A data management agency is an organization that facilitates computer hardware and software to maintain customers' data. Sponsors who do not have computer facilities may use the service provided by data management agency on a contract base. For trials with multiple sites and low eHR accessibility, this is the preferred architecture.

Figure 2(c) shows a CTS architecture similar to the one illustrated in Figure 2(b) with a local front-end database at each site. It, therefore, shares the same advantages/disadvantages discussed previously. In this design, the CTS front end has most of the complexities and is considered a completed CTS where the backend has only limited functionalities like data migration and synchronization. For CTs without a designated data management agency, this is the preferred architecture because the CTS at a particular site can serve as the data center. Detailed designs of CTS may vary from case to case.

A process model is defined as "*a distinct set of activities, actions, tasks, milestones, and work products that are required to engineer high-quality software*" [24]. In the past few decades, many SDMs have been proposed such as the waterfall model, the incremental model, the prototyping paradigm, the spiral model, the joint application development, agile methods, and the IBM Rational Unified Process (RUP) [25–33]. Figure 3 shows the "plan spectrum" of SDM where the unplanned/undisciplined hacking occupies the left extreme, while the inch-pebble planning occupies the other end of the spectrum [23]. Based on this model, all SDMs fall between these two extremes. Some SDMs work more effectively than others in some domain, but a panacea does not exist. As a matter of fact, CTS development shares the same patterns and properties (categorized into phases, described in details in the following subsection) which is


Phase I Study to Evaluate the Safety of S&HD in Human Wounds

ASSESSMENT OF ULCER WOUNDS 01/10/2011

☐ NOT DONE

SUBJECT CENTER	SUBJECT INITIAL	SUBJECT NO	VISIT TYPE	VISIT DATE
[illegible]	[illegible]	0001	Screening/Baseline	01/10/2011

DATE EVALUATION WAS TAKEN: 11/04/2010

Assessment of ulcer wounds at baseline

1 Presence of infection:
- ☑ No
- ☐ Yes (Complete 1a and 1b)
  - 1a C-reactive protein = 0 mg/dl
  - 1b ESR = 0 mm/hr (First hour)

2 Level of exudate: *check only one*
- ☐ Minimal
- ☑ Moderate
- ☐ Heavy
- ☐ None

3 Appearance of ulcer bed: *check only one*
- ☑ Epithelium
- ☐ Granulation
- ☐ Slough
- ☐ Eschar
- ☐ Other

4 Duration of ulcer: 3 days

5 Size of target wound: 20 cm^2

6 Appearance of surrounding skin: *check only one*
- ☐ Normal
- ☑ Erythema
- ☐ Maceration
- ☐ Cellulitis
- ☐ Other

7 Current dressing regimen: *check only one*
- ☐ Hydrocolloid
- ☑ Alginate
- ☐ Hydrofiber
- ☐ Foam
- ☐ Gel
- ☐ Other

8 Position of ulcer: Load Zoom

File Name: UI_Q1310001000_1

FIGURE 1: Example eCRF for "Assessment of Ulcer Wounds".

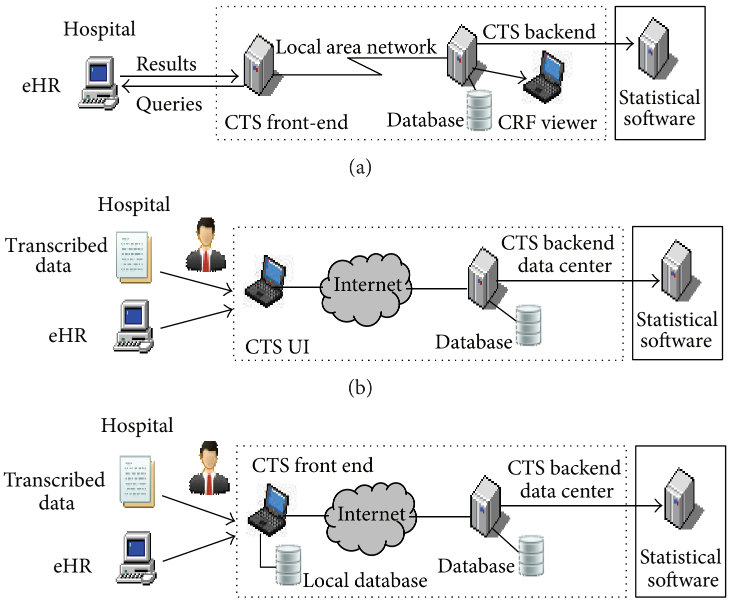


FIGURE 2: Three commonly used CTS architectures.

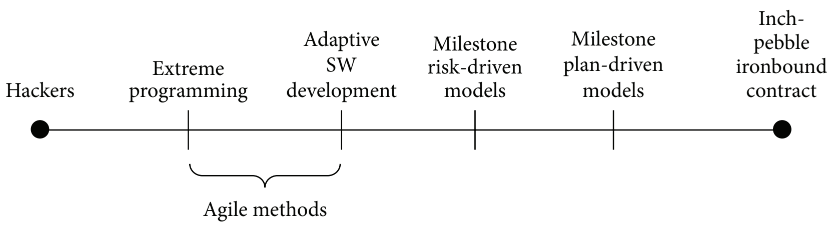

Hackers
Extreme programming
Adaptive SW development
Milestone risk-driven models
Milestone plan-driven models
Inch-pebble ironbound contract
Agile methods

FIGURE 3: The planning spectrum of SDM [23].

more similar to the SDMs on the right-hand side of the spectrum.

2.2. Process and Properties of Clinical Trials. The properties that distinguish clinical trials from other application domains are described with the example (shown in Figure 4) of FDA clearance for class III medical devices in the US market using the activity diagram of the Unified Modeling Language (UML) [34, 35]. The pathway consists of three blocks (swimlanes): a government agency (FDA), the sponsor, and participating hospitals. Activities inside these blocks can be grouped into three categories: initial, core, and postactivities.

(1) *Initial Activities*. The sponsor is responsible for developing the CT protocol. Prior to that, initial activities such as the hypothesis study, feasibility study, risk assessment, and animal study may be required. For CT of a medical device, FDA also suggests that a quality management system be established by the sponsor as soon as possible following FDA regulations [36]. If a study fails to receive positive results from the initial activities, the process will be terminated and therefore a CTS will not be needed. Hence, starting the CTS development too early is not practical.

(2) *Core Activities*. One important CT property which needs to be considered in CTS development is protocol evolution, also the main activity in this category. Three major versions of CT protocols could be generated: the initial protocol, protocol in the first submitted IDE application, and FDA approved protocol. The core activities start with the CT protocol development. The initial protocol, which can be created by the sponsor without interference from hospitals, is sent to the institutional review boards (IRB) of each hospital for reviewing. CT subject enrollment can officially start only after the IRB approves the protocol plus an approved investigational device exemption (IDE) from the FDA. An IDE application is submitted to the FDA with the CT protocol and at least one hospitals' IRB approval. After reviewing an IDE application, the FDA replies with one of three status: *approval* (CT is allowed to proceed), *conditional approval* (CT is allowed to proceed but with conditions), or *disapproval* (CT is not allowed to proceed). A conditional approval requires the sponsor to answer FDA's questions or even rewrite the entire protocol. Any changes or amendments to the protocol need to be submitted to all hospitals' IRB for reviewing. The modified protocol may contain updated CRFs due to new data specification. The hospitals' IRB may also conditionally approve the study, and then the sponsor needs to provide amendments with explanations or updates. The amendments have to be submitted to the FDA and the IRB of all sites for reviewing. If the sponsor cannot provide meaningful responses to the FDA, the study would most likely be rejected and thus terminated (no need for CTS). If the sponsor fails to provide a satisfactory answer to any of IRB's questions, chances are the sponsor has to drop this hospital from the CT since the IRB will certainly reject the study.

(3) *Postactivities*. The main focus of postactivities is to apply for premarket approval (PMA). The PMA application should be supported by statistical analysis of the CT data. The first activity in this phase is, therefore, defined as "analyzing trial data" which is obviously irrelevant to CTS development.

This pathway is complicated with certain patterns. Due to the involvement of government agencies, the pathway is different from most processes of software development in other domains such as financial, industrial, or insurance. Based on the discussion above, a CTS has the following unique properties.

CTSP1. The development time is extremely short. CTS needs to be ready when IDE is approved and subject enrollment begins, which may be the next day of the IDE approval.

CTSP2. Every CT certainly has an approved protocol. Subject enrollment cannot start without a government-approved protocol. The CT protocol is considered the requirement specification of CTS because it contains detailed information about what or how the data will be collected.

CTSP3. The protocol is subject to change anytime during the CT. The CT protocol can be changed even when the system is online. The amended protocol is considered a new version of the CTS specification.

CTSP4. Not every protocol amendment affects CTS development. Protocol amendments may contain only explanations without extra data to be collected. The CTS development team does not have to react to such amendments.

CTSP5. Protocol development may take months to finish. For the safety of the subjects, a protocol will be carefully reviewed by FDA and IRB though the government is required to respond to IDE applications within a time limit but may not necessarily approve the protocol. Hence, it normally takes an extensive amount of time to receive IDE approval.

CTSP6. The CTS development team does not participate in protocol reviewing process. Protocols are reviewed by the government agency and hospitals' IRB, and CTS development engineers have no role in the reviewing process and are therefore excluded.

CTSP7. There is no need at all for CTS developers to be colocated with customers. According to CTSP2 and CTSP6, engineers are informed of the nonnegotiable protocol (the CTS requirement specification) passively. In most cases, CTS customers (users at sites) are given a nonnegotiable software system passively. In this sense, there is no need at all to maintain face-to-face or day-to-day communication between developers and customers.

CTSP8. There is no "rapid value" of the CTS. The concept of "rapid value" of a software system comes from a situation where customers can profit from a functional subsystem. We can hardly find such cases in CTs where subject data are

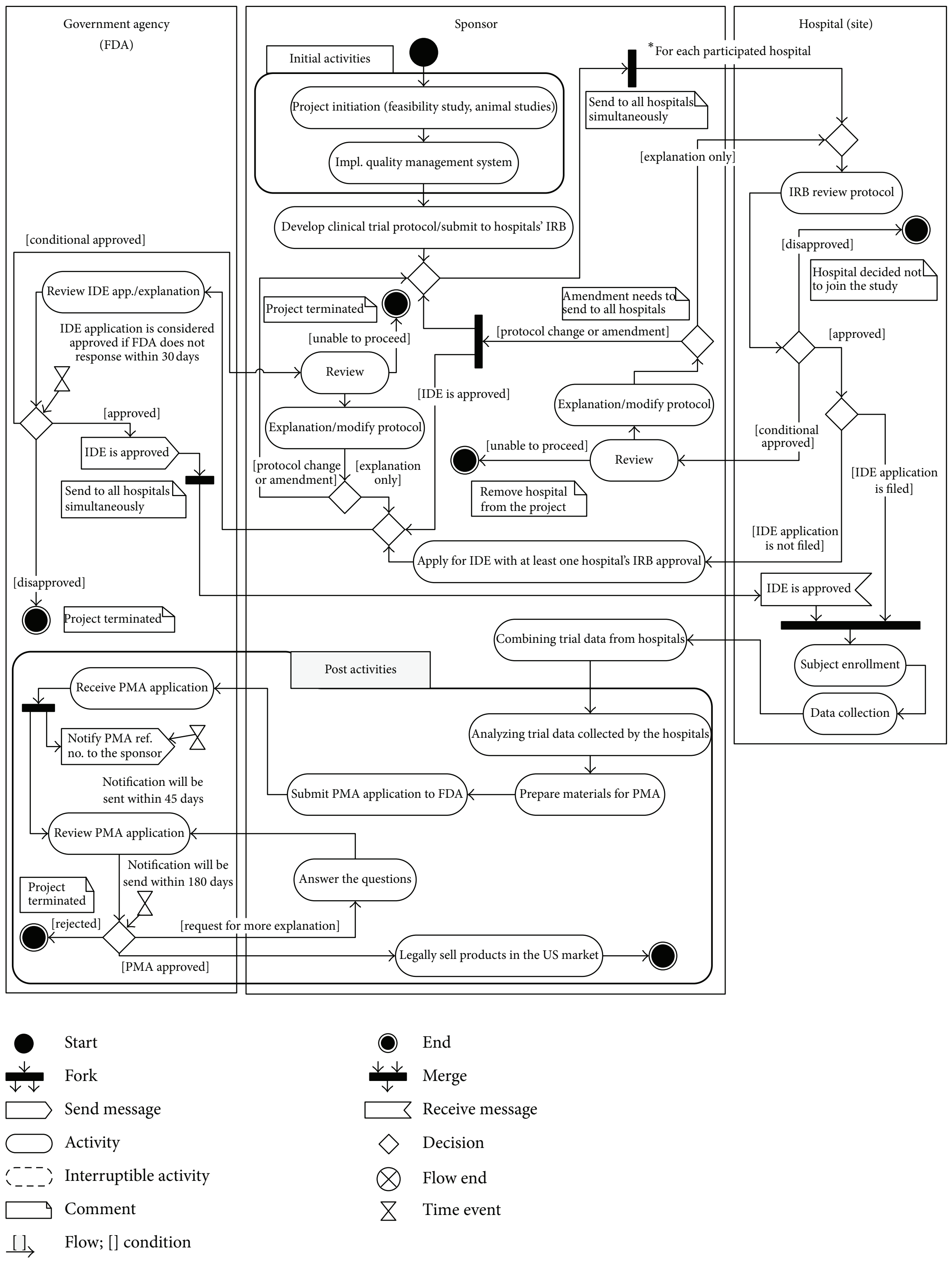

Government agency
(FDA)
Sponsor
Hospital (site)
Initial activities
Project initiation (feasibility study, animal studies)
Impl. quality management system
Develop clinical trial protocol/submit to hospitals' IRB
*For each participated hospital
Send to all hospitals simultaneously
[explanation only]
IRB review protocol
[disapproved]
Hospital decided not to join the study
[approved]
[conditional approved]
Amendment needs to send to all hospitals
[protocol change or amendment]
Project terminated
[unable to proceed]
Review
Explanation/modify protocol
[IDE is approved]
Explanation/modify protocol
[unable to proceed]
Review
Remove hospital from the project
[conditional approved]
[IDE application is filed]
[IDE application is not filed]
[protocol change or amendment]
[explanation only]
Apply for IDE with at least one hospital's IRB approval
[conditional approved]
Review IDE app./explanation
IDE application is considered approved if FDA does not response within 30 days
[approved]
IDE is approved
Send to all hospitals simultaneously
[disapproved]
Project terminated
IDE is approved
Combining trial data from hospitals
Subject enrollment
Data collection
Post activities
Receive PMA application
Notify PMA ref. no. to the sponsor
Notification will be sent within 45 days
Analyzing trial data collected by the hospitals
Prepare materials for PMA
Submit PMA application to FDA
Review PMA application
Notification will be send within 180 days
Answer the questions
Project terminated
[rejected]
[request for more explanation]
[PMA approved]
Legally sell products in the US market
Start
Fork
Send message
Activity
Interruptible activity
Comment
Flow; [] condition
End
Merge
Receive message
Decision
Flow end
Time event

FIGURE 4: The process of FDA clearance to market a Class III medical device.

collected by CTS first and later by paper-based CRF just because the CTS development is not fully developed yet.

CTSP9. Every trial is unique. Each CT study has its own hypothesis, procedures, and endpoints. Every CT study has its own unique exit criteria. Data collection and data analysis differ from case to case. For pharmaceutical companies, which host a large amount of CTS development, software modules may possibility be reused, but, in general, it is not practical to construct a universal CT protocol.

CTSP10. There is no working legacy system. Usually a functional system exists to handle current jobs for most software development in other domains. This is not necessarily the case in a clinical trial. For example, the development of a new Class III medical device may not take advantage of any existing function system for other devices. In most cases, an entirely new CTS process needs to be developed for every new CT.

CTSP11. CTS architecture can be defined without protocol. The sponsor can define the CTS architecture without an approved protocol because CTS architecture, as shown in Figure 2, only describes how data will be collected rather than what to collect and how to analyze them.

CTSP12. Data collection portion can be online without exportation module. Data collection portion can be activated without the exportation module because the exportation module is functionally independent from the rest of the CTS. Engineers need to complete data collection module for subject enrollment, but not the exportation part which will be needed only when sufficient data have been collected, which may take months or even years.

CTSP13. There are unique factors to measure the level of success for CTS development. For example, according to CTSP1, it is highly possible that subjects are ready for enrollment before CTS is available. In such cases, one can either reject the subject or use the paper-based CRF first as an alternative and then transfer the data to the CTS later when it is ready. If a CTS is delivered on time, then the number of subjects whose data are collected by an alternative method should be zero. Such numbers can be used as a measure of the successfulness of CTS development, but it is beyond the scope of this research.

2.3. Tools for Developing CTS. CTS can be developed by generic software development tools or CTS-specific tools, for example, Oracle Clinical [37]. The former includes tools which are not specifically created for CTS development; the later represents the opposite general purpose development tools, for example, Java or .Net [38, 39]. Differences between these two approaches are summarized in Table 1.

The purchasing fee for general development tools is relatively lower than the cost of a specific tool. The cost for hiring engineers or project managers familiar with a specific tool like Oracle Clinical is high since such people do have specific knowledge. In Figure 1, the user is allowed to draw on the picture at the lower right side, but for tools that do not support this kind of feature, adding this feature can cost a lot. In addition, Figure 1 shows a CTS form exactly the same as the paper-based CRF and end users with experience on paper-based CRF can easily learn how to use the CTS. On the contrary, for example, if the CTS is built by Oracle Remote Data Capture [37], then the end users must learn its user interface, which is different from what they were familiar with (paper-based CRF). If a sponsor already bought a software package, for example, a specific database management system, then it can be reused easily through generic tools, but not necessarily so for specific tools that may require modules from specific software vendor.

In general, developing a CTS with generic tools costs less than with CTS specific tools. Although free software is available, for example, EpiData [40], to generate an interface for data capturing, the stability, extensibility, and overall supports of such free software are relatively low, which not only raises the costs for adding features but also increases the risk of failure. End user of the CTS, nurses to input data or consultant to collect data on site, may need to spend more time to understand not only the CTS but other related systems of the specific tools. In practice, organizations, for example, pharmaceutical companies, host a large amount of CTS development with many experienced engineers, and project managers may prefer CTS-specific tools. Institutes with limited resources or individual researchers may prefer to use the relatively inexpensive generic tools. In the latter case, the process can be improved significantly by having an SDM built specifically for CTS development.

2.4. Reviews of Current Software Development Methodologies. This subsection reviews core concepts of the current SDMs in plan-driven methods, risk-driven methods, and agile methods as illustrated in Figure 3. Explanations of why these SDMs are not appropriate for CTS development will also be provided.

2.4.1. Plan-Driven and Risk-Driven Methods. Plan-driven methods include traditional SDMs like *the waterfall model*, *the incremental model*, *prototyping paradigm*, and *RUP*. All these SDMs define specific "phases". For example, the classic waterfall model contains five ordered linear phases: *requirement*, *analysis*, *design*, *constructions*, and *operations* [24]. The RUP contains four phases: *inception*, *elaboration*, *construction*, and *transition*. Documents and specifications will be generated at the end of each phase by engineers and customers [26]. However, all plan-driven SDMs require early finalization of system requirement, which clearly violates CTSP3. Delays in any of the early phases will compromise the development time and make CTSP1 even worse. Prototyping paradigm requires regular meeting between the developers and customers to clarify software requirement and system modules. But in CTS development, there is no need to clarify requirement since the CTS requirement is the FDA/hospital IRB approved protocol (CTSP2) which is developed and clarified by doctors and government agency without software

Table 1: Comparison of CTS development by generic and specific tools.

	Specific tools	Generic tools
Costs for acquiring tools	High	Low
Costs for qualified engineers	High	Low
Costs for adding special features	High if not supported	Low
End user learning curve	Relatively high	Low
Can CTS utilize software already acquired?	Depends	Yes
Any SDM to follow	Depends on vendors	No
Is the project manager required to know tools in depth?	Yes	No

engineers' involvement (CTSP6). Prototyping paradigm also requires constant face-to-face meeting with customers to verify their requirements or needs with a prototype. But with the FDA/hospital IRB approved protocol, there is no need for customers and developers to meet constantly. In addition, in CTS requirement, customers cannot profit from a CTS with only partial function (CTSP8). The RUP finalizes the architecture design at the second phase, while CTSP11 suggests it be done at the first phase. The RUP and other incremental models work well for developing a large-scale system that evolves as the process continues [24], but they may not be appropriate models in CTS development due to CTSP6 and CTSP8~10. Process models derived from the waterfall model, such as the incremental model [41, 42], have similar inherited disadvantages for developing CTSs.

The risk-driven spiral model is an evolutionary model with four regions: *determine objectives*, *evaluate*, *develop*, and *plan next phases* [24, 28, 43, 44]. Software is developed in a series of evolutionary releases (prototypes) from a successive loop. This model differs from other models mainly in that risk analysis is performed in each of the development cycle. The major risk of CTS development comes from its unique properties, and many of them cannot be resolved by risk analysis, for example, CTSP1. Spiral model inherits the core concept of prototyping paradigm and therefore shares the same disadvantages for CTS development. Process models derived from the spiral model, like WINWIN spiral model [45], exhibit the same problems for developing CTSs.

2.4.2. Agile Methods. Unlike the plan/risk-driven approaches, agile methods give up the concepts of "phases" and focus on the flexibility of the methods. Agile methods represent a group of light-weight methodologies including *scrum*, *extreme programming* (*XP*), *adaptive software development*, *feature-driven development*, and *agile unified process* (the light-weight version of RUP) while the XP is commonly known as the first agile method. In 2001, the *Manifesto for Agile Software Development* was published to define the approach now known as the agile software development [46]. The focal values of agile methods listed in this document are (1) *individuals and interactions* over processes and tools, (2) *working software* over comprehensive documentation, (3) *customer collaboration* over contract negotiation, and (4) *responding to change* over following a plan.

It is obvious that some of these focal values may not be proper for CTS development. For instance, for focal value (2), CTSP8 shows that a partially working CTS can hardly provide benefits to the CT. For focal value (3), contracts between the developer and customer may not be necessary if they are in the same organization, for example, a pharmaceutical company. For other cases where the CTS is developed by a software consultant firm (with CTSP1~6), negotiating a contract agreed by both developers and customers is important. For focal values (4), since all CTS development follows the same process, as shown in Figure 4, defining an SDM that matches the process can significantly help to reduce the risk of failure.

Furthermore, some of the 12 principles of agile development are not suitable for CTS development by nature. For instance, the third principle "*deliver working software... from a couple of weeks to months...*" violates CTSP8. The fourth principle "*business people and developers must work together daily throughout the project*" goes against CTSP5~6. And the seventh principle "*working software is the primary measure of progress*" violates CTSP8 and CTSP10. The sixth and eighth principles emphasize frequent colocated face-to-face conversation, but according to CTSP2 and CTSP6~7, neither face-to-face meeting nor colocation is necessary. Agile methods work well for business software development but may not work well for CTS development, since over one-third of the 12 principles of agile development will be unnecessary, difficult, or impossible to carry out. For CTS development to be successful, it requires the project manager, engineers, and the customer to carefully disregard some of the basic agile principles. For project managers and engineers without experience on CTS development, adopting agile methods may increase the risks of the project due to possible complications caused by following all of the agile principles. In some CTS development projects, the project managers, to their rich experience and understanding of clinical trials, actually disregarded some basic agile principles; therefore, it is hardly justifiable to claim that the CTS was developed by agile methods.

Table 2 summarizes the CTS properties that are improper for the plan-, risk-driven, and agile method. In reality, it is almost impossible to follow guidelines and phases of traditional SDMs in CTS development. On the other side, if one adopts agile methods, some of the focal values and principles must be ignored. Then the successfulness of CTS development totally depends on experienced engineers and project managers to ensure the project is on the right track.

Table 2: SDMs and the CTS properties that make the SDM improper for CTS development.

SDMs	CTS properties that make the SDM improper for CTS development
Plan-driven	CTSP1, 3, and 6~11.
Risk-driven	CTSP1 (the risks of CTS development are different from risks discussed in risk-driven methods)
Agile methods	CTSP1~8, 10

For CTs initiated by institutes other than large pharmaceutical companies, adopting agile methods may increase risks of the project.

3. Proposed Software Development Methodology

Due to CTSP3~5, engineers may not obtain the CTS specification at an early stage of the project, which is a necessary condition for plan-oriented SDMs. On the other hand, agile methods focus on prototypes, rapid values, and constant meetings, and so forth, disregarding that engineers can take advantage of the fact that every CTS development shares the same patterns and properties. To design an SDM that fully covers the CTS properties and takes advantage of both plan-oriented and agile methods, one needs to carefully review the protocol evolution. Hence, the new SDM for CTS development naturally resides in the middle of the spectrum as shown in Figure 3, balancing the two end points of opposite views [47, 48]. Figure 5 shows the state transition diagram of the protocol evolution at hospital and sponsor site in UML. The state transition is relatively simple on the hospital side with only three states compared to those on the sponsor side. It is because the sponsor is responsible for protocol development. The three versions of protocol are shown in the comment blocks between state transitions on the sponsor side.

To handle the unique patterns and properties of CTS development, a new SDM is proposed, as shown in Figure 6, with four phases: *architecture design*, *system analysis and design*, *construction*, and *exportation*. Activities in all phases are iterative and incremental in nature. The phases are linked with intermediate objects like documents and software. Phases will be executed one after another, but, different from the traditional plan-driven SDM, the last three phases are triggered based on receiving messages. Objects not related to other phases are placed inside the shaded blocks.

3.1. Architecture Design Phase. The first phase, *the architecture design phase*, starts when the initial activities (as in Figure 4) obtain positive results and the sponsor decides to proceed. Major work products of this phase include *architecture description*, *standard operating procedure* (*SOP*), *auditing plan*, *prototypes* (for architecture demonstration), *risk assessment*, and *project plan*. *Risk assessment*, *prototypes*, and *project plan* (resources and personnel allocation plans) are also required at the first phase in other SDMs while the rest are unique to the proposed model.

Different from most current SDMs, the sponsor and engineers can design the system architecture, for example, Figure 2, without detailed specifications (protocol). In practice, once the overall system architecture is defined, changes are highly unlikely.

CTSs with different architectures may have totally different software components. For instance, if trial data can be retrieved from hospitals' eHR, then the CTS can be designed without an interface for manual data input. If data must be manually entered, a standard 2-tier database system should suffice for a CT without complex semantic checks. Otherwise, a 3-tier system is more appropriate for hosting those checks. Along with the *architecture description*, the sponsor and engineers are able to finalize *risk assessment*, *prototypes*, and *project plan*.

The *SOP* of how to collect data is defined based on the finalized CTS architecture in this phase. Two documents, *architecture description* and *SOP*, are prerequisites for the next phase and must be completed beforehand. Since the CTS architecture demands stability, iterations of activities of this phase should be limited to the minimum.

3.2. System Analysis and Design Phase. The second phase focuses on system analysis and design. Major work products include the *analysis model*, *system and database design model*, *test plan*, *test cases,* and *alternative plan*. This phase starts when the initial protocol is complete. The major activities in this phase are system analysis and design. Once the system design is complete, the work product *system and database design model* will be created. Based on this design model, engineers now have a clear picture of how many software components and database objects a CTS will include.

For cases in which the protocol is approved by the FDA and the participating hospitals before CTS is completed, the sponsor and engineers have two choices: (1) reject qualified subjects before CTS is completed or (2) allow subjects' registration and prepare an alternative plan to collect data before CTS is ready. The most popular alternative plan is to use traditional paper-based CRF to collect data. Alternative planning requires the latest amended protocol and the *SOP*. For cases in which the initial protocol is amended, engineers should decide whether a system architecture change is required. If this is the case, the entire development process should start all over again because changing CTS architecture, for example, from Figures 2(a) to 2(c), requires developing totally different software components. For cases where changes are required only in the data fields or adding forms, engineers can go through activities defined in this phase to update the system and database design model. For cases in which the protocol is not updated (only explanations are needed), the activity flow will simply end because software specification is unchanged.

For the analysis/design paradigm, either a function-oriented or an object-oriented approach can be adopted since both approaches are well established in the field of software engineering [49]. The database design, schema design, and normalization can be achieved by following the guidelines recommended in the database system [50].

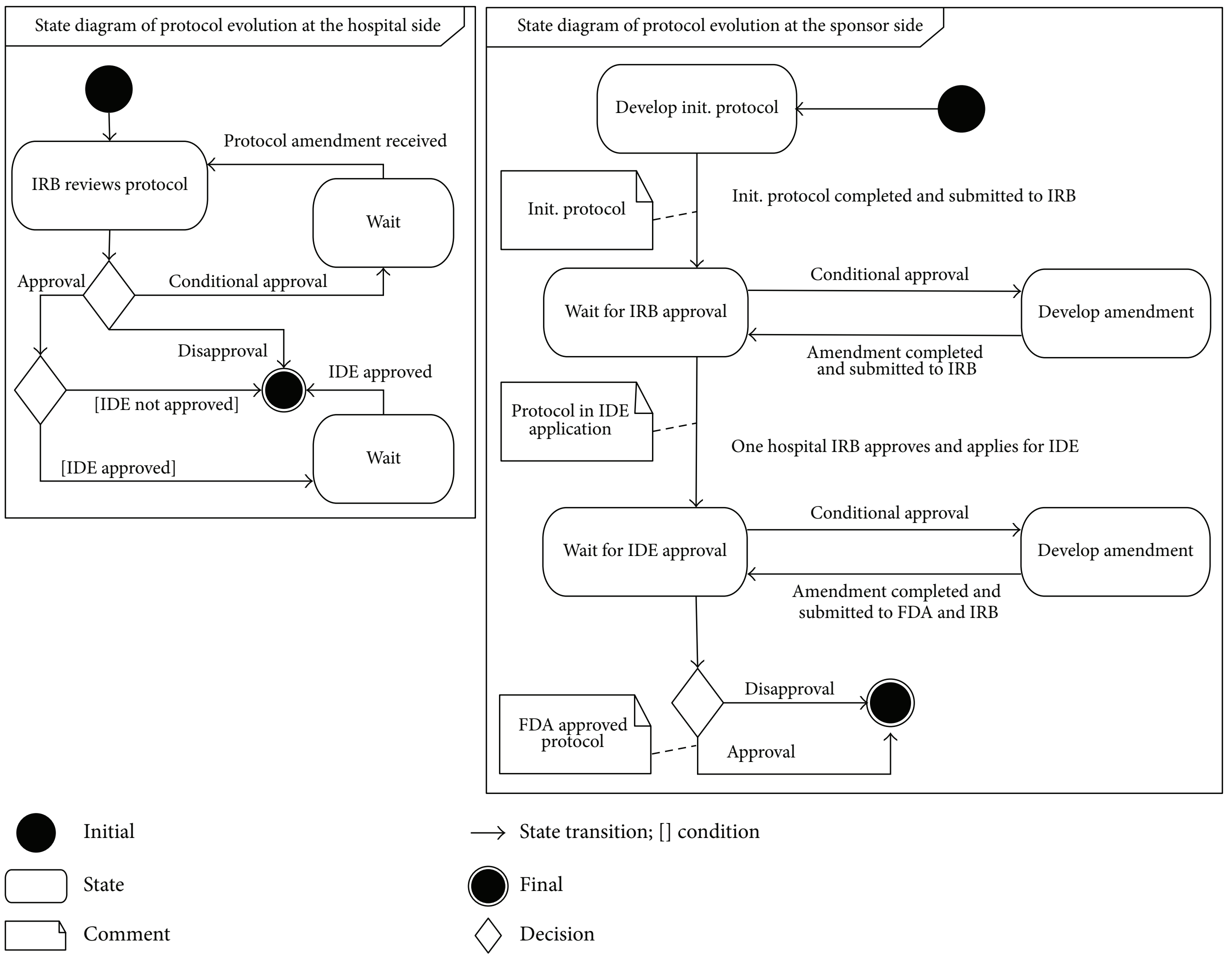

State diagram of protocol evolution at the hospital side
IRB reviews protocol
Protocol amendment received
Wait
Approval
Conditional approval
Disapproval
IDE approved
[IDE not approved]
[IDE approved]
Wait
State diagram of protocol evolution at the sponsor side
Develop init. protocol
Init. protocol
Init. protocol completed and submitted to IRB
Wait for IRB approval
Conditional approval
Develop amendment
Amendment completed and submitted to IRB
Protocol in IDE application
One hospital IRB approves and applies for IDE
Wait for IDE approval
Conditional approval
Develop amendment
Amendment completed and submitted to FDA and IRB
FDA approved protocol
Disapproval
Approval
Initial
State
Comment
State transition; [] condition
Final
Decision

Figure 5: The state diagram of the protocol evolution at hospital and sponsor site.

3.3. Construction Phase. The third phase, the construction phase, focuses on developing a functional CTS data collection subsystem. Major work products are the *software components, integrated CTS data collection subsystem*, *user manual,* and *deployment procedure*. This phase starts after the IDE application is filed, which means the protocol has received at least one hospital approval. Based on the work products defined in the previous phases, engineers create backend database objects, design (in low level) and develop software components, integrate them into a functional subsystem, and meanwhile perform the unit/system test based on the test plans/cases.

To implement a single eCRF, as shown in Figure 1, engineers need to go through a series of activities, called *construction cycle*, including component level analysis, component construction/testing, CTS integration, and integration test. Ideally each construction cycle is independent from others if the eCRF does not contain any complicated rules associated with other forms. For CTS adopting architecture shown in Figure 2(a), a component can be considered a module used to retrieve data fields for a single CRF from hospital eHR. In either case, construction cycles can be executed in parallel.

During implementation, engineers may receive conditional approval message "Data spec. changed" from FDA or IRB. In such cases, all activities must be interrupted (dashed rounded box in Figure 6). Amended protocol may "require architecture change" or "require system change". In both cases, messages will be sent and the activity flow will go back to previous phases as described in the previous subsection. Engineers may continue the construction cycles with no effects on a certain protocol amendment. Such design makes the proposed SDM maintain flexibility to handle specification changes (CTSP3~4).

If FDA approves the IDE after all construction cycles are completed, then the subject enrollment may begin with the integrated CTS data collection subsystem without any delay. Otherwise, the CTS *alternative plan* is invoked. The most important supporting documents are *deployment procedure* and *user manual*. Both of them will be updated in every construction cycle based on the integrated CTS data collection subsystem.

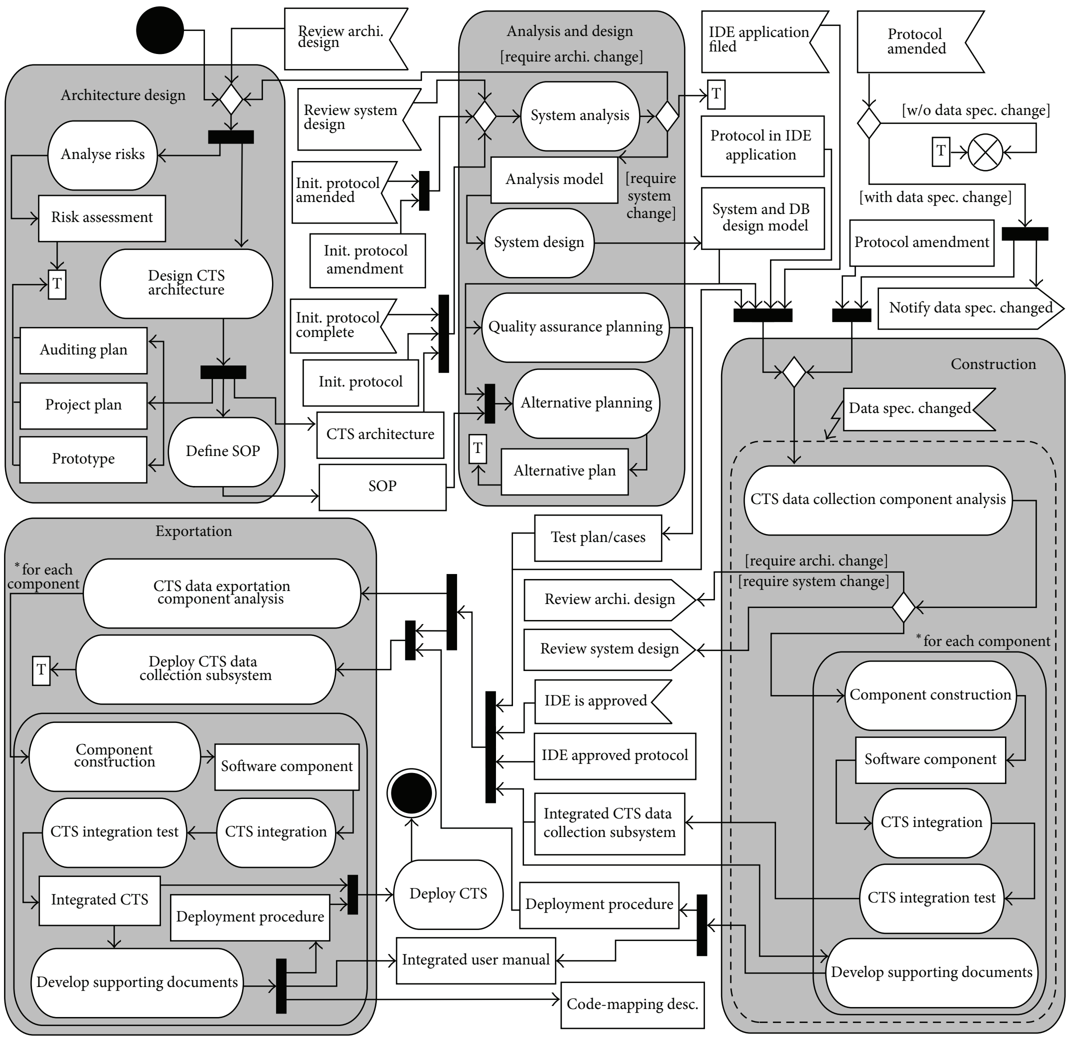


Figure 6: The proposed CTS SDM in UML activity diagram.

3.4. Exportation Phase. The last phase is the exportation phase because the CTS requires exporting data for statistical analysis. The development of data exportation subsystem needs not to be executed in the previous phase based on CTSP1 and CTSP12. Major work products are the *software components*, *integrated CTS*, *user manual*, *code-mapping descriptions*, and *deployment procedure*. This phase starts when the IDE is approved and subject enrollment is ready to begin. The first two activities, "deploy the CTS data collection subsystem" and "analysis of CTS data exportation subsystem", can be executed in parallel.

The construction cycle is similar between the third and fourth phases, but the fourth phase focuses on data exportation components. The construction cycle in both phases can be implemented in parallel if no complicated dependent rules exist. In the fourth phase, engineers work with statisticians on data exporting specifications and develop, test, and integrate software components into CTS. All supporting documents are supposed to be created at this time such as the *user manual* for data exportation and *code-mapping descriptions* for every exported data file. To generate quality analysis results, most CTSs use professional statistical software. It is also possible that sponsors wishes to use statistical features provided by commercial database management systems rather than statistical software; then "data exportation components" represent "interfaces" of database management systems for statisticians (e.g., views of database tables).

3.5. Features of the Proposed SDM. There are two backward flows in the proposed SDM from the construction phase to the first and second phases in order to preserve the flexibilities. The development is parallel since the CTS constriction is based on components. Components which are not affected by a certain amendment can stay in the third phase. This approach enables the proposed SDM to have the advantage of incremental model with agile flexibilities. As illustrated in Figure 4, there are many activity loops, for example, getting approval from a hospital IRB or to get the IDE approved from the FDA. In the meanwhile, the development team must halt and wait for the activities to finish. During that "waiting time", based on other SDMS, the development team can do nothing but wait. The proposed SDM carefully places activities in that "waiting time" as illustrated in Figures 5 and 6.

In summary, the proposed SDM covers all thirteen CTS properties with the following features: (1) architecture design is performed at the first phase, CTSP12; (2) the development of CTS is split into two well-defined subsystems, CTSP1; (3) all phases are designed to be iterative and incremental, CTSP2~5; (4) engineers do not need to be involved in protocol development or colocated with sponsors, CTSP6~7; (5) there are two concrete milestones at the end of third and fourth phases (delivery of data collection subsystem and delivery of integrated CTS), CTSP8~10; (6) the limited time for development is optimally utilized, CTSP1; (7) experiments for engineers and PMs are not necessary conditions.

4. Case Study and Discussions

4.1. Case Study. The proposed SDM has been applied on three CTS developments and one of them (for the Phase-I study of a new silicon human dermis developed in Taiwan) is discussed in this section to demonstrate the feasibility and effectiveness. The government regulations for CTs in Taiwan are very similar to those in the USA with a similar pathway as shown in Figure 4 except that the Taiwanese government is not obligated to answer either IDE or PMA application within a certain period of time.

The study initially involved four participating hospitals without a common eHR; therefore, a 2-tier architecture similar to Figure 2(c) was adopted as the CTS architecture and one of the four sites was designated as the data center. A previously developed CTS was used as the demonstration prototype. The activities of the first phase were smoothly completed before the initial protocol was completed. These activities were executed only once, since the finalized architecture was never changed throughout the entire project. The object-oriented approach was used for system analysis and design in this study. The engineers finished all activities in the second phase before receiving the approval from the first hospitals' IRB. The construction phase is iterated three times due to protocol updates. In order to implement the CTS more efficiently, components without complicated constraints like "Screening Summary", "Death Report Form", and "Physician's Comments" were built first. Two new forms and several data fields were added later in response to government/IRB requests. The final CTS involves twelve visits from each subject with several dozens of constraints adopted and over one hundred forms. Figure 1 shows one of the forms. Since the data specification becomes stable after three conditional approvals (from three iterations), the engineers were told to proceed to the last phase to develop the data exportation subsystem. The whole CTS development was completed when the IDE was approved, and the CTS alternative plan was naturally not used in this case. In the end, the new proposed SDM was successfully applied to develop and deliver a large scale CTS with high quality on time, within budget, and without using any additional resources.

4.2. Discussions. For comparison of cost, all three CTS developments adopting the proposed SDM are considered against the previous two CTS developments. The development cost was measured by accumulating all costs (for project manager, system designer, software developers, database developers, testers) divided by the number of eCRF. The average cost per form (as shown in Figure 1) was reduced over fifty percent. The later three CTS developments (with the proposed SDM) are more complex than the previous two with more forms and visits per subjects, and yet the number of meetings between sponsor and software consultant firm was reduced to one-third. Both previous CTS development failed to deliver on time. Data from some subjects were recorded on paper-based CRF without using the CTS at all. From these subjects' point of view, the CTS development totally failed.

One of the many advantages of this new SDM is that phases and work flows can be followed easily so that PMs can coordinate the CTS development effectively and efficiently. It is therefore not necessary to have members from the sponsor side to assume the role of PM; engineers or primary investigators can also be the PM, too. On the contrary, the CTS experiences of PM may become a major roadblock in a conventional SDM. As aforementioned, the risk of CTS development is high if it adopts agile methods and is supervised by an inexperienced PM. Similarly, the CTS experiences of engineers can reduce the risk of failure, but it is no longer a necessary condition because the activities are well defined and, as addressed earlier, the major risks of CTS development come from its unique properties rather than the experiences of its developers.

The results of the case study show that the proposed SDM outperforms conventional SDMs in many ways. First, the proposed SDM has the advantages of both the plan-oriented methods and agile methods. On one side, activities and phases in the proposed SDM are well planned, defined, and easily followed as in plan-oriented methods. On the other side, the proposed SDM maintains the flexibility to respond to specification changes as in a typical agile method. Second, the proposed SDM specifically discards the guidelines/principles suggested by both plan-oriented methods and agile methods which are unrelated to or violate CTS properties. Third, time is used as efficiently as possible in the new SDM to reduce the number of subjects needed to use the CTS alternative plan. Fourth, the flexibility of adopting a specific SDM for individual construction cycle is maintained. Engineers are allowed to adopt a familiar SDM, for example, the waterfall model, for a particular construction cycle for data collection and exportation to develop individual software components. Finally, the flexibility of a CT with multiple CTSs in different architecture is preserved. It is not necessary to have a CTS for all sites in the proposed SDM. Each CTS with different architecture can be implemented independently as long as the data exported to statistical software are considered acceptable by statisticians.

Based on the experiences of developing CTS with the proposed SDM, a few guidelines are proposed to reduce the risk of project failure. First, engineers should work closely with both the sponsors and the statisticians. Before the third phase is initialized, engineers must have a clear idea of what data will be collected and what data format will be used. At the exportation phase, engineers should provide data files in the formats the statisticians demand. For example, data in the same form with different visits can be exported differently (into one file or file per visit). Next, it is important to make sure that the engineers follow the proposed SDM and understand the unique properties of CTS beforehand. The sponsor cannot assume that the engineers who may have never participated in CTS development are familiar with the pathways as illustrated in Figure 4.

Handling late updates is always a challenge; that is, protocol changes after CTS is online. It may be rare but can happen. If the missing data are available in hospital records, the currently online CTS can still be used to enroll subjects. In the meantime, engineers should restart the activities in the second phase to retrieve necessary data. If the amended protocol requires data that are unavailable in hospital records, the data collected from enrolled subjects should be discarded and a new CTS development should start all over again.

5. Conclusion

Over half a century ago, software engineers recognized the importance of SDM and proposed several primitive and pioneering models like the code-and-fix and stage-wise model [51]. Since then, many SDMs were introduced, and the chance of successfully delivering quality software has been greatly improved by adopting a proper SDM. In this paper, the properties and patterns of developing a CTS are carefully examined. Based on these properties and patterns, a new SDM is proposed specifically for developing a CTS. The proposed SDM is verified in a large-scale real-world case. Results show that the advantages of plan/risk-driven and agile methods for CTS development are preserved by this new SDM. With the proposed SDM, project managers' and engineers' experience with CTS development is no longer a necessary condition. This SDM increases the successful rate of delivering quality software on time within budget and reduces the risk of the project. Moreover, it benefits both parties of software development because the sponsor/project managers can gain more control of software development and the developers can focus on software engineering. With a high quality software, data collection, management, and analysis can be more efficient, accurate, and less expensive. Consequently, the overall quality of a clinical trial can be significantly improved.

As for future works, more case studies of CTS developments can be performed with the proposed SDM. Research teams with recourses can apply the proposed SDM with an existing SDM on a same CTS development at the same time as the experiment group and control group, respectively, to compare the results.

References

[1] R. Tiruvoipati, S. P. Balasubramanian, G. Atturu, G. J. Peek, and D. Elbourne, "Improving the quality of reporting randomized controlled trials in cardiothoracic surgery: the way forward," *Journal of Thoracic and Cardiovascular Surgery*, vol. 132, no. 2, pp. 233–240, 2006.

[2] US FDA, "Final rule for good laboratory practice regulations," *Federal Register*, vol. 52, pp. 33768–33782, 1987.

[3] R. D. Kush, P. Bleicher, W. R. Kubick et al., *eClinical Trials: Planning and Implementation*, Thomson CenterWatch, Boston, Mass, USA, 2003.

[4] L. Clivio, A. Tinazzi, S. Mangano, and E. Santoro, "The contribution of information technology: towards a better clinical data management," *Drug Development Research*, vol. 67, no. 3, pp. 245–250, 2006.

[5] T. H. Payne and G. Graham, "Managing the life cycle of electronic clinical documents," *Journal of the American Medical Informatics Association*, vol. 13, no. 4, pp. 438–445, 2006.

[6] T. Schleyer, H. Spallek, and P. Hernández, "A qualitative investigation of the content of dental paper-based and computer-based patient record formats," *Journal of the American Medical Informatics Association*, vol. 14, no. 4, pp. 515–526, 2007.

[7] T. J. Kuhn, "The difference between eDC and eCRF," http://tjkuhn.wordpress.com/2008/03/14/the-difference-between-edc-and-ecrf.

[8] G. Nahler, *Dictionary of Pharmaceutical Medicine*, Springer, New York, NY, USA, 2009.

[9] B. Ene-Iordache, S. Carminati, L. Antiga et al., "Developing regulatory-compliant electronic case report forms for clinical trials: experience with the demand trial," *Journal of the American Medical Informatics Association*, vol. 16, no. 3, pp. 404–408, 2009.

[10] I. Pavlovic, T. Kern, and D. Miklavcic, "Comparison of paper-based and electronic data collection process in clinical trials: costs simulation study," *Contemporary Clinical Trials*, vol. 30, pp. 300–316, 2009.

[11] R. Kush, L. Alschuler, R. Ruggeri et al., "Implementing single source: the STARBRITE proof-of-concept study," *Journal of the American Medical Informatics Association*, vol. 14, no. 5, pp. 662–673, 2007.

[12] D. Kim, S. Labkoff, and S. H. Holliday, "Opportunities for electronic health record data to support business functions in the pharmaceutical industry–A case study from Pfizer, Inc.," *Journal of the American Medical Informatics Association*, vol. 15, no. 5, pp. 581–584, 2008.

[13] K Yamamoto, S. Matsumoto, H. Tada et al., "A data capture system for outcomes studies that integrates with electronic health records: development and potential uses," *Journal of Medical Systems*, vol. 32, no. 5, pp. 423–427, 2008.

[14] Integrating the Healthcare Enterprise International. IHE IT Infrastructure Technical Framework Supplement—RFD supplement, http://www.ihe.net/Technical_Framework/upload/IHE_ITI_TF_Suppl_RFD_TI_2006_09_25.pdf.

[15] J. R. Deitzer, P. R. Payne, and J. B. Starren, "Coverage of clinical trials tasks in existing ontologies," in *AMIA Annual Symposium Proceedings*, p. 903, 2006.

[16] R. H. Dolin, L. Alschuler, S. Boyer et al., "HL7 clinical document architecture, release 2," *Journal of the American Medical Informatics Association*, vol. 13, no. 1, pp. 30–39, 2006.

[17] R. L. Richesson, J. E. Andrews, and J. P. Krischer, "Use of SNOMED CT to represent clinical research data: a semantic characterization of data items on case report forms in vasculitis research," *Journal of the American Medical Informatics Association*, vol. 13, no. 5, pp. 536–546, 2006.

[18] J. E. Andrews, R. L. Richesson, and J. Krischer, "Variation of SNOMED CT coding of clinical research concepts among coding experts," *Journal of the American Medical Informatics Association*, vol. 14, no. 4, pp. 497–506, 2007.

[19] M. W. Haber, B. W. Kisler, M. Lenzen, and L. W. Wright, "Controlled terminology for clinical research: a collaboration between CDISC and NCI enterprise vocabulary services," *Drug Information Journal*, vol. 41, no. 3, pp. 405–412, 2007.

[20] CDISC, http://www.cdisc.org/.

[21] T. Strasser and G. Lotz, "An integrated system for workflow and data management in clinical trials," *Investigative Ophthalmology & Visual Science*, vol. 49, E-Abstract 5216, 2008.

[22] US FDA, "21 CFR Part 11: electronic records; electronic signatures; final rule," *Federal Register*, vol. 62, no. 54, p. 13429, 1997.

[23] B. Boehm, "Get ready for agile methods, with care," *Computer*, vol. 35, no. 1, pp. 64–69, 2002.

[24] R. S. Pressman, *Software Engineering: A Practitioner's Approach (7/e)*, McGraw-Hill Inc., New York, NY, USA, 2009.

[25] W. Royce, "Managing the development of large software systems: concepts and techniques," in *Proceedings of the WESCON*, 1970.

[26] H. Mills, "Top-down programming in large systems," in *Debugging Techniques in Large Systems*, R. Ruskin, Ed., Prentice Hall, Upper Saddle River, NJ, USA, 1971.

[27] L. Bally, J. Brittan, and K. H. Wagner, "A prototype approach to information system design and development," *Information and Management*, vol. 1, no. 1, pp. 21–26, 1977.

[28] B. W. Boehm, "Spiral model for software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[29] J. Wood and D. Silver, *Joint Application Development*, Wiley, New York, NY, USA, 1995.

[30] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, Boston, Mass, USA, 1999.

[31] J. Highsmith, *Agile Software Development Ecosystems*, Addison-Wesley, Reading, Mass, USA, 2002.

[32] IBM Rational Unified Process (RUP), http://www-01.ibm.com/software/awdtools/rup/.

[33] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software DevelopmentProcess*, Addison-Wesley, Boston, Mass, USA, 1999.

[34] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, Mass, USA, 2nd edition, 2005.

[35] US FDA, "Running Clinical Trials," http://www.fda.gov/ScienceResearch/SpecialTopics/RunningClinicalTrials/default.htm.

[36] US FDA, 21 CFR Part 820: Quality System Regulation; 2007.

[37] Oracle Clinical, http://www.oracle.com/us/industries/life-sciences/clinical/overview/index.html.

[38] Java Platform, http://www.oracle.com/technetwork/java/javase/overview/index.html.

[39] Microsoft.Net, http://www.microsoft.com/net/.

[40] S. Bennett, M. Myatt et al., "Data Management for Surveys and Trials: A Practical Primer Using Epidata," http://www.epidata.dk/downloads/dmepidata.pdf.

[41] H. Mills, "Top-down programming in large systems," in *Debugging Techniques in Large Systems*, R. Ruskin, Ed., Prentice Hall, Upper Saddle River, NJ, USA, 1971.

[42] J. Mcdermid and P. Rook, *Software Development Process Models. Software Engineer's Reference Book*, CRC Press, 1993.

[43] L. Maciaszek, *Practical Software Engineering: A Case-Study Approach (8/e)*, Addison-Wesley, Reading, Mass, USA, 2004.

[44] I. Sommerville, *Software Engineering (8/e)*, Addison-Wesley, Reading, Mass, USA, 2006.

[45] B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy, "Using the WinWin spiral model: a case study," *Computer*, vol. 31, no. 7, pp. 33–44, 1998.

[46] Agile Alliance. Manifesto for Agile Software Development, 2001.

[47] B. Boehm and R. Turner, "Using risk to balance agile and plan-driven methods," *Computer*, vol. 36, no. 6, pp. 57–66, 2003.

[48] B. Boehm, *Turner. Balancing Agility and Discipline—A Guide for the Perplexed*, Addison-Wesley, Reading, Mass, USA, 2004.

[49] S. R. Schach, *Object-Oriented and Classical Software Engineering (7/e)*, McGraw-Hill, Boston, Mass, USA, 2006.

[50] A. Silberschatz, H. Korth, and S. Sudarshan, *Database System Concepts (6/e)*, McGraw-Hill, Boston, Mass, USA, 2010.

[51] H. Benington, "Production of large computer programs," in *Proceedings of the Symposium on Advanced Programming Methods for Digital Computers*, pp. 15–27, 1956.