Linking System Development Processes Together by Documentation

WANWU GUO

School of Computer and Information Science
Edith Cowan University
2 Bradford Street, Mount Lawley, WA 6050
AUSTRALIA

Abstract – Management centric and programming centric approaches have been widely adopted in guiding systems development. Progressive system development is based on separate but yet linked development phases. In this paper, based on the author's and other supervisor's experiences in supervising student projects, a documentation centric approach is proposed for guiding the progressive development of software and information systems. This approach targets on achieving the highest possible reusability of the outcomes from the current phase to the next phase. Case studies show that projects adopting this approach have all made smooth transitions from one phase to another whereas those failing to do so have to abandon the further development due to insufficient supporting information inherited from its previous development.

Key-Words: - Systems development processes, Project documentation, Progressive systems development, Systems development model, Project management

1 Introduction

Experience in supervising student projects shows that students have a great difficulty in applying their project management knowledge mainly learnt from theoretical study to a practical project, especially in the early stage of a project. The project supervisors must in the first instance give students an appropriate guideline that can be easily understood and subsequently followed by the students, also against which the progress of the project can be checked by the supervisors during development [1][2]. An effective and widely used guideline is to set up project milestones and deliverables with progress [3]. Most of these deliverables are captured in a number of documents, such as project proposal, progressive reports, and final project report. The detailed requirements on these documents shape the pathway of the progress and determine the format of the final delivery of a project. These requirements vary largely depending on both the types of projects and the management styles of project supervisors. The subsequent outcomes are vastly different.

In this paper, based on the author's and other supervisor's experiences in supervising such projects, a documentation centric approach is presented for guiding student projects on software and information systems that are subject to consecutive developments. This approach aims to achieve the highest possible inheritance to and/or reusability for the next development phase of the system. However, it does not change the project management hierarchy and thus can be easily integrated into the current systems development models. Case studies show that projects adopting this approach have all made smooth transitions from one phase to another whereas those failing to do so have to abandon the further development due to insufficient supporting information inherited from its previous development.

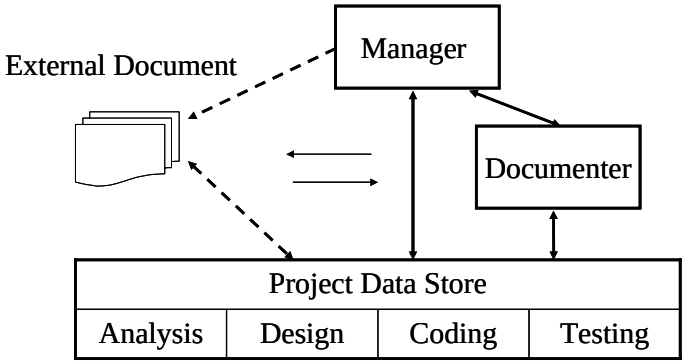
2 Linking Systems Development Together Using Documentation

Project practitioners are familiar with the concepts of management centric and/or programming centric methods in systems development [3][4][5]. In large scale system development involving contributions from many different departments or developers, management centric approach is vital in coordinating all the major processes. Documentation is largely handled at the subsystem level because these processes are carried out concurrently. On the other hand, the programming centric approach must be adopted if the development follows the extreme programming (XP) method. Documentation plays a minor role in such system development.

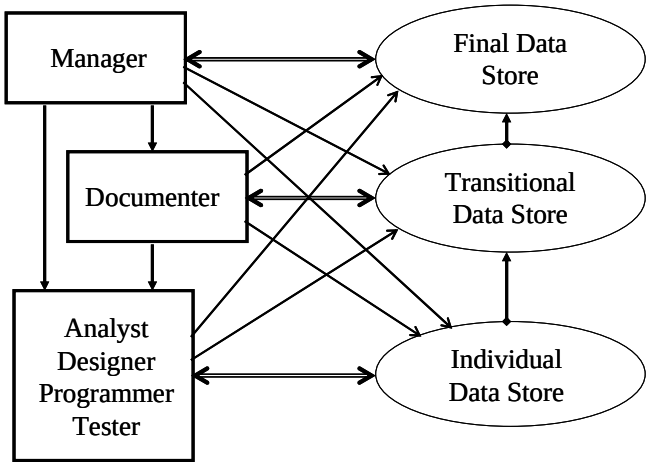
For software/information systems to be developed over a few consecutive development phases required by the clients, the tasks within each phase are relatively easier to achieve due to the focus of the

development being well defined. However, the articulation of the outcome of the current phase to its subsequent development becomes the top priority, which means that the documentation centric approach should be adopted in this case.

Documentation centric does not mean that the member responsible for documentation, or the documenter, is controlling the development progression. Instead, it is a new way to restructure the data flow path and redefine the documentation responsibility for individual team members. Figure 1a illustrates the status of documentation for a conventional system development. Individual project analysts, designers, programmers, and testers have their own directories to store their outcomes. A copy of these data directories forms the project data store for the project secretary or documenter to compile various types of documents required by the project leader who would have the right to finalise them. Everyone concentrates on his/her own share of documentation and the final project report is a structured collection of final documents from individuals. The project documenter is mainly a document editor.



(a)



(b)

Figure 1 Data flow path on documentation in traditional (a) and documentation centric (b) approaches

In the documentation centric approach, there are three types of separate but yet linked data stores for supporting the project (Figure 1b). The key factor of responsibility allocation is that the project analyst will take the leading role in documentation after finishing the analysis stage, and has the sole right to do write operations on documents in the Transitional Data Store (TDS). This is because the analyst should know the details of the project requirements for different development phases, and thus is able to prioritise the items in terms of their inheritance and reusability for the subsequent developments. The documenter also has the sole right to transfer any final draft from TDS to the Final Data Store (FDS) managed only by the project leader or manager. In addition, the documenter is able to access any particular document in the Individual Data Store (IDS) managed by the designer, programmer, and tester respectively so as to gain more detailed information on a specific issue.

The project manager has the sole authorisation to do write operations on the final drafts in FDS sent by

the project documenter only. This allows only the manager to modify and finalise the final documents required in the project handover so as to maintain the highest consistency.

Table 1: Documents stored in different data stores

Final Data Store (FDS)	Managerial Documents	• Minutes of meetings • Project plan • Task breakdowns • Budget balance
	Deliverable Documents	• Project proposal • Progressive reports • Project development overview • Final results of system analysis • Final results of system design • Final collection of source code • Final testing plans • Final results of tests and their analysis • Final version of user's guide • Final version of implementation track record
Transitional Data Store (TDS)	Internal Formal Documents	• Internal memos • Final draft of project proposal • Final draft of project development overview • Final draft of results on system analysis • Final draft of results on system design • Final draft of collection of source code • Final draft of testing plans • Final draft on test results and their analysis • Final draft of user's guide • Final draft of implementation track record
Individual Data Store (IDS)	Internal Formal Drafts	• Analyst's final document on system analysis • Designer's final documents on system design • Programmers' final documents on system implementation • Tester's final documents on testing plans, cases, results and analysis • Programmers' final notes on system installation and management • Developers' training notes • Programmers' final version of implementation track record
	Internal Temporary Items	• All records for intermediate results • All original records on analysis, design, implementation, and testing • Notes on implementation trials

The designer, programmer, and tester have their own IDS to keep their initial, intermediate, and final outcomes of components or processes during the course. The final version of a component document with highlighted changes made to the original design is transferred to TDS from IDS only by the corresponding designer, programmer or tester. It is a huge advantage for the system designer to lead the testing because this person must know what exactly the system should respond to the testing cases. To maintain a high level of transparency during the project, all data stores are also made accessible to all the team members.

Different types of documents are stored in different data stores. The details of these data items are shown in Table 1. Although many documents are kept in all data stores, the final project report delivered externally to the client only consists of an extended project development overview (main body of the project), along with three attachments – a technical reference, a user's guide, and an implementation track record (optional). The contents, purposes and readers of these constituents are listed in Table 2.

It should be noted that the system design in the first development phase sets up the framework of the whole system and a brief scope for the next development phase as well. The objectives of next development phase should be outlined in the current project development overview so as to smoothly articulate the two phases together. The implementation track record will play an important role in programming for the subsequent development phase if the extreme programming or rapid

prototyping method is used to direct the system development. This is because the track record provides both the successful and failed attempts in realising a specific function. This information allows the next development not only to adopt the similar technique for successfully realising system functions, but also to avoid the repetition of the failed attempts in the new development phases.

Table 2: Documents included in the final project report

Document	Contents/Purpose	Reader
Main body of report (project development overview)	• introduce project objectives and development procedures • summarise achievements and failures • illustrate system architectures • summarise realised system functionalities • analyse system performances using testing outcomes • explain reasons for unachieved tasks • suggest ways to improve the system in the future	division chiefs, department directors, branch managers, project leaders, and other managerial staff in the client organisation
Technical reference	• references to systems analysis (eg DFDs etc.) • references to systems design (eg DB design, page layout, UML diagrams etc.) • collections of source code • references to test design, cases, outcomes, and analysis	system analysts, designers, programmers, and testers in the client organisation, and service providers
User's guide	• system installation guide • system user's manual • system administrator's manual • trouble shooting instructions	system users and administrators, and service providers
Implementation track record	• list of both successful and failed attempts using different implementation techniques	programmers, designers, and service providers

Documentation centric approach is only a new presentation of the data flows and a new allocation of documentation responsibility for individual team members during project, so it does not change the overall procedure of systems development processes. Therefore, it can be easily integrated into the current systems development models.

The example shown in Figure 2 illustrates the development procedure using the waterfall model. The forward development stages keep the same sequence in turn with each stage adding its documents to the project data store for referencing to other stages. The backward links among the stages are replaced by the two-way connection between the data store and each of the five processes. This means that a change at a lower stage incurs the updates of the corresponding section at the higher stages.

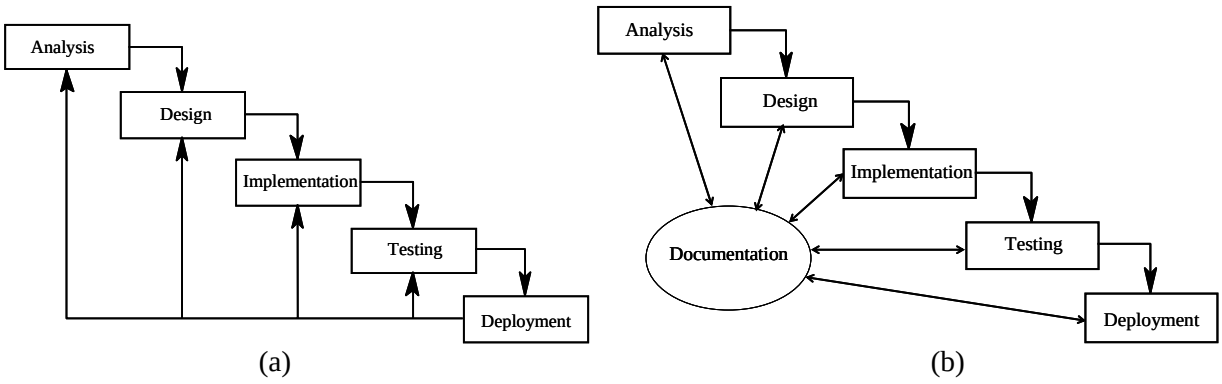


Figure 2 The waterfall system development model (a) and its documentation centric structure (b)

3 Case Studies

A government record authority proposed a Web-based information system as a student project in 2003. It was to create a Website for propagating the 250-year history of a port city. The Web pages should be classified by different themes, and the events within each theme are sorted in chronological order. In addition to the text information on each

page, some relevant historical items, such as photos, paintings, sketches, maps, and handwritings, would be digitised and displayed on the page as well. The initial system should only be installed in a few government libraries for the general public and school students to assess locally. Its subsequent developments should make the system accessible anywhere through the Internet.

In the second semester of 2003, a group of four students started this project. The students were instructed by the supervisors to adopt the documentation centric approach for their system development. Its focus should be on firstly designing a system framework that could be followed by all the development phases, secondly setting up a prototype system for immediate use, and thirdly documenting the needs for future development of the system.

A three-phased project proposal was developed by this team with the mutual agreement between the authority and the project supervision group. The tasks for each development phase are listed in Table 3. This team delivered a Flash-based Website for the authority in the end of the four-month project. Although the system was created using client-server architecture and should be accessible through the Internet, this prototype was primarily installed in a few government libraries for local access due to the highly demanded bandwidth in transferring the multimedia rich features through network. A detailed final report was compiled, which was also passed to the next team for further development.

In the first semester of 2004, another team of three students implemented a HTML-based Website that works in parallel with the Flash-based Website. This team also created a number of templates for staff at the authority to easily build new pages. The students sensed the emerging of new technologies and techniques in constructing dynamic Web applications in the near future, so in the final report they suggested the authority to update its Web server so as to support a new database-driven information system for the long term services.

Table 3: Specifications of development phases for the Web-based information system

Development phase	Tasks	Outcome
Phase 1	• design system framework • standardise fundamental layout • create Flash-based Website • outline needs for next development	• a usable Flash-based Website accessible locally • detailed project report
Phase 2	• implement HTML-based Website • provide templates for adding new pages • outline needs for next development	• a fully functional HTML-based Website coexisting with Flash-based Website • a set of page templates • detailed project report
Phase 3	• create database system interacting with Web server • facilitate partly dynamic Web contents • enable users to register and upload data to database • outline needs for next development	• integrated database-driven Website • detailed project report

Adopting this recommendation requires the approval from the authority management, which may not be the priority to the authority at the time. Also staff at the authority was satisfied with the performance of the HTML-based system and the easy inclusion of new themes and events using the templates provided. This led to the discontinuation of the system development till the first semester of 2006.

The success of this Web-based information system in use in 2004 and 2005 creates new demands from the interested public groups for expansion. One strong demand is to set up more similar Websites for other historic locations or heritages. The other is to allow individuals to share their personal/family collections related to a specific theme or event with general public by enabling them to upload their digitised data items to the database. To meet these needs, a new and more powerful Web server interacting with a separate database has to be enabled, which was suggested in the final report of the second development phase.

A new project was carried out by a small team of two students in the first semester of 2006 to achieve some modified tasks. Instead of directly implementing a database-driven Web system, the two students were asked to create a separate back-end system using PHP and MySQL to facilitate dynamic web content technologies and user-enabled uploading. The rationale was that once this trial is successful, a real system can then be easily set up by migrating/modifying and expanding the existing

stuff to the real system. The prototype of this database-driven system was completed and tested, and it was ready to be adopted in the future system development. An integration plan was proposed in the final report of this trial development phase.

This case demonstrates that the documentation centric approach is useful and effective in systems articulation and maximising reusability in systems development.

In contrast, two groups of students took two projects on software development for online gaming in the first semester of 2005. Their project supervisor gave the teams flexibility in choosing their own system development models as long as they could deliver usable prototypes that could be further developed by other teams in the coming semesters. One team chose the rapid prototyping method and the other preferred the XP approach. In the end, one team produced a workable prototype and the other even failed to complete the prototype on time. Even worse was neither team had documented project proposal, documents on system analysis, design, and testing, and implementation track record. The programmers admitted that they even could not remember how many different tries they had made during the development. Repetition of previously failed attempts often occurred to the same programmer or other programmers in the same team during the course because there was no document to check with. The further development plan had to be abandoned after the first trial.

4 Conclusion

Developers are familiar with the concepts of management centric and/or programming centric methods in systems development. The former is vital in coordinating all the major processes in large scale system development whereas the latter is preferred to the development following the extreme programming (XP) method. Documentation plays a minor role in such system development. In progressive software/information systems development over a few consecutive development phases, the documentation centric approach is useful and effective in systems articulation and maximising reusability.

Documentation centric does not change the project management hierarchy. It is to restructure the path of data flows and redefine the responsibility on documentation for individual team members. Therefore it can be easily integrated into the current systems development models.

In the documentation centric approach, the key factor of responsibility allocation is that the project analyst should take the leading role in documentation after finishing the analysis stage. The analyst knows the details of the project requirements for different development phases, and thus is able to prioritise the items in terms of their inheritance and reusability for the subsequent developments.

The two case studies show that project adopting this approach have all made smooth transitions from one phase to another whereas the projects failing to do so have to abandon the further development due to insufficient supporting information inherited from the previous development.

References

[1] W. Guo, A principle-tactics based approach for guiding student project management, *WSEAS Transactions on Computers*, 2, 2003, pp. 48-52.

[2] W. Guo, Guide students managing project team from their own practice, *Informing Science*, 7, pp. 267-277.

[3] R.T. Futrell, D.F. Shafer, L.I. Shafer, *Quality software project management*, Prentice Hall, 2002.

[4] J. McManus, T. Wood-Harper, *Information systems project management*. Prentice Hall, 2003.

[5] I. Sommerville, *Software engineering*, Addison-Wesley, 2001.