An Outline of a

Knowledge

Acquisition Based Approach

to

Software Project Planning

TR002 (1991)[1]

by Karl Reed, MSc, FACS
Rev 1.0 2006[2]

Amdahl Australian Intelligent Tools Program

Department of Computer Science and Computer Engineering

La Trobe University

Bundoora, Victoria 3083

Australia

[1] .Originally a Position Paper for CASE90, Irvine CA, Dec 1990.
[2] A PDF version of the original document (TR002kr) was created 22/4/2006, with some adjustment to layout of the index. Otherwise, the word version (TR002kr.complete.r1.doc) and the corresponding pdf version (TR002kr.complete.r1.pdf) are identical to the original.

Abstract

This paper argues that the relative levels of skills or "knowledge" available at the beginning of a software project, and the skills needed to perform the tasks constituting the project rather than some pre-ordained "software process" model should be used to generate a Project Plan. These approaches in turn should also influence the design and use of software development tools.

The paper identifies and describes a series of Knowledge or Skill Domains which can be used to develop a Software Project Plan.

Acknowledgments

Discussions with Professor R Jeffrey of UNSW, and Mr J Cybulski of AAITP and Mr N Stern of RMIT have proved helpful.

Any errors are entirely the responsibility of the author.

Table of Contents

1. Introduction

Unless the factors which determine software productivity are understood and influence the design of software development tools, then the impact of those tools will be limited. In addition, the factors which determine the evolution of a software project should influence Tool design.

Equally importantly, these factors, rather than some fixed software process model, should determine the way a software project is actually planned.

It is now widely accepted that a variety of "Software Process Models" are necessary. (eg. The Waterfall Model [Royc70], Boehm's spiral model [Boeh87] and various prototyping models ([Agre86], Part III). There have also been proposals ([Oste87], the "Process Programming" approach of Arcadia, and the contract model proposed for ISTAR [Dows86] which lead to infinitely variable process models. Considerable debate has occurred on the relative merits of these proposals. However there is almost no advice available on how to choose between them or on how to tailor the process model.

We shall use the term "Software Project Plan", to imply that a particular plan is an instantiation of a more general software process model.

2. Knowledge and Software Engineering

The process of moving from some concept of an artefact to the realization of the artefact itself can be considered to consist of two parts:-

a) The acquisition of the **skill** necessary to carry out the various tasks necessary to fabricate the artefact, and

b) The actual performance of the tasks themselves once the skill is acquired.

I have argued that the major difficulties with project planning and estimating occur because this fact is not generally recognized. [Reed76].

We can take this into account when planning a Software Project by ensuring that the necessary skill is available at the point in time when a particular task is to be commenced.

This is a knowledge or skill acquisition process of a more general kind, involving training or problem analysis as necessary.

An examination of the steps taken during, and of the techniques and tools used, in a software project, suggests that there are a small number of (relatively) distinct knowledge domains.

So far, the only software development knowledge domain explicitly dealt with in the literature is what we will call the Application Domain. This is the Domain Analysis process. ([Pret90], [Aran89], [Simo87] for example).

These ideas by others (see for example [Most87] and [Aran89]), and discussed in embryonic form by Brooks [Broo75] in his answer to the so-called "siren song" of software development (ibid. pp. 47-50).

The skill is also implicitly recognized in most estimating techniques (See [Boeh81]).

First, however, we describe the five software knowledge and skill domains.

3. The KABASPP

3.1 The KABASPP Domains

The five domains are:

a) **Application Domain:** the physical laws, organizational structures, procedures etc. which govern the software artefact to be produced.

b) **Application Solution Domain:** the collection of machine executable descriptions (algorithms) which make it possible to realise the application as software.

c) **Development Environment Domain:** the complete set of tools, techniques and methods used to both develop elements of a) and b), and to realise them as software.

d) **Run Time Environment Domain:** the set of characteristics, relating to the particular machine environment under which the software must run.

e) **Managerial Domain:** the techniques necessary to plan, estimate and manage the project.

These "domains" are not mutually exclusive.
Their existence is hinted or implied at, but not elaborated in other work.

The contents of each domain are summarised in the list below:--

3.2 The Domain Contents

KABASPP Domains and Components are not equivalence classes. The lists are given are not necessarily complete. However, they provide a clear indication of the categories of skill in the use of or knowledge about subsystems, techniques or tools required in each case.

APPLICATIONS DOMAIN

Examples

- Acceleration characteristic of a train
- Organizational structure of business
- Rules for issuing air-line tickets or degrees
- Procedures for organizing work flow
- Procedures for design of pressure vessel etc.

Discipline Responsible

- Commercial systems analysis
- Engineering
- Engineering Design Analysis
- "Knowledge" Engineering

APPLICATIONS SOLUTION DOMAIN

Examples

- Algorithms for searching lists
- Approximate method for calculating acceleration of train
- Procedure for allocating seats on a vehicle given multiple access
- Path optimization's procedure for routeing of information
- Algorithm for rotating graphic images
- Procedure for recovering disc-space
- Sort procedures

Discipline Responsible

- Computer Science
- Graphics
- Artificial Intelligence
- Software Engineering, etc.

DEVELOPMENT ENVIRONMENT DOMAIN

Examples

- Programming languages
- Methodologies {JSD, SD, Modular Design}
- Tools - CASE, other development aids, Test tools
- O.S. and control language - shell, MCP, DOS, JCL etc.

- Utilities - loaders, file manipulation, editors, configuration managers
- File structure, Database management systems.

Disciplines Responsible

- Computer Science
- Software Engineering

RUN-TIME DOMAIN

Examples

- Operating Systems interfaces
- Database management systems calls
- Instruction set, external interfaces
- Resource constraints (ie. profile of available cpu time, input/output, mem for the system)
- Response time
- Device peculiarities
- Hardware Reliability's Design goal

Disciplines Responsible

- Computer Science
- Computer Engineering
- Software Engineering

and finally

PROJECT MANAGEMENT DOMAIN

Examples

- Estimating
- Project Planning
- Project Organization
- Resource acquisition
- Selection of people and development and run-time domains!
- Project Tracking
- Customer Liaison
- Quality Assurance

- System Delivery
- Maintenance Planning

Disciplines Responsible

Commercial EDP and Software Engineering and KABA.

3.3 Use of KABASPP

The following general procedure can be used to develop a software project plan, using an existing software process model as a base.

a) Identify the components of the KABASPP domains required for the project,

b) Specify the times at which they must be available, based upon the software system being produced,

c) Assess their current availability, in terms of skills and specific "solutions" etc.,

d) Using a, b and c above, construct a plan which ensures that the necessary skills are available when required.

Further work is needed to clarify the precise manner in which a complete methodology should function, and to specify tools capable of simplifying its use.

4. Conclusion

We have outlined the beginnings of an approach to Software Project Planning.

Additional work is required to develop a complete methodology.

However, the technique has the capacity to allow managers to choose between conflicting process models. It provides a basis for understanding the failure modes in the Waterfall model, and the characteristics of other process models.

5. References

[Agre86] Agresti, W.W., (1986), "*New Paradigms For Software Development Tutorial*" IEEE Computer Society Press.

[Aran89] Arango, G., (1989), "*Domain Analysis - From Art Form to Engineering Discipline*" Proc, Fifth International workshop on Software Specification and Design, pp 152-159.

[Boeh81] Boehm, B.W., (1981), "*Software Engineering Economics*" Prentice-Hall.

[Boeh87] Boehm, B.W., (1971), "*A Spiral model of Software Development and Enhancement*" in Software Engineering Project Management pp 128-142, Thoyes, R.H. Ed IEEE Tutorial.

[Broo75] Brooks, F.B.J., (1975), "*The Mythical Man-Month: Essays of Software Engineering*" Addison-Wesley.

[Dows86] Dowson, M., (1987), "*ISTAR - An Integrated Project Support Environment*" Proc. Second SDE Conference (1986), Sigplan Notices Vol. 22, No.1, pp 27-33, Jan. 1987.

[Ibco88] Ibcoe, ., (1988), "*Domain-Specific Reuse: AN Object - Oriented and Knowledge based Approach*" in IEEE Tutorial on Software Reuse.

[Most85] Mostow, J. "*Foreword: What is AI? And What Does It Have To Do With S.E.*". IEEE Transactions on Software Engineering Vol. SE-11 No.11, Nov. 1985, pp. 1253-1256, Special issue on AI&SE.

[Oste87] Osterweil, L., (1987), "*Software Processes Are Software Too*", Price. 9th International Conference on Software Engineering, pp 2-13.

[Pret90] Preto-Diaz, R., (1990), "*Domain Analysis: An Introduction*" ACM Software Engineering Notes Vol. 15, No.2 pp 47-54.

[Reed76] Reed, K., (1976), "*Economic Factors in Using Private Software Services*" Proc. Soft-where, Soft-why, Soft-how, Australian Computer Society SIC Seminar, 1976.

[Royc70] Royce, W.W., (1970), "*Managing the Development of Large Software Systems*" proc. IEEE Wescon pp 1-4.

[Simo87] Simon, M.A., (19??), "*The Domain-Oriented Software Life Cycle Towards an Extended Process Model for Re-useability*" Proc. Workshop on Software Reliability and Maintainability, Oct. 1987.