Sparse Design: Cross-Disciplinary Communications in Software Development

Gail Thornburg
Online Computer Library Center (OCLC), Dublin, OH, USA

thornbug@oclc.org

Abstract

One of the precepts of traditional models of software development is that detailed specifications, that is, what precisely the software should do, must precede the design and then creation of the code. This paper details scenarios of non-conformance to this model, and explores the issues that arise from the failure of the model.

Keywords: knowledge representation, software design, information retrieval

Introduction

The Austrian philosopher Karl Popper has been described as promoting “a world in which the give and take of debate is highly esteemed in the precept that we are all infinitely ignorant, that we differ only in the little bits of knowledge that we do have, and that with some co-operative effort we may get nearer to the truth” (Percival, 1991). Complex software development projects tend to require thoughtful cooperation and careful interaction with numerous other individuals. While it would be grandiose to describe many software engineering efforts as empirical research, the notion of testability is more than the focus of a formal system test plan. The flexible frame of mind that can see problem definitions evolving over a project lifetime is an asset when dealing with technical challenges incompletely solvable by software.

When the software being developed attempts to emulate to a degree the expertise of professionals in a certain discipline, it can be helpful to view the project as an instance of knowledge representation. Knowledge representation can assume a wardrobe of conceptual garments, but can be simply defined as expressing problem domain knowledge in a computer tractable form, preferably one amenable to human understanding (Shanahan, 2000; Kramer, 1990). The domain is the subject or disciplinary specialty, and consult with one or more *domain experts* is necessary to develop the system and verify its results. Usually a feature of the domain is a level of uncertainty. Rules for making judgments thus cannot always be expressed with complete certainty, and so mechanisms are developed to analyze and combine the levels of uncertainty in a given case, so to arrive at what a human expert would call a reasonable decision. The decisions are heuristic rather than certain. So, implicitly, a computer system modeling them will not always guess right.

At the same time, more general model of the process of software development involves the elicitation of user requirements from those parties requiring the software, flowing into detailed technical requirements, and then detailed design documents. The detailed design specifies the overall architecture and may include specifics for construction of the software itself. Formal tests and their review constitute the basis for acceptance of the software’s functionality. The traditional idea is that software implementation cannot progress reasonably and systematically without detailed analysis of functional requirements having proceeded first.

This paper describes problems software engineers encounter in projects that combine a need for knowledge representation and the absence of usable requirements from the users, what might be termed a sparse design environment. What must be remembered is that expertise in a given subject, and even ability to describe that expertise, does not automatically confer skill in creating requirements or specifications for software to perform in that subject area. Yet given the many studies documenting abilities of experts to describe the ways they make decisions [e.g. Ericsson and Simon (1993)], there would seem to be reason for persistence in working with experts in the hope of making the software more intelligent.

Development Scenario

Discussion in this draws on experience encountered in several settings, and draws on the Wade-Giles to Pinyin Conversion project (Pinyin) as an exemplar. The Pinyin project is described more fully elsewhere (Bennett, 2000, Thornburg, 2002).

In brief, it involved the attempt to convert all transliterated Chinese text in several very large, library-oriented databases from the older standard of Wade Giles to the more current Pinyin scheme. (A brief background on Chinese transliteration schemes can be found in a recent paper (Yewang, 2000).) Each record in the databases constituted a surrogate, or meta representation, for an item which the record described. For example the item could be a book, map, journal article, web page, or other object.

The key project aspects to be elaborated here:

- Why this conversion project posed unusual complexities for software engineering. Wwhy did the project team consider this a problem of modelling expertise.
- What were the special constraints on use of a traditional model to organize software development
- What cooperation was involved
- What were the communication issues
- Why did it seem helpful to see this as knowledge representation
- How might experiences in this project help devise better methods for future projects

Why Expertise

The problem to be explored involves the challenges of interacting most effectively with domain experts to develop the algorithms behind the requirements. To distinguish the two by example: a requirement could be that the software convert all the Wade Giles Chinese to Pinyin, but not convert anything that merely contained syllables that matched Wade Giles, while the algorithms would express the strategies employed by the software to decide this. An “algorithm" here is loosely defined to include strategies, rules, design safeguards, and anything else that the software can do to accomplish an outcome. More than one algorithm could lead to the same requirement being satisfied.

So some techniques for developing requirements don't always help with the actual algorithms behind the requirements. In fact it may be best the domain experts not be charged completely with developing them -- but they must be involved. One's experts may be able to discern good strategies implementable in software, or they may ask for something easy enough to state but technically infeasible [semantic distinctions might in some cases be an example].

For complex processing, there can be numerous diverse strategies employed to make things work . For simplicity's sake, suppose we assume that for one user requirement, we need to develop one algorithm.

The problem of converting alphabetic representations Chinese characters from one transliteration system to another, where one knows both the value of first representation [syllable] and the second, could be seen as a simple process of dictionary lookup. The Wade-Giles (WG) syllable hs`ueh will always translate to xue, ching will always translate to jing, and so on. Where's the challenge to that?

One challenge lay in the nature of the database to be converted. The nature of the record layouts for the databases to be translated was such that a record coded Chinese was not necessarily Chinese in all its subparts. With WG including generic syllables such as "a, an, no, to, de, la … " the possibility of mistranslation of non-Chinese phrases was significant. The overlap of less common Chinese syllables with transliterations of Japanese or Korean was even more confusing.

If that were not enough, even assuming the phrase to be Chinese, one could not assume that finding a given syllable in a Wade Giles dictionary lookup meant that it should be translated. The Pinyin representations of syllables have considerable overlap with the WG, and worse yet, syllables that look like WG but represent different sounds. Effectively, to reconvert Pinyin would mangle the meaning and be quite difficult to identify and clean up after the fact.

At one more remove were the differing rules governing construction of the subparts of the records, library cataloging procedures which are detailed and complex. These rules in some cases dictated different handling of the same WG phrase in different parts of a single record. Yet to devise different decision rules for each part of a complex record was intractable and probably not maintainable.

Lastly, semantics. The nature of Chinese transliterations is that syllables are short and quite generic; one might have a dozen different meanings depending on context. This semantic understanding of Chinese was beyond the scope of the software design. One hedge was to attempt to achieve the longest translations of phrases as a group, then shorter ones, before attempting to convert everything left on a syllable by syllable basis. This also allowed for conversion of Chinese place names and subject phrases that were not technically Wade Giles. This made the number of phrase dictionaries large, with their ordering important. The rules for use of these dictionaries was stated subpart by subpart for elements of the record, in prescribed orders of precedence, with each conversion needing to know had been converted already and what had not.

Special Constraints

The deadline for the milestone conversion was the first constraint. The technical team, meaning the programmers and project management who were not domain experts, became involved in the Pinyin effort at the time the Library of Congress was announcing that Day One of the switchover for library cataloguers to use of Pinyin would be October 2000. Since the deadline for the conversion not only involved the joint efforts of the OCLC Online Computer Library Center, Inc. (OCLC), the Library of Congress (LC), and the Research Libraries Group (RLG), but also its tacit acceptance by libraries worldwide, schedule flexibility was minimal.

In addition the three organizations were spread over several different parts of the country, and various team members traveled enough that continual accessibility by phone or email was not possible. The technical staff at each organization used different operating systems and hardware, programmed in different languages, and generally could not share each other's expertise.

Finally, the project by nature had high visibility. The Library of Congress serves as the de facto national library in the United States, and OCLC and RLG are two of the largest bibliographic utilities, serving library organizations around the world. Librarians and academic administrators all over the world had every opportunity to observe and complain about the direction and outcomes of the project.

Two types of record formats needed conversion: those for what are described in cataloging as authority records, and bibliographic records. Since authority records are used in cataloging to do authority control

(Berger, 1985) on the second type of records, the team also had to ensure that the authority record conversion could take place first. This was a constraint in the sense that there was much less time to complete the first type of software, despite its importance.

Cooperation Factors

In terms of cooperation, what stood out was the corsortial nature of the project. While the MARC formats (Library of Congress, 2002a, 2002b) for the records in all the databases were shared standards, implementations and database architecture varied from one organization to another. The organizations were not bound contractually to agree with each other, but such agreement was vital for success.

Records in each database were also shared among the various organizations by regular updates, and capabilities of one system compared with another had to be analyzed to assure, for instance, that the integrity of the record subparts would be maintained.

The public announcements and guidelines were published by LC, but with the acceptance and coordination with the other two organizations. The three had to agree to the same set of conversion specifications, and attempt to keep these specifications synchronized. LC was responsible for the approval of the software tests conducted on the software developed by the other two. Official discussion of timeline, test status and problems identified took place once a month in a telephone conference call among the three organizations.

Communication Factors

Cross disciplinary communication was required among the Chinese language experts, library science cataloging experts, the programmers, project management, and librarians who needed to deal with the effects of the conversion efforts on their East Asian collections. Some of the experts in Chinese were also cataloging experts, but some of the experts in cataloging or other areas of librarianship knew little Chinese, and the software developers generally knew no Chinese and fairly little cataloging.

The conference call was intended as a mechanism to assure that issues from team members had a forum. Since these were only 90 minutes in length and difficult to schedule, they were limited in coverage.

The other formal communication mechanisms were web page updates by LC, and presentations on plans and status at annual conferences that took place during the project. In addition LC would email all the team members new versions of the specifications each time they were issued.

In one or two cases a domain expert from one organization would visit one of the other organizations to discuss problems.

Less formal communications by email and less frequently by telephone evolved in the necessity of rapid clarifications. Even major milestones such as the approval of software for conversion tended to be communicated by email. The levels of these communications multiplied, as noted below. In fact few channels, from phone to fax to email to ftp to wireless palm device, were not exploited to the max -- whether day, night, or weekend.

Though version control was practiced in individual cases, there was no effective short term way to assure this across the three participant organizations. Although rules for conversion decisions needed to be uniform across institutions, the software was developed independently, dictionaries were loaded and corrected independently at each organization, and the two test sets of records chosen by LC had to be maintained separately on servers at each organization's location. This was a challenge both in coordination and cooperation. Dictionary cleanup and specification markup were ongoing efforts

Rules and Guesses

The software developers reading the initial Pinyin conversion specifications could be said to have experienced distinct disbelief. The first section of the specifications included general discussions of types of materials such as personal names, and a second part enumerated specific field /subfield to convert within the record format. Internal contradictions between the general descriptive section and the field-by-field detail were immediately apparent. Such a situation isn't unique to this project, but it can be fatal to a deadline.

Certain areas of the descriptive part, such as those dealing with personal names conversions, were quite detailed and explicit in rules and examples. Other areas within the description were sketchy at best. Examples of conversions using some of the phrase dictionaries were unclear or contradictory. Description of special handling required for Taiwan place names, except for a few brief examples, was absent. LC gave the explanation that they would prefer to see what the software could do before they provided any written specifications for the Taiwan section. The team accomplished this, but learning from examples is hard work.

Other parts of the versions one specification contained requirements for conversion of very specific types of place names. The software was in fact written before it developed the writers believed that not instances of such conversion should exist.

A pattern developed somewhat reminiscent of a key concept in Popper's view, that knowledge can be said to progress by unjustified guesses, by tentative solutions to our problems, or conjectures (Popper, 1963). What holds this random-sounding process in check is falsifiability, that is, we require of ourselves that our hypotheses be phrased in a form that would permit them to be refuted.

Now, the opportunity for making guesses was clear and present in the initial specifications. One part of the OCLC development team, that handling the personal names conversion software, took the stance of asking no questions of the specification drafters, but rather coding as tightly, narrowly, and literally as possible to the description in hand. This was on the conscious assumption that requirements could be relaxed more easily than they could be tightened, in the code itself.

Other parts of the team could not take this approach as there was simply too much ambiguity. Discussions frequently took the form of eliciting comments from the specification writers, with guesses as to the missing sentences.

Design "Discoveries"

One issue that arose with the first reading of the specs was an expectation on the part of the developers. Having looked at the initial spec, it was not hard to predict that numerous changes would take place at very detailed levels for subfields within a record. Though it flew in the face of ordinary good software design, the modules were broken down to the level of fields and subfields. This was intended to serve the need to trace negative fallout from changes, and to allow realistically for a low level of granularity in the specifications. This meant many more modules to control and change, but a large proportion of changes could be accomplished in a routine manner.

There was a corollary decision early in the design process, that the software would not be optimized. The development team decided that the extreme domain complexity and perceived need for numerous changes was already an enormous challenge to the codability of the project in the language/platform in use. Moreover the expectation that the software would be retired in time (i.e. the Pinyin conversions would be done) argued for expending the scarce programmer energies toward traceability and rapid change.

Though the test records selected for conversion were excellent in their exploration of diverse cases, scalability was simply not demonstrable with the 300-450 records of the test sets. (The databases were up to

50 million records in size.) Larger scale tests were conducted by subteams and specialized subtests were devised to gather larger samples of specialized cases of conversion, i.e., to test the completeness of the examples.

Conjectures, Falsifications too

With so much to be clarified in the initial specification, communications proliferated among team members within OCLC, and then with LC. Programmers would go first to an internal cataloging expert on the team, then if necessary one of them would ask LC for clarification (and copy the other). Significant emails received were archived in the document management/control system in use at LC.

In the multiple chains of communication among all the parties, inevitably separate threads began to spin off for different areas. It was tricky to manage the email communications. Parties might remember to copy every team member on a question, or not. In retrospect, use of a threaded archived email list might have been useful, but it is not clear whether that would have been more informative (in the immediate sense) or merely overwhelming, given the volume. In addition the highly varying levels of expertise, and the fact that team members didn't knew each other beforehand, might have inhibited persons more experts in one area from openly querying those with differing expertise, due to fear of being perceived as having asked a "stupid" question.

Nonetheless the email discussions led to significant software revisions and clarifications. They also formed the basis for discovering the rules for deciding what should be converted and what should not.

As the project progressed, a new version of the spec recognized the complexity of the overlap in the WG and Pinyin syllable dictionaries by proposing a decision table for the convertability of a phrase of text. It identified four categories of Chinese syllables: 1) those unique to Wade Giles 2) those unique to Pinyin 3) those syllables which are the same in WG as they are in Pinyin (no translation, same sound), and 4) those which occur in both WG and Pinyin but map to different meanings. Anything else was labeled as "Other" in the analysis of the phrase.

This helped quite a bit to develop screening rules for conversions of text. If for instance the software encountered a string of text whose syllables were all either type 4 or a mix of type 4 or type 3, the software would have no way of determining whether it was already converted to Pinyin or not. Such cases were generally flagged for manual review.

With this much categorization of the syllables, the developers could write screening software that would serve as a gatekeeper, to avoid attempting high risk conversions. This was progress. This was still not enough. The databases to be converted were cooperative efforts among hundreds of institutions, not tightly controlled as to contents, so effective prediction of cases to avoid was impossible. Yet examples of "don't convert that" were the type of comments usually obtainable from the experts. Generalizing about identification of such cases in uncontrolled free text fields was much harder.

Text strings the software thought was Chinese but too risky to convert had to be flagged for manual review. At the same time no one wanted to write software for a conversion which only resulted in half the records being flagged for this labor-intensive fate.

A limitation of the draft categorization of Chinese syllables described above was that it did not take account of mixtures of English or other languages which could co-exist in one specific subfield of a record. A way had to be found to reduce the cases where such mixtures were flagged for manual review. So for particular fields likely to contain such mixtures, the subfields might be broken down based on separator punctuation s traditionally used in cataloging practice (such as brackets, colons, slashes). Then each of multiple parts of one subfield could be analyzed for "safe Chinese" conversion by itself.

This helped reduced the rates of records flagged, but they were still too high. Discussions ensued, and ensued. Stoplists were developed to prevent non-Chinese common phrases from preventing conversion.

Cases of special patterns of types of syllables and "Other" were described, experimented with, often rejected. This was a difficult, imprecise process, but probably was necessary to achieve a feasible outcome for the conversion of the databases as a whole.

In addition, more explicit accounting for syllables which might mimic Chinese but were not. A "too-common" list was developed empirically for the prevalent multi-language syllables de, la, juan, san (etc.) the presence of which might influence a Chinese syllable counter to think the text was reasonable to convert. For instance it might be nice to have software 'smart' enough to know it could reasonably convert a such a string :

From	Edited and annotated by **Yen tzu ch'un ch'iu**.
To	Edited and annotated by **Yan zi chun qiu**

But be wary of converting

From	A concordance to **Yen tzu ch'un ch'iu**
To	A concordance **duo Yan zi chun qiu**

Rules were developed experimentally (and not without trepidation) which in effect weighted some rules and discouraged others, to minimize bad conversion decisions.

Expectations about the nature of records in the databases were continually violated. Unmarked instances of Japanese/Korean in supposedly Chinese records sent up a warning flag. Rules for distinguishing seeming Chinese from possible Japanese/Korean had been developed. Yet this highlighted the need for the requirements not only to describe cases of good examples, but also discriminators and warnings of negative cases. In communicating possible rules it helped to be able to present 'falsifying' cases to the domain expert. Discrimination can be much more powerful in pruning the search space than positive description.

Where no good rules could be found for special conversions, and an area of the specification was still weak, methods for flagging converted records with temporary markers were devised. This was to make a posteriori discoverable levels of conversion cases which were "correct according to the spec" but which might not have made semantic sense.

So, gradually, the developers were getting used to making guesses about patterns that might work to reduce noise, and at the same time the domain experts were becoming accustomed to suggesting rules for dealing with problem patterns of text in a probabilistic way.

Outcomes

At the literal level the Pinyin project could be said to be reasonably successful, in the sense that the conversions were approved and completed, outright misconversions were low, and the average manual review rate was 12-15% for most categories of the databases.

Perhaps its more notable success was that of achieving such intense consortial cooperation over such a span of time and change. It could be argued that such a consortial effort needs at least one "champion" in each organization – one person determined to make it happen, whatever is required. Nor could the project have succeeded without the extraordinary efforts at communication by the domain experts, and intense efforts by the development team.

Future Work

What could be helpful is a framework of analysis to identify major types of checkpoints for developers. Such a schema or structured checklist might include many considerations, a few of which are noted below:

- Does the requirement and its algorithm apply to enough data to be worth implementing at all?
- Does a rule not only describe but if possible discriminate?
- Do cases have as much effort at falsifying built in as is possible?
- Can expert derived rules/cases be simplified or synthesized by a learning approach, perhaps using conceptual clustering? This might be employed in the process of developing rules, or even implemented in a sort of learning system.

These are only starting points for a framework of analysis; further study is needed.

Traditional versus What: Conclusions

An outline of this development cycle could have been captioned “From Speculations to Specifications.” The technical challenges of the Pinyin project above were unusual in levels of complexity. Some development staff predicted in advance that the Pinyin conversion simply could not be done by machine. What was also unusual was the level of opportunity for developers to become intrigued, challenged, even excited by the process of requirements development itself.

Yet other projects in software design for information retrieval take on problems that are not one hundred percent solvable by machine. That is one of the gratifying aspects of this specialty. The Pinyin project was hardly the first case of so-called satisficing. It may be more the norm to have problems where only a certain percentage correctness is the best that can be conceived. Certainly the current compression effect of life cycles on schedules, and the increasing numbers of projects competing for the time of a domain expert, mitigates for imperfect requirements/ specifications being available to the development staff at the time their clock starts ticking.

The traditional waterfall methodology for software design described in the introduction is not the only philosophical approach to software engineering. At the other end of the spectrum, Extreme Programming has recently gained some prominence as an approach to dynamic, risky development scenarios (Wells, 2001). Extreme Programming, or XP, emphasizes teamwork, pair programming in which two programmers work at one module but accomplish more from the interaction, and programmers are shifted regularly from one area of code development to another.

This approach has intuitive appeal for cases such as the Pinyin project where requirements were highly uncertain and programmer expertise tended to clump in areas rather than being spread evenly over the team. While working in pairs tends to have a learning curve, and the pair programming idea requires that each pair have a non-neophyte, it is alleged to pay off in productivity.

Constraints to use of this methodology should be noted. In XP one caveat is that the team owns the schedule. A related idea is that overtime is discouraged. In cases such as Pinyin, where a major deadline has been established, in particular by multiple cooperating institutions, control of the schedule becomes problematic. The discouragement of code reviews may pose issues for organizations conforming to audited certification requirements such as ISO or IEEE. Another expectation is that the customer or user of the software is always available to the developers, which is unlikely in some information retrieval domains.

Elements of several approaches are probably useful in a dynamic development project. One commonality with the hybrid approach taken by the Pinyin project and other approaches is that they share the recognition that a project team cannot afford to stall itself in the syndrome "but **that** can't be done yet because **this** hasn't been developed…" In some cases it is simply necessary to pretend, and generate some form of limited results, so they in turn can generate an important and underrated product – discussions.

Still, the conclusion seems clear, that flexible views of software design, where *firm* definitions of requirements are recognized to be *future* definitions, have a real place in modern software development environments. Software engineers need to be able to make guesses, challenge rules, and above all suspend their disbeliefs.

References

Bennett, Rick (2000). Bringing Chinese Cataloging Records Up To Date. *OCLC Newsletter*. March/April, 18.

Berger, Robert H. (1985). *Authority Work; The Creation, Use, Maintenance, and Evaluation of Authority Records and Files*. Littleton, CO: Libraries Unlimited, Inc.

K. Anders Ericsson and Herbert Alexander Simon (1993). *Protocol Analysis: Verbal Reports as Data* Cambridge, MA: MIT Press.

Ericsson, K. Anders, and Jacqui Smith [editors] (1992). *Toward a General Theory of Expertise: Prospects and Limits*. Cambridge University Press.

Kramer, B. M., and Mylopoulos, J. (1990). Representation, Knowledge. *Encyclopedia of Artificial Intelligence*, Volume 2, 882-890. Stuart C. Shapiro, Editor-in-Chief. John Wiley and Sons, Inc.

Library of Congress (2000). Library of Congress, Other U.S. Libraries Join International Community on Use of Pinyin. Retrieved July 9, 2002, from the World Wide Web http://lcweb.loc.gov.catdir/pinyin/romconver.html .

Library of Congress (2002a). *MARC 21 Concise Format for Bibliographic Data*. Retrieved December 13, 2002 from the World Wide Web http://www.loc.gov/marc/bibliographic/ecbdhome.html

Library of Congress (2002b). *MARC 21 Concise Format for Authority Data*. Retrieved December 14, 2002 from the World Wide Web http://www.carl.org/tlc/crs/Auth0001.htm

Percival, R. S. (1991). About Karl Popper; an Introductory Biography. Retrieved December 13, 2002, from the World Wide Web http://www.eeng.dcu.ie/~tkpw/intro_popper/intro_popper.html .

Popper, Karl R. (1963). *Conjectures and Refutations: The Growth of Scientific Knowledge*. Routledge, London.

Shanahan, James G. (2000). *Soft Computing for Knowledge Discovery; Introducing Cartesian Granule Features*, p. 23. Kluwer Academic Publishers.

Thornburg, Gail. (2002). The Syllables in the Haystack: Technical Challenges of Non-Chinese in a Wade-Giles-to-Pinyin Conversion. *Information Technology and Libraries* 21 (3), 120-126.

Wells, Don (2001). Extreme Programming; A Gentle Introduction. Retrieved December 14, 2002 from the World Wide Web http://www.extremeprogramming.org .

Yewang, Wang. (2000). A Look into Chinese Persons' Names in Bibliography Practice. *Cataloging and Classification Quarterly* 31 (1), 51-81.