Patterns in Software Assisted Software Engineering

Srinivasa Gopal

International Institute of Information Technology, Bangalore, India

E-Mail: srinivasa32@hotmail.com

Abstract ;

Software Assisted Software Engineering is used to automate different life cycle phases by using tools. But using such automated tools also creates problems that have to be resolved. In this paper two such problems of using tools during a Software Engineering Life Cycle have been presented.

1.0 Introduction :

Software Assisted Software Engineering or Computer Aided Software Engineering is used to automate different life cycle phases of development. The output of SASE or CASE is a tool for automating a particular phase in the Software Development Life Cycle. A collection of such tools becomes a framework such as Tools for Configuration Management, Tools for Verification and Validation etc. Some advantages of using these tools are to Improve Productivity, Improving Quality, Using Standard Models, Improving and Assisting Intra Team Communication

Some of the problems in this domain are the limited capacity of these tools, the commitment to invest from the management of organizations who expect an ROI from their investments in CASE tools, the learning curve required to master these tools. In this paper two potential patterns in this domain have been presented.

The first pattern looks at the problem of why available tools are not used and the second one, the difficulty of creating an automated tool that can anticipate human behaviour and statistical meta-data about a query space.

2.0 The Patterns

2.1 Pattern Name: Reluctance to Use Tools during Software Development

2.1.1 Problem: Although automated software tools (CASE Tools) that automate activities involved in different life cycle phases exist, software professionals do not use them. This problem has been persisting ever since the mid 90s, and till date there has not been any visible improvement.

For example, during the requirements development phase, modeling tools can be used to assist end users to visualize how an application will look after development. Subsequently,

system designers can use automated design tools to transform the requirements into a detailed design. Programmers can then use automated code generators to convert the design documents into code.

Tools can also be used for performing several maintenance activities. Some examples are maintenance projects such as application migration, application upgradation; program optimization etc can use Reverse Engineering Tools.

Tools can also be used to assist debugging activities.

However, in addition to these well understood advantages and disadvantages of using CASE tools is the lesser understood reluctance of software professionals to actively embrace and integrate tools into their work routine or into different life cycle phases.

Also the fact that the Software Engineers do not use and embrace Software Engineering tools is not clearly surveyed and hence not clearly documented into Software Project Plans.

2.1.2 Context

Software companies and universities that have active software development /maintenance projects.

2.1.3 Forces

Automated Software Engineering tools are not understood by the professionals. CASE tools require specialized training. There is also a heavy upfront cost involved before making the purchase. People may lose their jobs if their work was assigned to automated tools**.** Automated tools are deployed scantily even if they are available in plenty.

2.1.4 Solution

A survey with the following questions was sent to 70 Software Professionals during Jan 2014. The goal of this survey is to estimate and document clearly the problems associated with the usage of tools by Software Engineers. There were totally 13 responses received. The survey was tracked over a period of 48 hours. This size (13) of the respondants is roughly the size of one or two project teams in Software Development companies. A summary of the questions/responses are as follows.

Q1. Which of the following best describes your IT job role?

Database Specialist	7.69%
Hardware Specialist	7.69%
Networking Specialist	23.08%
Programming Specialist	53.85%
Software Specialist	30.77%
Technical Writer	15.38%
Web Specialist	23.08%
Total Respondents:	13

Q2. Have you worked on any large development project (> 6 months)

Yes 84.62%

No 15.38%

Total Respondents: 13

Q3 Have you worked on a medium/small development project (small < 1 month, medium less than 6 months)

Small Development Project	0%
Medium Sized Development Project	7.69%
Both	92.31%
Total Respondents	13

Q4. If Yes to 2 or 3, what were the problems you encountered?

Too many Bugs.	50%
My work was not up to the same standard as my co-workers.	8.33%
My co-workers work was not up to the same standard as my work.	8.33%
The team had communication problems.	41.67%
There were problems with the management.	41.67%
There was fear of job loss on failed delivery.	58.33%
The methods adopted were too manual.	66.67%
Total Respondents:	12

Q5. Did you use any tools for up gradation?

Yes.	46.15%
No.	30.77%
Not Aware of any tools pertaining to Upgradation but aware of other tools.	0%
Not aware of any tools pertaining to any software development/maintenance or other activities during the SDLC.	0%
Aware of such tools, but do not use them.	15.38%
Have not worked on any up gradation project.	15.38%
Total Respondents:	13

Q6. Did you use any tools for debugging?

Yes.	69.23%
Not aware of any tools for debugging.	0%
Not aware of any tools for any other software activities.	0%
Aware of such tools, but do not use them during my project work.	30.77%
Total Respondents:	13

Q7. Have you used any tools for software development?

Yes, Have used them.	69.23%
Not aware of any tools.	0%
Aware , but do not use them.	30.77%
Total Respondents:	13

Q8. Do you use tools for configuration management?

Yes.	76.92%
Not aware.	15.38%
Aware, but do not use them.	7.69%
Total Respondents	13

Q9.What best describes your experience with Computer Aided Software Engineering.

Not aware	15.38%
Aware of these tools, but do not actively use these tools.	38.46%
Am not convinced about the benefits of using these tools and so do not use them.	0%
Am not inclined to actively integrate tools into my project work.	15.38%
I just follow the trend, if every other developer uses tools, then I will also use them.	30.77
Total Respondents:	13

Q10. In Summary, In your opinion using Tools for automating software development processes such as development, Coding, Debugging, Configuration Management will

Will not help my project work.	0%
Will help improve productivity.	69.23%
Will add to my misery.	7.69%
Will use these tools, if I am aware of the benefits.	46.15%
Will use these tools, If I am provided training.	23.08%
Will use these tools, if I all of my colleagues use them,	7.69%
Total Respondents:	13

The results of the survey can be broadly summarized as

1. The programmers took part in software projects of all sizes. They had ample opportunity to integrate automated tools into any of the life cycles.
2. A majority of the programmers were aware that such tools exist, but only in a few cases they were not aware of these tools.
3. Nearly 40% of those surveyed were aware of these tools, but were reluctant to use it.
4. Nearly 70% of the respondents' felt that these tools would improve productivity
5. Nearly 23% of the respondents' felt that they would use these tools if they were trained.
6. About 16% were not inclined to use these tools

Not much has changed since the mid 90s when CASE tools were first conceptualized and deployed. There have been some surveys done during the mid 90s , but the results about the reluctance or active usage still remains the same.

The survey successfully proved the intuitive feeling about the reluctance of software engineers to use such Tools. A large percentage feels that on being provided training, they would start using tools.

2.1.5 Resulting Context.

The survey is equivalent to a project manager trying to understand his own team members' inhibition to use Tools. So there is a problem on hands. The next phase calls for making those reluctant to experience the benefits/productivity gains of using tools. In case the survey does indicate that the existing professionals are not reluctant then there is no further action required to be taken.

2.2 Pattern Name: Getting Nowhere As Efficiently as Possible.

2.2.1 Problem: Queries are database objects that retrieve data from one or more tables as per some criterion specified by a user. Queries are based on Relational Algebra and use SQL in its declarative form. Queries use Joins, where, having and group by clauses. Sometimes these queries do not retrieve any data after execution and return a "NOT FOUND" status indicating that no valid sub set of data as per the select criterion has been found. Strangely, however when a NOTFOUND condition occurs, the query processor expends the same amount of resources as when a Query returns a valid sub set of data.

2.2.2 Context

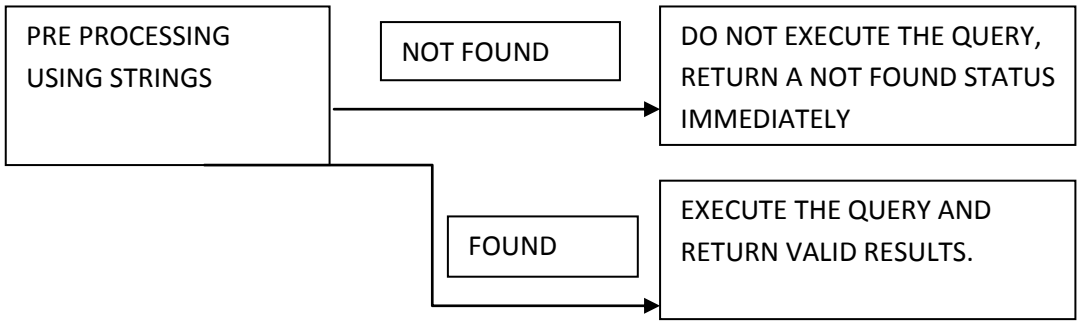
Queries and other search algorithms.

2.2.3 Forces

The query parser cannot anticipate that a NOTFOUND occurs and so expends energy searching for data until a NOTFOUND case finally occurs. Ideally this energy should be 0. Also humans have a problem expressing search criterion (which gets translated as joins and predicates) when NOTFOUND should not occur. Neither the human nor the query parser can anticipate which portion of the query (Predicate or Join) is failing and so expends energy until the failure occurs.

2.2.4 Solution

The solution is to introduce a pre processing step which will determine quickly if the query is going to fail.



Here the pre processing steps are explained further. The pre processing step will hold a snap shot of relevant data in the form of strings. The conditions imposed in the query in the form of joins and predicates will be determined quickly.

The declarative version of a query in SQL is listed below

Select a.col1, b.col2 from

Table 1 a, Table 2 b where

a.id = b.col3 and

a.col1 > 5 and b.col4 < 7 and b.col4 = “YES”.

Here failure can occur when a.id <> b.col3 or a.col1 is not greater than 5 or b.col4 is not less than 7 or when b.col4 is not equal to “YES”.

When the actual query is executed after the syntax of the query is checked by the parser, the query optimizer computes different execution plans. The plan with the lowest cost of query processing is chosen by the optimizer. However, as has been discussed earlier if the query is doomed the energy expended is as much as the case when the query returns valid data.

By introducing a pre processing step, the validity of the “where” and the “join” criteria is quickly determined.

For example all the row values in the column a.id are concatenated into a string “0010002003004005006” (for example), all the row values in the column b.col3 are also concatenated into by a string “433221444666” (example). The two strings are concatenated and any repeating pattern of length =3 is checked using standard string pattern matching algorithms. If no pattern exists then it can be quickly determined that the query will fail when it is finally executed or a.id <> b.col3.

Similarly the predicate b.col4 = “YES” can be evaluated quickly by concatenating all row values in column b.col4 into a string and also by concatenating the string “YES”. In case the pattern does not repeat then it can be inferred quickly that b.col4 <> “YES”.

Join conditions are similar to equality conditions and can be pre processed in the same way as described in the paragraph above.

The algorithmic complexity in the examples described in the previous two paragraphs is the complexity of the best known string pattern matching algorithm.

In the case of the expression “ a.col1 > 5”, the data in a.col5 is concatenated into a string and is sorted. The last element in the sorted list is determined and the truth of the predicate a.col1 > 5 is quickly determined using an algorithm which is as complex as the best known sorting algorithm.

In summary equality conditions can be quickly determined to be valid using string pattern matching algorithms. Inequalility conditions (<,>) can be quickly validated using sorting algorithms.

However, the pre processing step has to be beneficial. It should not be more expensive than the cost the actual query when it encounters a "NOTFOUND" case. This is illustrated in the Table 1 below.

As different query optimizers function differently, the solution presented above should be adopted in cases where cost of a doomed query is greater than the cost of pre processing (cases Columns (A,D), (B,D) Table 1).

In the cases where cost of a doomed query is less than the cost of pre processing (cases Columns(C,D) Table 2) , the pre processing step is not effective.

Table 1 : Cost of Query Vs Pre Processing

Rows Affected	**Cost of a Doomed Query(Steep Increase per row) Column A**	**Cost of Doomed Query(Moderate Increase per row) Column B**	**Cost of a Doomed Query(Low Increase per row) Column C**	**Cost of Pre Processing Column D**
0	10	10	10	5
100	100	50	15	50
1000	1000	500	20	100
10000	10000	5000	25	250
100000	100000	50000	30	500

Resulting Context

The resulting context will reduce the energy expended when a query fails by pre processing the data using string search, matching and sorting algorithms.

3.0 Referances:

[1] Gerard Meszaros and Jim Doble. 1997. A pattern language for pattern writing. In *Pattern languages of program design 3*, Robert C. Martin, Dirk Riehle, and Frank Buschmann (Eds.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA 529-574.

[2] Surajit Chaudhuri. 1998. An overview of query optimization in relational systems. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems* (PODS '98). ACM, New York, NY, USA, 34-43. DOI=10.1145/275487.275492 http://doi.acm.org/10.1145/275487.275492

[3] Donald E. Knuth. 1998. *The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[4] Computer Aided Software Engineering. In: *FFIEC IT Examination Handbook InfoBase*. Retrieved 3 Mar 2012.

[5] Software Engineering: Tools, Principles and Techniques by Sangeeta Sabharwal, Umesh Publications

4.0 Acknowledgements:

The author wishes to thank Professor K.V.Dinesha, IIIT-Bangalore for conducting a Introductory Workshop for GURUPLOP at IIITB. The author also wishes to thank Professor David West for shepherding the initial drafts of this paper.

5.0 About the Author.

The author obtained a MS by research degree in Information Technology from the International Institute of Information Technology, Bangalore in the year 2013. Prior to this, the author obtained a Mechanical Engineering degree from the Indian Institute of Technology, Chennai in the year 1990 a MS by Research degree in Industrial Systems Engineering 1992 from the University of Regina in Saskatchewan, Canada. The author has published in the India Software Engineering Conference 2012, “International conference on Computers and Industrial Eng 1991, Orlando, Florida”, International Journal of advanced Manufacturing Technology 1996. The author was also awarded the Governor General of Canada’s academic medal (Gold) in 1992.