TEMPLATE BASED FRAMEWORK FOR RAPID FAST DEVELOPMENT OF ENTERPRISE APPLICATIONS

Srinivasa Varma Mantena[1], G. P. Saradhi Varma[2], Syamala Rao P[3]

[1]*Chief Technology Officer, ModeFinServer Pvt. Ltd, India*
[2]*Director P.G. Courses, S.R.K.R. Engineering College, India*
[3]*Associate Professor, Shri Vishnu Engineering College for Women, India*

Abstract

High Quality Deliverables in Shortest duration is the key to win any future software business opportunities. Every Software Organization wants to achieve this but suffers problems like Delayed deliverables, Customer complaints, Poor Quality deliverables, Poor Turnaround time, Rework, Lack of time for reviews, Maintenance nightmares, resource dependencies, complex technology frameworks resulting slow learning curve, and challenges dealing with resources. Automation is the key to many of the problems; however, regular automation cannot address the issues of most commercial or enterprise applications at global level. Needs of every application is different making automation tough. There are many frameworks and custom standards available setting the expectations for development team, but considering complexity, it is practically impossible to ensure consistency of implementing the set expectations considering typical human ignorance tendencies.

Other alternative for Software Organizations is to make use of Specific Tools available from market. Most Automation tools are expensive and are catered design only specific category of problems. On the other side, Organizations making use of automation tools from market end up getting into a Vendor Lock for upgrades, maintenance, highly expensive consultant costs and support. This paper provides a Framework which can significantly address these challenges of Software Organizations. Irrelevant of technology area, most applications are database driven. Every operation that gets done on UI or through a service will have to be reflected in database. Considering specific needs of application or organization standards, an initial working flow (UI, Controller, Service, DAO) will be prepared by an expert for all levels of the framework to be used. Once the working flow is prepared, a template will be generated based on that. Template will be applied for all tables in database.

Keywords: *Automation, Development Framework, Productivity Improvement, Template based development, Server Side Development*

----------***----------

1. INTRODUCTION

Every Software Organization strives to achieve best quality product. However, due to cost constraints, many service based software organizations engage fresh engineers and few senior resources. Due to lack sufficient skill, lot of resources delivers poor quality code and to address it organizations engage additional team members for reviews and testing.

Although reviews and testing helps in closing the gaps to some extent, it cannot cover entire code base and compromise will happen after certain extent to meet the deliverable deadlines. On the other side, it is challenging for Software Organizations to meet the demands of employees with potential. As a result, in the name of Cost Saving, organizations are increasing cost of project either to Customer or to Software Organization.

The more we add team members the more complex the system becomes. For every enhancement, defect fix or platform upgrade, the risk of existing deliverable breakage is significant high and results in nightmares.

2. PROBLEM SCENARIO

Automation is the solution for many of the challenges being faced by Software Organizations. However, 100% automation is not practically possible considering the dynamic nature of customer requirements and market. The core Business Logic should be customizable.

There are various configurable solutions available in the market to reduce development effort and achieving high quality deliverables.

Spring, Hibernate in Java World are few frameworks which are meant to achieve this goal. These solutions are adopted by many Software Organizations. The challenges with these frameworks are dependency on these libraries and lack of

control on these libraries in case of any issue. On other side, these solutions are not easy to learn by Software Engineers.

Popular Integrated Development Environment tools like Eclipse, NetBeans generates some code automatically but it is very limited.

Other options that organizations have are, procurement of Specialized Domain Products which act as platform and on which customizations can be made. Majority of times, these products are very expensive and can cause Vendor Lock to the procuring Organization.

3. PROPOSED FRAMEWORK

All database operations by applications are usually limited to 4 statements - Insert, Update, Select and Delete. Other operations like Create, Drop etc. are not typically required at application level. Any Business Application will only do these 4 operations on database. Heart for any software system will be the schema of the database.

Generation of Code in an automated way based on Schema for all layers - Presentation, Controller, Service, Data Access layer etc. can address majority of issues highlighted. This approach supports making use of any libraries in between as required by the organization.

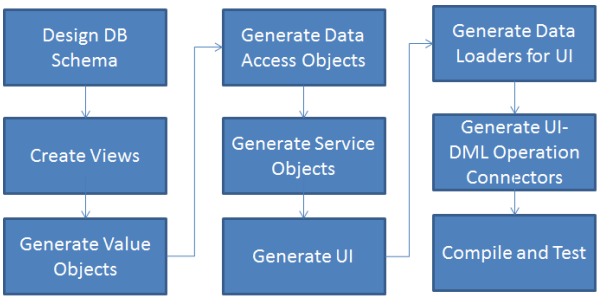

Fig. 1 Flow of Activities for Automated Code Generation

For this approach, DB schema is the input. However big the database schema is, by making use of Database Meta Data methodologies, entire DB schema can be read and code can be generated based on Database Schema. To address join scenarios among relationships in database schema, Views can be used. For views, only select options are allowed. Value Objects are getter, setter methods for each field based on type. Value Objects can be equipped with print content of Value Object with names and values for debug facilities and logging.

Once the Value Objects are generated, generation of associated Data Access Object code for each relationship can be achieved. Data Access Object code can have as many variations of select, insert, update, delete operations as required by the target system. For select statement, various possible approaches can be fetch record by id, fetch records matching input query, fetch records marching certain fields etc.

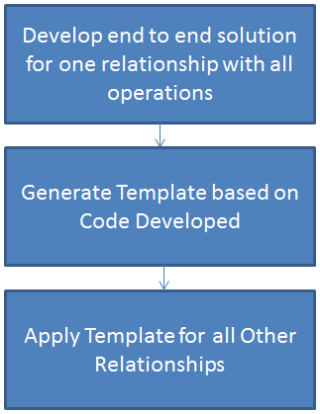

Fig. 2 Flow of Activities for Automated Code Generation

Service layer acts as common area for all interfaces. Service layer objects can be generated once DAOs are ready. In the same manner UI, data loaders from UI can be generated.

Some tables may not require all operations by application. However, with today's speed of processors, existence of few extra lines of code will not harm. Support for all operations can address majority of future requirements by default.

For this solution, the key is to make one relationship work with all operations. This should be done by a senior resource that is skilled in the area. From User Interface till Database Operations all operations should be implemented in full as required by Customer. The review can be done on this alone. Once this is made to fully working, convert it as a template. i.e. replace all Relationship names to #RELATIONSHIPNAME#, #FIELDNAME#, #FIELDTYPE# etc. Comments can be added to template. Once template is ready, using Database Meta data, code for entire system can be generated.

Comments can be accommodated in the template. Any operation that is not required on UI, can simply be commented in separate package. Deletion or commenting of unnecessary operations is much easier than developing them.

```
public boolean
insert$RECORDCLASSNAME$($RECORDCLASSNAME$
record) throws Exception {
        try {
        logger.debug(\"insert$RECORDCLASSNAME$:\" +
        record);

        String beginMesssage =
        logger.generateFunctionBeginTimerMessage(\"insert
        $RECORDCLASSNAME$\", null);
```

```
        $DAOCLASSNAME$ dao = new
        $DAOCLASSNAME$();
        boolean result =
        dao.insert$RECORDCLASSNAME$(record);
        logger.debug(\"insert$RECORDCLASSNAME$:Res
        ult:\" + result);
        logger.generateFunctionEndTimerMessage(beginMes
        ssage);
        return result;
        } catch(Exception exception) {
        logger.error(\"insert$RECORDCLASSNAME$\" +
        getStackTrace(exception));
        throw exception;
        }
}
```

CONCLUSIONS

This solution is not limited to any technology or area. This can be applied to all applications during development stage. This solution is useful for Service Based and Product Based organizations.

Benefits of the solution include

- No Vendor Lock
- No License issues
- No New technologies to be learn
- Full Control on entire code base
- Supports any library
- Supports all future enhancements
- Technology Independent
- Rapid Fast Development
- Easy Maintenance
- Significant Improvement in turnaround time
- Significant reduction in Resource Dependency
- High Quality Deliverable

This solution can be further extended by making it available on the web for dynamic generation of code on net. Schema should be given as input.

RESULT ANALYSIS

Solution proposed has been applied on 3 real time enterprise applications and results were captured in the graph below. Amount of Custom code is very less compared to overall size of project. Auto Generated code is more reliable in quality as template is prepared by expert resource. Organizations can pay very close focus to Core Business Logic which contributed only from 7% - 20% of overall size of project. Auto Generated code is directly proportional to saving for the organization.

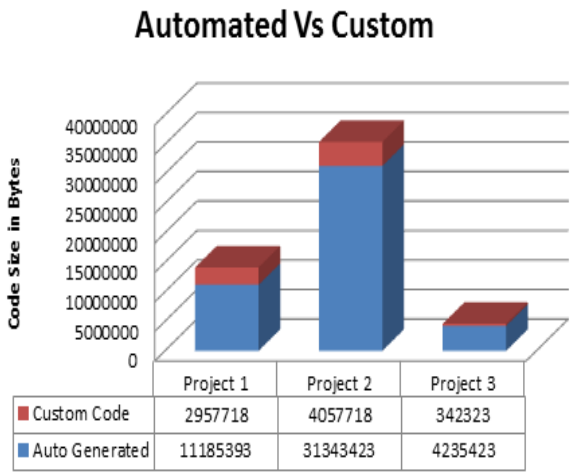

Fig. 3 Comparison of Automated Vs Custom Code

REFERENCES

[1]. Xuhao Chen, Zhong Zheng, Li Shen and Wei Chen, "GSM: An Efficient Code Generation Algorithm for Dynamic Binary Translator", IEEE Conference Publication Dec 2011

[2]. Cheng Wang, Wei-Yu Chen, Youfeng Wu, Saha, "Code Generation and Optimization for Transactional Memory Constructs in an Unmanaged Language", IEEE Conference Publication Mar 2007

[3]. Qing Yi, "Automated Programmable Control and Parameterization of Compiler Optimizations", Code Generation and Optimization, 2011 9th Annual IEEE/ACM International Symposium

[4]. J. Ramanujam, P. Sadayappan, "GPU Programming Models, Optimizations and Tuning", Code Generation and Optimization, 2011 9th Annual IEEE/ACM International Symposium

[5]. Ian Gartley, Marius Pirvu, Vijay Sundaresan, and Nikola Grcevski, "Experiences in Designing a Robust and Scalable Interpreter Profiling Framework", IEEE Symposium on Code Generation and Optimization 2013