EliXir: a framework for Building e-business applications

Ammar Joukhadar
Information Technology Faculty, Damascus University
ammarj@scs-net.org

Abstract:

This paper presents an integrated framework that allows us to build 3 Tiers application in just 2 steps Specify and Deploy. This framework merges the power of Model Driven Architecture (MDA), the simplicity of Business Process Management Notation (BPMN), the breadth of the web and the flexibility of Service Oriented Architecture (SOA). Applications are specified using UML and BPMN and can be deployed on any platform.

Actually eliXir can be deployed on J2EE platform, using MySql or Oracle as database, using Jboss or BEA weblogic as application server and windows or Linux as operating system.

1 Introduction

E-business systems are becoming more and more complicated because they involve a large number of functional and non functional constraints. USA spends more than 250 billion USD per year for more than 175 thousand projects on which several million of people work. But 30% of these projects fail before being used and 50% of these projects cost twice the initial budget [1][2][3].

There are three major factors that make e-business system very difficult to build:

1- Requirements become more and more complicated such as supporting a large number of users distributed over several locations, supporting different types of channels such as http, mail, fax, SMS, mobile, web service, etc, delivering in months rather than years, supporting different types of non functional requirement such as scalability, availability, security, testability maintainability, reliability fail over, fault tolerance, performance, safety, portability, user friendliness, traceability

2- Meeting those requirements needs advanced technologies, such as using technical frameworks [6] [7], applying some standard design patterns [8] [9], decomposing code into a set of modules, applying modern design patterns (3 Tiers, AOP, SOA, MDA[5], MVC, IoC, ORM, DAO, etc) [10] [11], using generic programming techniques, or using code generators.

3- These technologies increase the complexity of the system, and depending on programmers experience this may reduce the code quality dramatically.

Reducing program complexity requires separating the different aspects of the program which allows programmers to concentrate on one aspect at once. This can be made using frameworks. Current frameworks have some limitations:

- They do not generate a complete program, but just a part such as the view, the model, or the storage.
- They are not developed to be integrated together. This integration is a burden that the user has to bear.
- They are difficult to configure (Model, View, Security, Mapping)
- They free the programmer from writing some code, but do not free him from understanding the sophisticated infrastructure.

Our solution: eliXir framework

EliXir is an MDA framework based on BPMN (http://www.bpmn.org/). It takes as input class diagrams (CD) and Business Process diagrams (BPD), and generates a complete 3 tier web based application. Thanks to our eliXir Task Definition Language (TDL), eliXir can work in just 2 steps: *Specify and Deploy*.

2 Elixir main features

- Based on UML diagrams, eliXir stereotypes, and eliXir semantics, eliXir infers different types of metadata such as Security metadata, GUI metadata, Interaction metadata, and Storage metadata,
- Based on its rich Metadata, eliXir can generate Code or Execution Languages for different platforms.
- It frees administrators from the hard configuration task by generating configuration from the UML itself based on eliXir Semantics. This configuration includes Security, Reporting, GUI, and ORM.
- It separates business logic from techniques.
- It separates non functional requirements from each other.
- It frees programmers from understanding the underlying architecture.
- It frees programmers from having to modify any generated code; instead it allows him to add a kind of business plug-ins.
- It frees programmers from having to master sophisticated non functional requirements which are added by eliXir itself.

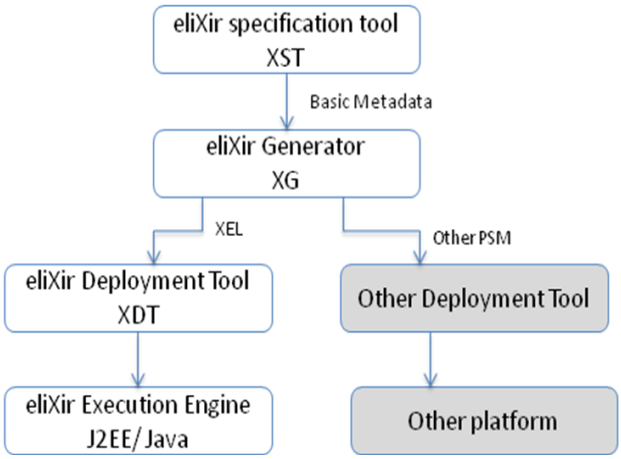
3 EliXir main components

Elixir has 4 main components: eliXir Specification Tool XST, eliXir Generator XG, eliXir Deployment Tool XDT and eliXir Execution Engine XEE :

- XST: provided with XSE (eliXir Semantic Engine), XST assists users to specify UML (CD and BPMD).
- XG: allows to generate metadata from UML, and to generate execution language from metadata.
- XDT: allows user to deploy his application and to integrate it with other applications in the environment (business applications, data base, devices).
- XEE: allows executing the deployed application.

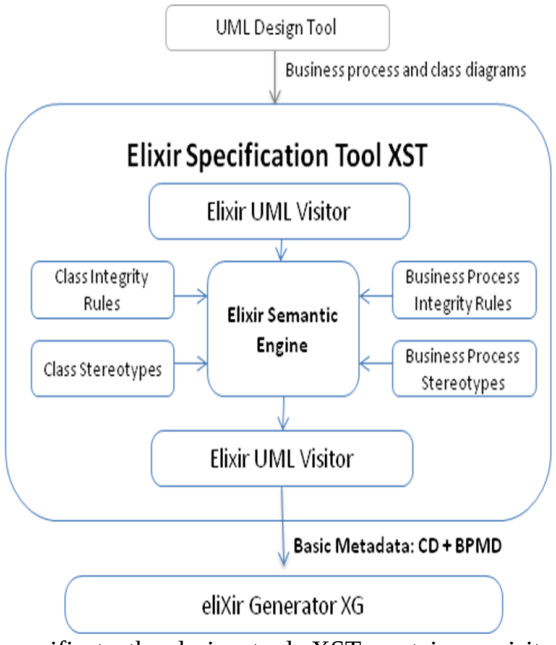
Elixir has already been used to develop e-business applications such as e-government, web based ERP and billing systems.

4 EliXir architecture



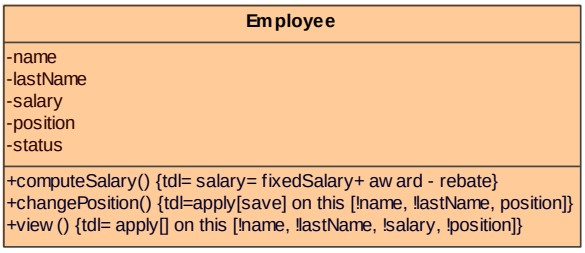
5 EliXir Specification Tool architecture

eliXir Specification Tool XST takes as input UML class diagrams and business process diagrams in a format

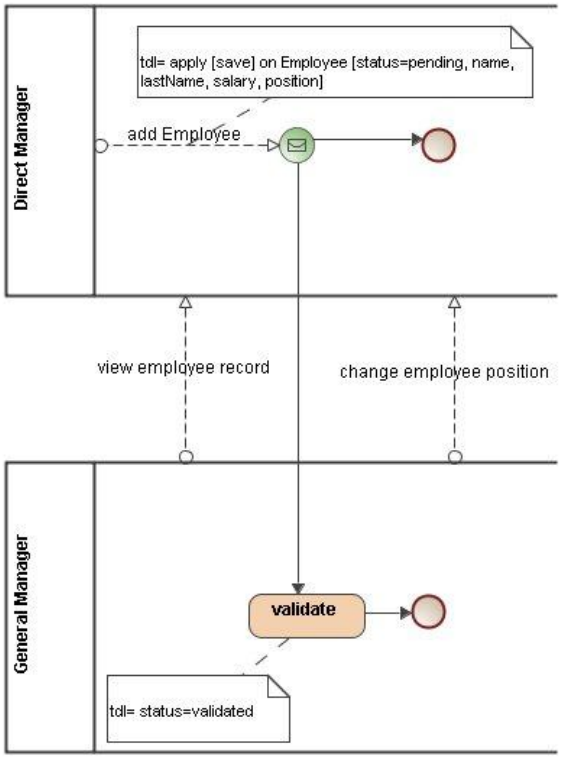


specific to the design tool. XST contains a visitor that visits the input diagrams and passes the result to elixir Semantic Engine which uses a variety of integrity rules (e.g. class integrity rules, business process integrity rules... etc.) and elixir Stereotypes to convert the diagrams to eliXir specific format which will be input to the Metadata Generator. Integrity rules used by the

semantic engine include business process integrity rules such as detecting overlapping cycles in addition to class diagrams integrity rules which include rules such as detecting fields with unspecified type. The Metadata Generator then generates basic metadata which is input to eliXir Generator. Basic Metadata consists of class diagrams and business process management diagrams metadata in addition to GUI and external ports metadata. Those types of metadata are linked together using Data Object in business process diagrams and TDL (Task Definition Language). TDL is a set of formal expressions which are added to the business process diagrams and class diagrams by the user in order to link them formally. TDL extends the method concept to include not only side effects but also user interaction. Thus, TDL can be of two kinds: an expression that has a side effect and a GUI TDL which defines interaction with the user. In order to clarify the role of TDL, we will provide the following example.



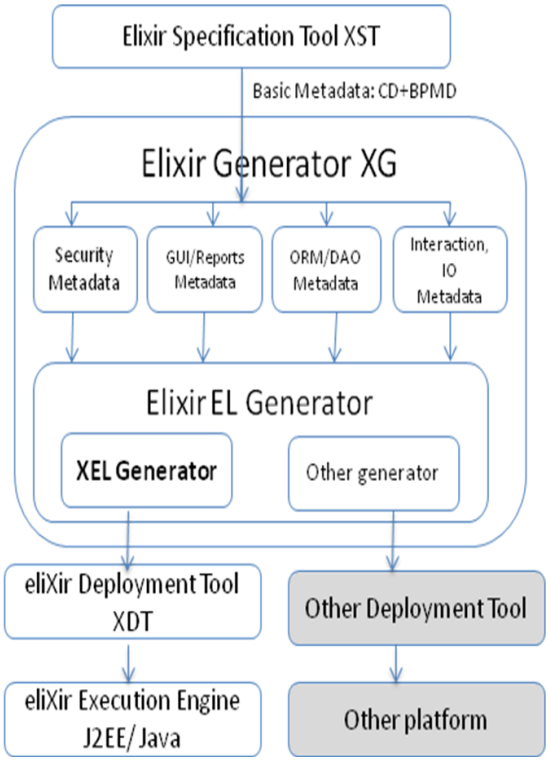
We have a class Employee which have the following attributes: name, lastName, salary and position. This class has the following operations: computeSalary, changePosition, and view. ComputeSalary operation is implemented using a TDL expression that calculates the salary of the employee. Whereas changePosition and view are operations that need user interaction so they are implemented using GUI TDL that creates a GUI through which the user can change the position of the employee (in case of changePosition operation) and view his attributes (in case of view operation). To define which user has the right to what and when, we need to use a business process diagram as follow.



The above diagram shows that the direct manager has the right to add new employees but this operation has to be validate by general manager. Only the general manager can change the position of an employee.

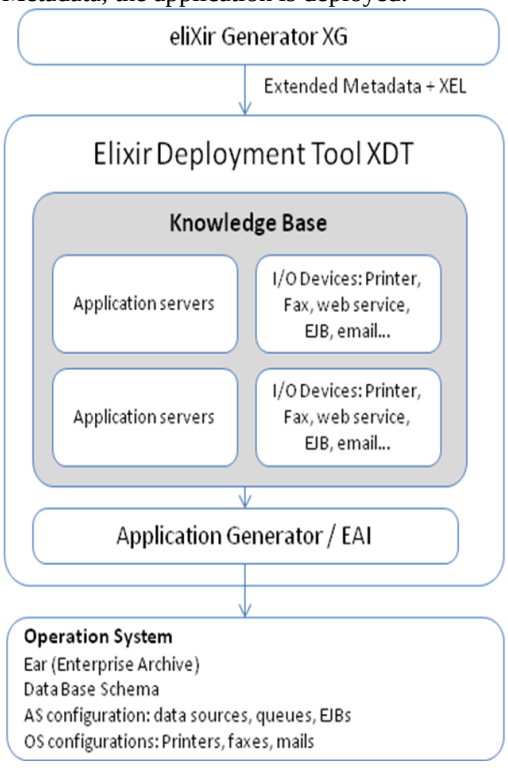
6 EliXir Generator

EliXir Generator takes as input Basic Metadata generated by eliXir Specification tool and infers semantically high level metadata called Extended Metadata which consists of security metadata, GUI/ Reports meta data, ORM/DAO metadata and interaction/ IO metadata. eliXir Generator contains eliXir Execution Language Generator (eliXir EL Generator) which generates eliXir specific execution language XEL or other PSM (Platform Specific Model) . The resultant execution language is then passed to the deployment tool which can be eliXir deployment tool or any other deployment tool.



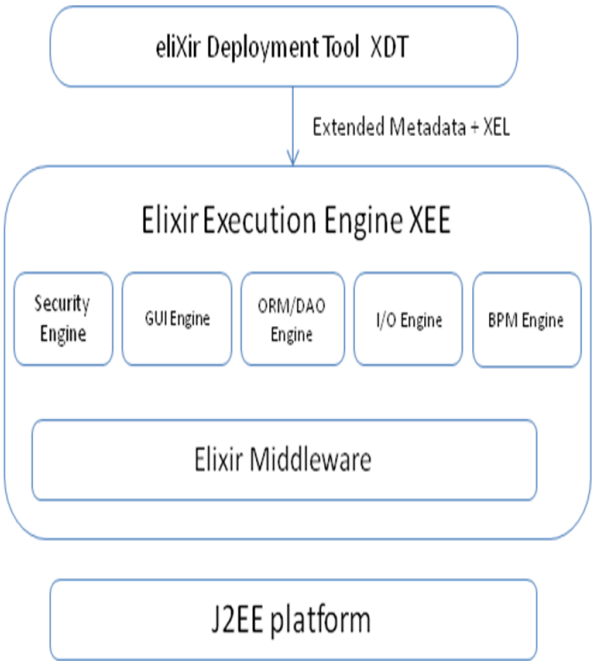
7 EliXir Deployment Tool architecture

eliXir deployment tool XDT takes as input eliXir Execution Language (XEL) and eliXir Extended Metadata generated by the XEL generator. XDT has a knowledge base that contains knowledge acquired from the user about the operating systems, databases, application servers and I/O devices. Based on this knowledge and on the input XEL and Extended Metadata, the application is deployed.



8 EliXir Execution Engine Architecture

Elixir Execution Engine XEE consists of a number of engines such as Security Engine, GUI engine, BPM Engine. XEE takes as input elixir Execution Language XEL generated by eliXir XEL generator and uses Extended Metadata to execute the deployed application. XEE contains a middle ware which is used to interact with other applications.



9 Conclusion and Future Work

In this paper we have presented eliXir framework. This framework allows us to build a complete 3 tiers applications in just two steps specify & deploy. eliXir can generate an extended meta data basing on some business rules and on our Task Definition Language. This extended meta data includes data object model, security, presentation, interaction, storage. The Task Definition Language allows us to express an action as well as an interaction with the user. This is why eliXir can build a complete application not only one tier. A future work aims to infer the data object from the business process and TDL, which allows us to simply the task of business specification.

References:

[1] MDA Distilled: Principles of Model-Driven Architecture, Stephen J. Mellor, Kendall Scott, Axel Uhl, Dirk Weise, Addison Wesley, March 03, 2004. ISBN 0-201-78891-8

[2] http://www.ovum.com

[3] http://www.standishgroup.com

[4] UML source, http:// gentleware.com/ fileadmin/media/synergy/Course/index.htm.
[5] “The Essence of Model Driven Architecture”, Wim Bast – www.jaxmagazine.com/itr/online_artikel/psecom,id,548, nodeid,147.html
[6] RICK HIGHTOWER, “An Introduction to Spring”, Java Developer's Journal, 2005.
[7] Casey Kochmer, “Introduction to Struts”, JSP Insider magazine, April 2001.
[8] Peter Varhol, “Applying the MVC Design Pattern Using Struts”, javapro magazine, may 2002
[9] Firesmith, Donald G. , Deugo, Dwight , “Applying Design Patterns in Java," in Java Gems book, Cambridge University Press, 1998.
[10] Karl Lieberherr, Doug Orleans, and Johan Ovlinger, “Aspect-Oriented Programming with Adaptive Methods”, NU-CCS-2001-02, 15 pages.
[11] Martin Fowler, “Inversion of Control Containers and the Dependency Injection pattern”, http://martinfowler.com/articles/injection.html