Decision Engine Platform

BITS ZG628T: Dissertation

by

Pramod Kumar N

2014HT13292

Dissertation work carried out at

Coextrix Technologies Pvt. Ltd., Bengaluru

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)

November 2016

Decision Engine Platform

BITS ZG628T: Dissertation

by

Pramod Kumar N

2014HT13292

Dissertation work carried out at

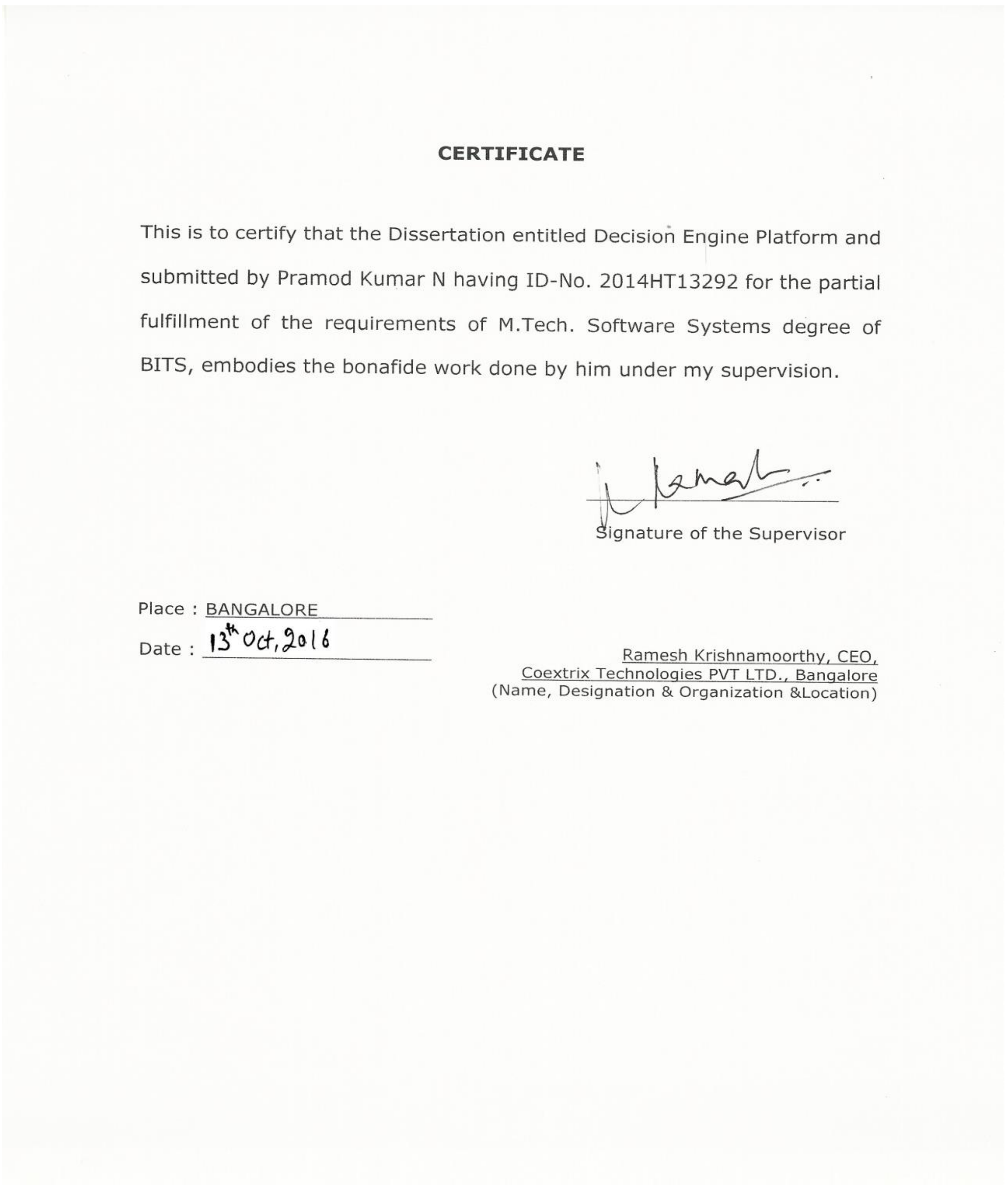
Coextrix Technologies Pvt. Ltd., Bengaluru

Submitted in partial fulfillment of M.Tech. Software Systems degree programme

Under the Supervision of
Mr.Ramesh Krishnamoorthy, CEO,
Coextrix Technologies Pvt. Ltd., Bengaluru

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)

November 2016



CERTIFICATE

This is to certify that the Dissertation entitled Decision Engine Platform and submitted by Pramod Kumar N having ID-No. 2014HT13292 for the partial fulfillment of the requirements of M.Tech. Software Systems degree of BITS, embodies the bonafide work done by him under my supervision.

[illegible]

Signature of the Supervisor

Place : BANGALORE

Date : 13th Oct, 2016

Ramesh Krishnamoorthy, CEO,
Coextrix Technologies PVT LTD., Bangalore
(Name, Designation & Organization &Location)

Birla Institute of Technology & Science, Pilani

Work-Integrated Learning Programmes Division

First Semester 2016-2017

BITS ZG628T: Dissertation

ABSTRACT

BITS ID No.	**: 2014HT13292**
NAME OF THE STUDENT	**: Pramod Kumar N**
EMAIL ADDRESS	**: pramod974@gmail.com**
STUDENT'S EMPLOYING ORGANIZATION & LOCATION	**: Coextrix Technologies, Bengaluru**
SUPERVISOR'S NAME	**: Ramesh Krishnamoorthy**
SUPERVISOR'S EMPLOYING ORGANIZATION & LOCATION	**: Coextrix Technologies, Bengaluru**
SUPERVISOR'S EMAIL ADDRESS	**: ramesh.krishnamoorthy@coextrix.com**
DISSERTATION TITLE	**: Decision Engine Platform**

Abstract

In modern day applications that provide more visibility into data irrespective of domain, require collecting data from several different sources. Collected data needs to be consolidated, normalized and then presented through an application. There is a lack of correctness in the data aggregated owing to the nature of issues with the sources of data itself. Since the original data sources are doing nothing to clean up the data they provide or generally there is no standardization maintained across domains in context. Several applications are designed to clean up the data coming through various sources and allow customers to perform complex analytics on top of the data. Transforming and reconciliation process is extensively complex, but is required to create higher accuracy in the data. The transformation and reconciliation process comprises of multiple rules, interactions with multiple components in order to achieve the accuracy. In popular existing open source technologies the way the rules are written and configured is stereotype and hard to maintain owing to the quantity of rules modelled and its ability to scale in future.

There are no matured open source decision engines built using python which satisfies applications current needs, where there is an end to end feature to model and maintain rules which could interact with multiple sources and components. Accurate data has been always sought for and is the next gen thing to get custom insights irrespective of domain. The underlying need to build a configurable and maintainable platform is highly compelling, where a Decision engine facilitates connecting to various sources and components in order to write domain specific configurable rules so as to achieve valuable decision in limited turnaround time. The Goal is to build, deploy and scale the existing stereotype rules/models to a more maintainable and configurable Decision engine and also provide an easy interface and framework to model rules on the fly build, test and deploy it without any manual intervention.

Broad Academic Area of Work: **Software Architecture**

Key words: **Rules, Engine, Decision, Python, Platform, Framework, Configurable**

[illegible]

Signature of the Student

Name: PRAMOD KUMAR. N.

Date: 13th Oct, 2016

Place: BANGALORE

[illegible]

Signature of the Supervisor

Name: Ramesh Krishnamoorthy

Date: 13th Oct, 2016

Place: BANGALORE

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of dissertation would be incomplete without mentioning the people who made it possible, whose constant guidance and encouragement crowns all the efforts with success.

I express sincere gratitude to Mr.Ramesh Krishnamoorthy, CEO, Coextrix for being kind enough to provide me an opportunity to work on a project in this organization and also being my mentor.

I am thankful to Mr. Ganesh Guruswamy, CTO, Coextrix for providing valuable ideas, suggestions, facilities, co-operation and his encouragement at all moments of my approach towards tackling the problem.

Last but not the least, I would like to express my love and gratitude to my beloved family and friends, for their understanding & motivation, through the duration of this project.

Table of Contents

Table of Figures

GLOSSARY

REST – Representational State Transfer

API – Application Program Interface

CSV – Comma Separated File

MSEXCEL – Microsoft Excel

JSON – Java Script Object Notation

IDE – Integrated Development Environment

Jenkins – Continuous Integration Tool

Chapter 1: Introduction

1.1 Data Sources

Data has been most sought for in the modern era to gain deeper and meaningful insights into ones business. The insight from data is required irrespective of which domain one is working on, in order to gain such insight data has to be curated from multiple sources. The data source [1] is can be typically a connection set up to a carious computer databases running as a server or it could be as simple as a file (CSV, text, MSEXCEL, etc.) [2] or it could just be a stream of data [3] coming in via API or live feed.

1.2 Problems and Challenges with Data

The volume, correctness and consistency pose a major challenge with the data collected from various sources has to be processed before they can be analysed. Some businesses require domain specific incorporations as well to meet the data quality issues. Some of the common data related problems and challenges are as follows.

1. Poor data quality such as noisy data, dirty data, missing values, inexact values, inadequate data size and poor representation.

2. Integrating conflicting or redundant data from different sources and other forms like audio, video and images, geo data, text, social, numeric, etc.

3. Proliferation of security and privacy concerns by individuals, organisations and governments.

4. Unavailability of data or difficult access to data.

5. Efficiency and scalability issues to effectively extract the information from huge amount of data in databases, dealing with huge datasets that require distributed approaches, dealing with non-static, unbalanced and cost-sensitive data.

6. Constantly updating model to handle new incoming data, processing large, complex and unstructured data into a structured format.

1.3 Data Transformation and Reconciliation

Owing to multi natured problems with collected data, it has to be transformed and reconciled. Typically along with all standards process available for transformation and reconciliation, business use domain specific rules and knowledge to transform and reconcile the data so they are analysis ready.

Data transformation is the way of converting data from one format to another; generally this involves conversion of format of a source system into the required format of a new resultant system. The typical process involves converting documents, but data
Conversions at times involve the conversion of a program from one computer language to another in order for the program to run on a different platform. The usual reason for this data migration is the adoption of a new system that is completely different from the previous one. [4]

In real world, data transformation involves the use of a special program that can read the data’s original base language, determine which language the data that must be translated for it to be usable by the new program, and then move ahead to transform that data.

Data Transformation consists of two phases:

Data Mapping: This involves assignment of elements from the source system toward the destination to capture all transformations that commonly occur. This is not trivial when there are complex transformations like many to one or one to many rules for transformation that need to occur.

Code Generation: The creation of the actual transformation program. The resulting data map specification is consumed to create an executable to be executed on computer systems.

The Reconciliation process [5] consists of validating, computing, generating data mathematically so the integrity of the data in picture is validated to most correct extent from available assumptions and resource.

1.4 Software Architecture and Design Patterns

Software architecture [6] refers to the fundamental structure of structures, the field which help in creating such structures, and also documenting these structures. The structures created help in describing about the software system. Each structure consists of software elements, relations between them, and also properties of both put together as well, add to this with a touch of rationale for the introducing and configuring each element.

The architecture of any kind of software can be thought of as an analogy to the architecture of a building. The software architectures comprises of following activities.

Architecture supporting activities

Software architecture supporting activities are done during core software architecture activities. These activities help a software architect to do analysis, synthesis, evaluation and evolution. For example, architect curates knowledge then goes on to make decisions and then documents everything during the analysis phase.

Knowledge Management and Communication involves exploring and managing knowledge that is critical for designing software architecture. A software architect does do work all by himself. They get details such as functional and non-functional requirements then go on to design them in contexts, from all the intended stakeholders; and provide outputs to these stakeholders. Software architecture knowledge is generally a tacit and kept in the minds of stakeholders.

Design Reasoning and Decision Making involves evaluating design decisions. This activity is important to all three of the core software architecture activities. It consists of gathering and associating all decision in contexts after this it involves formulating design decision problems, which culminates in finding solution options and taking into account the ups and downs before making decisions. This process occurs at various levels of decision, one during evaluating significant architectural requirements and then during software architecture design decisions, and finally involves analysing, synthesising, and evaluating everything.

Documentation is key activity of making note of all the design is done during the software architecture process. The system design is emulated using several views that commonly include a static view depicting the code of the system, and then it consists of a dynamic view depicting the actions of the whole system during execution, and finally a deployment view depicting the way system is executed using the hardware.

Design Patterns [7] are set of standard solutions for commonly occurring problems and issues in regular software designing process. These patterns are generalized solutions and best practices which a programmer or a designer takes into account while solving the problem at hand.

Chapter 2: Motivation

2.1 Introduction

Every organization will have business to do a lot of different things, each of these business span across different domains and need rules to maintain their business generally the rules are very enormous in nature and are if there are no business rules it is very chaotic and the intended output is very poor in nature and results in more work.
Business organizations in order to tackle this rules use the help of skilled domain experts or go in leveraging specific automated software to suit their needs, sometimes there is both combination of software and human effort.
Few organizations use software to meet their needs as there are few tools available to help them automate their requirements. The major issues are when there is change in the business rules, making these changes come in to action is a very cumbersome task.

2.2 What are Business Rules?

A business rule [8] is a rule that defines or constrains some characteristic of business and always results in the final answer being either true or false. [6] Business rules goal is to assert business structure or either to control or influence the behaviour of the business. For instance, a business rule might state that if the price and volume petrol is right to be lifted from a terminal, another one would be to select a most good pick of supplier from list of preferred suppliers and supply schedules.

Business rule are usually informal in nature. Making note of the rules clearly and also ensuring that they don't overlap is an indispensable activity. When carefully steered, rules can be used to aid the organization in achieving goals, eliminate obstacles to market and increase growth, decrease expensive mistakes, foster communication, comply with legal requirements, and increase drastically customer loyalty.
All the business rules fall into any of the following four categories:

- Definitions:
  These define the actual domain specific definitions how data needs to act. These are generally the high level concepts which are fundamentals.
- Facts:
  These are atomic level calculations which define how the data has to be or how it has to be calculated or formulated
- Constraints:
  The constraints define how far the data can be or how they are quantitatively controlled.
- Derivations:
  These are derived from facts and generally the most common rule which interleave into the facts to form more complex formulations.

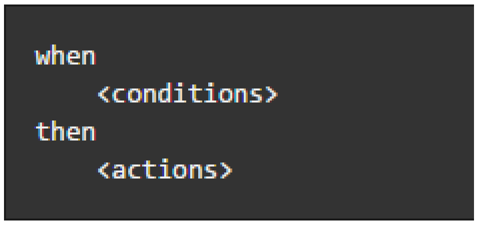
2.3 Problems associated with Business rules

The rules are generally chaotic though they leverage people and automation software there are a lot of problems associated with the Business rules. The most important problem is adding and changing business rules. There is always a major technical surgery required. There is a need for an application which provides consistent enforcement and constant even efficiency gain. The major technical surgery required for each change causes slow rate of change resulting in high costs.

Generally the rules are in the hands of the developer. Whatever the creation, modification, updates, deployment and testing is generally done as a whole process. So the bottom line of the problem is to find a solution which is more flexible and moves the change and modifiability overhead to a non-developer or a business consultant.

2.4 Rule Engine

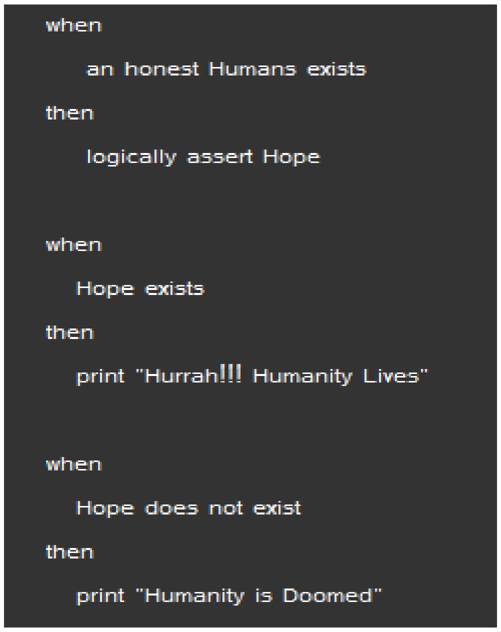
The concept of Rule Engine [9] is very ambiguous in nature. It could be any system that makes use of rules, in any form, in order to leverage upon it so that it can be applied to data in order to produce meaningful outcomes. The common instance of rule engine includes simple systems like that of a form validation and dynamic expression engines which work on a stream of data. Rules can be embedded in the application or maybe as a service. Rules are generally "stateful" and are always an integral part of an application.



```
when
    <conditions>
then
    <actions>
```

Figure1 : Rules Skeleton

The rule engine typically has a lot for rules similar to the skeleton shown in figure shown above, that are executed based upon a hierarchical structure defined by another entity / user. The end goal of the rule engine is to help produce meaningful outcomes. The main end goal of the rule engine is to provide a scalable environment to build rules and make use of them. The figure shows a typical inference with the help of a rule engine. The figure explains an outcome of either how one can take a decision of living of humanity or if humanity being doomed by taking a call, based on if honest human exists. The series of when and then conditions helps the rule engine to decide the outcome. The initial rule is to assert when honest humans exist. If the assertion is true then the rule engine will print a message "Hurrah!!! Humanity lives", if the assertion is false then the rule infers doom of the humanity by printing a message "Humanity is Doomed".



```
when
    an honest Humans exists
then
    logically assert Hope

when
   Hope exists
then
   print "Hurrah!!! Humanity Lives"

when
   Hope does not exist
then
   print "Humanity is Doomed"
```

Figure2 : Rules

2.5 Advantages of Rule Engine over if… then structure

2.5.1 Declarative Programming

Rule engines emphasis is on "What one has to do" not "How once can do it".
The major advantage of this by leveraging on rules makes it easy to give solutions to difficult problems and consequently have these solutions verified since rules are much simpler to read than code.

2.5.2 Logic and Data Separation

Your data is contained in domain objects; the logic is contained in the rules. This is fundamentally removing the OO coupling of data and logic together, which can be boon or bane depending on how one views it. The advantage is that the logic is easy to maintain as and when the changes are found in the future, since the logic is emulated as rules. This especially true advantage when the logic belongs to cross domain or even spans across multiple domains.

2.5.3 Speed and Scalability

The separation of data and logic to one’s domain object data helps in speed, scalability and also especially efficient when one has datasets that do not change entirely.

2.5.4 Centralization of Knowledge

Making use of rules, one can create a repository of knowledge in the form of rules which is executable. This makes it a single point of truth, for instance business policy rules are generally clearly readable, and this can also be used as documentation.

2.5.5 Tool Integration

Tools such as PyCharm [10] and in future maybe web based User Interfaces can be used to provide an easy way to edit and maintain rules and also get instant feedback, validation and content aid. Auditing and debugging tools are also incorporated as part of this. The rule when implemented on a platform can also connect to multiple sources thus making it a one stop solution for all the integration and consumption.

2.5.6 Explanation Facility

Platform effectively provides an explanation facility by providing log of the decisions taken by the rule engine. The log also includes reasons describing why the decisions were made, thus helping the end user formulate for rules and take control of the randomness.

2.5.7 Understandable Rules

The object models and Domain Specific Languages that model ones problem domain, can set oneself up to write rules that are very similar to natural language. They provide themselves to logic that is more understandable to non-software-developers, business analysts, and Domain experts as they are laid down in their language.

2.6 When to use a Rule engine and when not to use Rule engine?

The simplest answer to this question is when there are no sufficient traditional programming approaches to solve this problem. We could use a rule engine when we have the following problems:

- The traditional problem at hand is just too tough to handle for stereotype code.
- The problem need not be too complex, but we cannot see a simple way of building it.
- The problem is cannot be solved by an algorithm based solution.
- The problem is not completely understood and needs iterative modifications and the logic changes often.
- Domain expert / Business analysts are easily available and have moderate technical knowledge; they are generally having great knowledge regarding business rules and processes. Rules facilitate them to express the logic in their own way.

Rule engines are mostly dynamic in the context that the rules can be saved, managed, modified and updated as data in a file, they are often seen as a easy way to easily deploy software, If this is the reason one wishes to use a rule engine, then it is not a good idea as rule engine suits well only when there are declarative rules. It is not a wise option to use a rule engine to replace process flows or work flows as there are several alternatives available in the market to do the same.

2.7 Rules Design Pattern

The Rules Pattern [11] works by pulling out the rules from the rules processing logic by using the Single Responsibility Principle. This makes it simpler to add new rules without making any changes to the rest of the system by using the Open/Closed Principle.

The single responsibility principle states that every module/class should have responsibility over only a single part of the functionality provided by the entire software.[12]

In object-oriented programming paradigm the open/closed principle states that software entities such as classes, modules, functions, etc. always should be open for extension, but closed for modification i.e., such entities can be flexible and allow its behaviour to be extended without modifying any of its source code. [13]

With the Rules Pattern there is an Evaluator class that goes through a collection of rules and executes them. It evaluates the result and then takes a call on what action to take. In the simplest of all the cases this would just execute all the rules, but it is also feasible to incorporate some selection logic such that each rule that allows the Evaluator class to decide whether or not to execute the rule. The fig below describes how the classes interact in a rules design pattern.

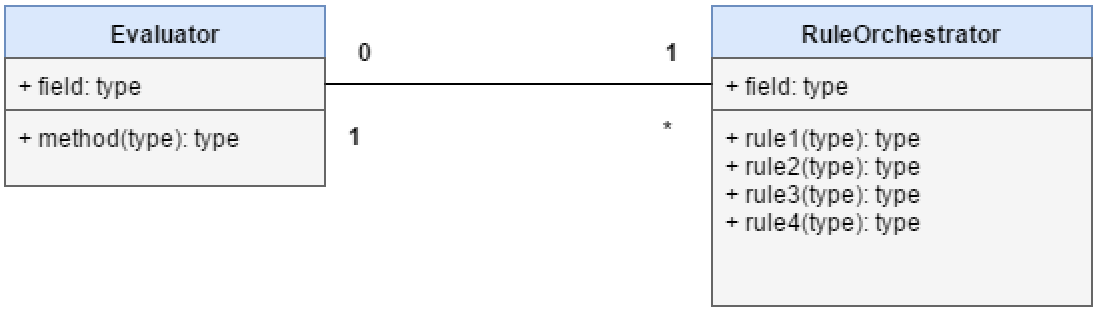


Figure3 : Rules Design Pattern

2.8 Need over available solutions and true motivation

There are multiple options available in today's technology world, but the available options are not so mature to suit out requirement of current application. The summary of few of the tools explored explaining the pros and cons of them are as follows:

- Drools [14]:
  This is a Java based business process rules management system. The positives are they have a sufficient user interface available to manage rules and also integrated them, these rules are not flexible and do not integrate easily will other open source technologies. The feature provided are very strongly coupled to java based platforms and applications. The code needs compilation, integration and testing whenever there are modifications and are difficult to maintain. The application is free but the learning curve is too huge to accommodate our application needs.

- Decisions [15]:
  This is another platform similar to Drools, it contains features that are most similar to out applications requirement but, it is again not open sources and license is expensive. The operational cost, learning curve and maintenance is too high.

- In rule [16]:
  This is a rule engine for business built using .NET platform. This has good features that can be only integrated in the .NET environment. The code needs compilation, integration and testing similar to DROOL. The application is not free there are lot of licensing cost and the learning curve is also relatively high.

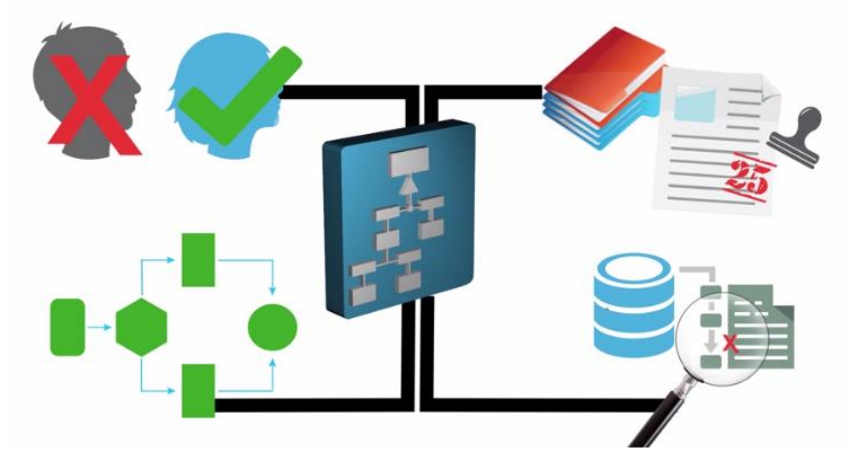


Figure4 : Rules.NET by Decisions

Chapter 3: Project Overview

3.1 Existing System

The existing system consists of multiple nests of conditional statements and rules in order to reconcile the data to take decisions. The domain specific rules written are stereotype and interlock themselves and difficult to scale them by adding new rules. The natural tendency of any software is to undergo changes in the due course of time. The changes if any are very difficult to incorporate and integrating and creating the new rules and propagating through the system are very complex in nature. Due to infeasibility the original system design cannot be incorporated due to compliance issues. The goal of the existing system is to take data as input from various sources such as CSV, text files, Database systems and data feeds etc. apply certain domain specific rules to the data and give the output of the file and also state the different anomalies so that the business analysts can take decisions.

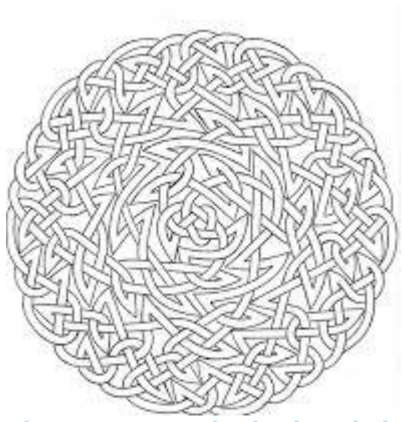


Figure5 : Knotted rules in Existing System

3.2 Problems with Existing System

The code in the existing system has a lot of conditional logic and duplication, thus making it very hard to incorporate new rules as the code gets very difficult to understand and to digest what is actually going on. This sort of code generally has comments elaborating what the different sections of conditional logic are doing. The complexity only gets worse as we have to add more conditions over time and also gets extremely knotted.
There are lot of complex and nested conditional statements which are very difficult to maintain and scale it, any new addition should undergo rigorous integration testing and there is lot of resources, time and cost involved. There are many loose ends that needs to be joined together so various components have more visibility into each other.

3.3 Available Software Solutions and their Issues

Modern technologies have spanned across various business aiding them at different levels. Business Rule engines form the foundation in modelling multiple business rules. There are lot of matured technologies available as proprietary software. Few of the available popular softwares are Drools, InRule etc. which are expensive and learning curve is huge. Since our existing systems components consist of only open-soures technologies, integrating them with available proprietary software requires lot of effort and huge licencing cost.

3.4 Proposed System

The rules grow exponentially and the business analysts want to explore innovative way to model rules on the fly and also add them without any manual intervention with little scripting background they want to leverage open source options where they can have a lot of flexibility and no licensing cost. The decision engine platform provides end to end features so as to satisfy all the current application needs to be more configurable , scalable and automatically deployable to get insights. The whole systems leverages the rules design pattern and completely built using open source technologies.

The proposed system will have the following components

- Rule repository
- Rule Orchestration
- Decision Engine

3.4.1 Rule Repository

The Rule Repository consists of multiple modules which can be modified, deleted and incorporated on the fly. The rules in this repository is easily configurable and for the foundation for the Rule orchestration. The key functionality of the repository is to provide security, version control and easy of managing and changing it on the fly.

Figure6 : Rule Repository

The figure explains the various rules that are available and can be stacked, sorted, consumed, modified, and updated. The high level implementation aspects are explained in the Implementation chapter.

3.4.2 Rule Orchestration

The Rule Orchestration consists of Meta class which whose modules and attributes get generated on the fly depending on which got modified, deleted and incorporated on the fly in the rule repository. The rules in repository are made ready for the Decision engine to consume. The high level implementation aspects are explained in the Implementation chapter.

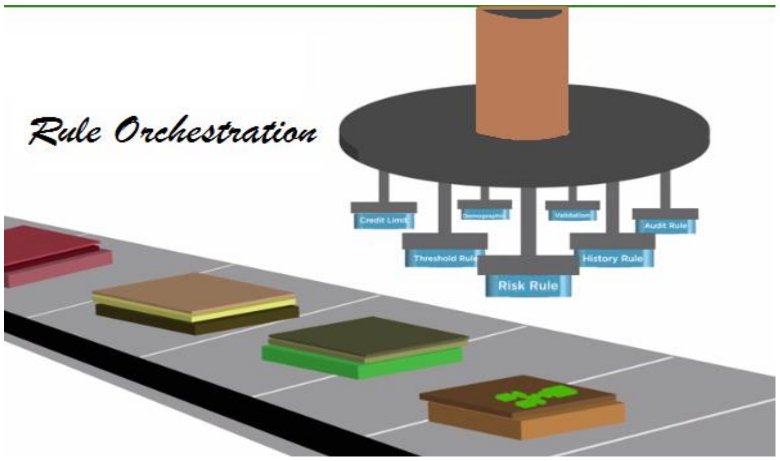


Figure7 : Rule Orchestration

3.4.3 Functionality Repository

The functionality repository consists of all the features the Business Analyst wants to execute. The type and sequence in which the rules are required for individual feature is stored as a JSON by the analyst, the flow sequence stored is pulled up by the decision engine to execute. The implementation details are explained in the Implementation chapter.

3.4.4 Decision Engine

The decision engine forms the crux of this whole platform. It uses the rule orchestration and Functionality repository class to execute the required domain specific rules on the data got either from the API or a CSV. Figure 2 shows the high level flow of how multiple rules can be applied on streams from an API or CSV. The engine has flexibility for the user to pick and choose the flow as deemed necessary based on the situation and requirements of the analysts.

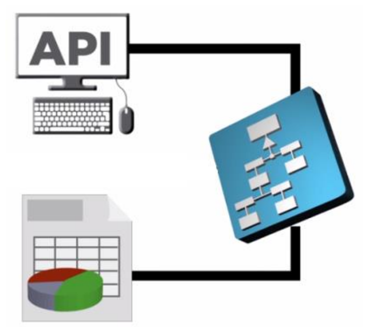


Figure 8: Sources on which Decision Engine operates

The engine chooses a subset of rules from the rule orchestration, the order of execution of subset of rules are based upon the configuration set by the user for a specific functionality.

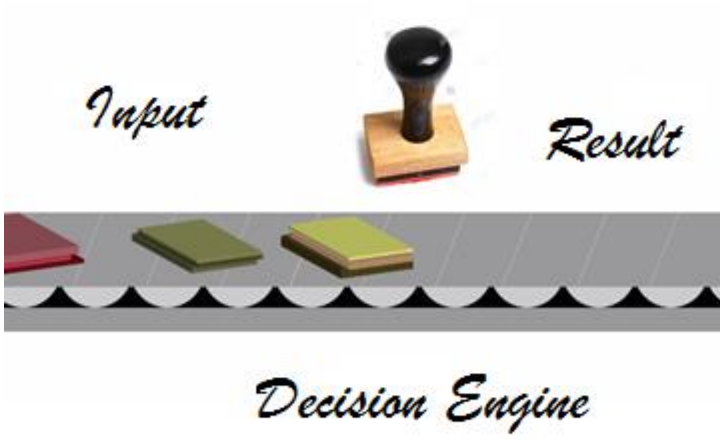


Figure9 : Decision Engine Flow Step 1

The decision engine can be at an high level visualised as figure above the step 1 shows the input of data as a stream, where there is a stamp like engine which applies rules on it and finally the stream keeps moving where the figure below shows the step 2 of the decision where the result of the decision engine on each stream set comes out as a final output upon application of the rules using the orchestra and various features as deemed necessary and available within the frame work.

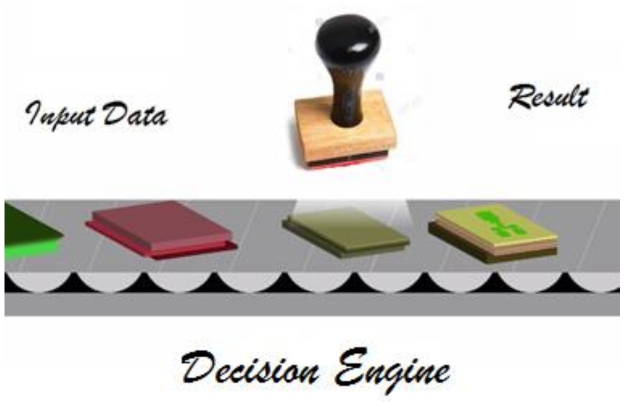


Figure10 : Decision Engine Flow Step 2

The API/CSV is a source of continuous stream of data that when fed to the decision engine gives a resultant output. The output can be something like one of the options as shown in the figure below. The figure shows the output of decision engine, the output is something as follows:

- True or false :
  The data stream can be subjected to several rules and the output is Boolean in nature.
  Example:
  - Does the data pass the integrity test and conform to the standard by all numbers are being type float?
  - Provide an approval or contractual decision.
- Value output :
  The data stream can be subjected to several rules and the output is a value resultant of a computation.
  Example:
  - Does the sum of all the supply is greater than the demand?
  - Score a Credit application.
- Triger:
  When a sequence of data is subjected to a feature then if there are any anomalies found then the output of such anomalies are captured in the data bases and also an email is triggered so the intended person is notified.

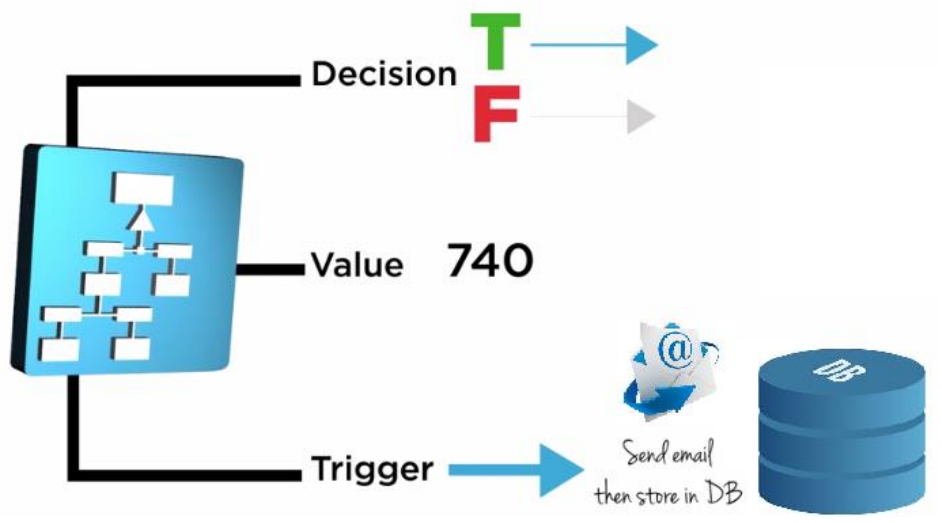


Figure 11: Output of the decision engine

3.5 Objectives

There are no matured open source decision engines built using python which satisfies applications current needs. Hence objectives is to

- To build, deploy and scale the existing stereotype rules/models to a more maintainable and configurable Decision engine.
- To provide an easy interface and framework to model rules on the fly build, test and deploy it without any manual intervention.

3.6 Scope of work

- Studying the existing decision engine environments and architectures.
- Use open source tools and build an engine which can interact with various components.
- Designing/updating existing applications to use the engine.
- Validating the applications and analyzing the post deployment activities.

Chapter 4: Decision Platform

4.1 What is a Platform?

Long back we used to reserve the use of the word platform [17] to show a complete and holistic software programming development environment and its underlying subsystem which consist of language, runtime, components and all associated libraries and binaries. The result of using a platform was generally called a software application, which in further years everyone started calling it an "app". In the new modern era, a platform became something more different. Where one thought it as the underlying computer system, but present day one has to accept the fact that the software industry visualizes a platform to be anything that you can build upon. Products or applications come with predefined business logic that narrows down their ultimate breadth of scope. Platforms on the other hand segregate out the logic functions of applications so that IT software can be built for change.[18]

4.2 Expected Features

The solution should be made to bring the features of the system to replace the existing one and also into operation in all the previous systems fallacies fixed. The following are the features required

4.2.1 User friendly design

The project should provide the user with the ease of avoiding the hassle of hard coding multiple conditional logics and move the manual development effort into hands of non-developer. The design being pro non-developer it will save a lot of effort as the user interface that is being designed is very easy considering the factor that anyone of the non-technical person can use it with ease.

4.2.2 Cost Reduction

Reduce the number of Person-hours required to deploy and testing the changes that need to be incorporated thus saving a lot of effort for the project APIs that are run manually as of now and need a skilled developer. This is a custom application tweaked to suffice the application current needs, the development effort is one time and minimum and makes the whole system reusable and reduced the cost in terms of licensing.

4.2.3 Training

The solution should be designed in such a way that end user can create and modify rules with minimum scripting knowledge and the deployment is completely automated process.
User should be able to use the software with very less training.

4.3 Decision Platform Details

4.3.1 Python Open-Source Scripting Language

Python [19] is the language used to emulate the decision engine.

- Python is Interpreted: Code is processed during runtime by the interpreter. There is no compilation required before executing it.

- Python is Interactive: One can actually sit at a Python prompt and interact with the interpreter directly emulate the programs, this forms the crux of the applications where one can incorporate new rules with limited coding experience.

- Python is Object-Oriented: It has all the features of Object-Oriented style of programming that encapsulates code within objects thus brining in security.

- Python is a Beginner's Language: An amazing language for the beginner-level programmers and supports the good development for the decision engine.

4.3.2 Continuous Integration Using Jenkins

Jenkins [20] is a cross-platform, continuous integration and continuous delivery application that increases significant productivity. The decision engine platform uses Jenkins to build, test and deploy the rules continuously making it easier for non-developers to integrate changes to the rules, and making it easier for end users to use it easily. It also allows the Business analysts to continuously deliver new rules by providing powerful ways to define the build pipelines and integrating with python scripts and version controlling.

4.3.3 Connection to various API and Commercial Databases

The platform provides easy wrappers and libraries for extracting data from various commercial databases as well as from API's that provide data.

4.3.4 Output Handlers

The platform provides an interface which would provide how the output has to be generated. Either it should print on screen or save it as a CSV or persist it in database.

4.3.5 Integrated Development/Execution Environment

The integrated Development and execution environment used is Pycharm. Pycharm provides perfect support for code analysis, graphical debugging and an integrated unit test environment, perfect integration support with version control systems, and supports easy development with python, XML and editing files. PyCharm supports cross-platform development.

Chapter 5: Implementation

5.1 Project Overview

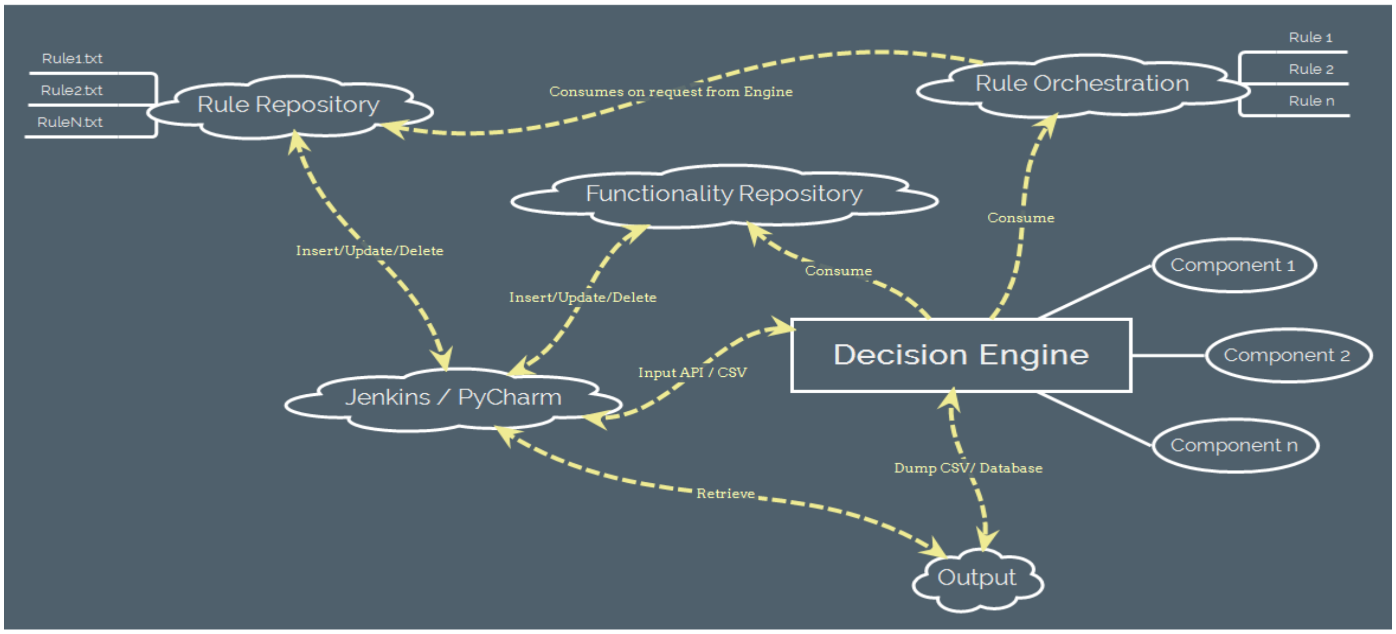


Figure 12: Project Overview

The above figure (figure 12) shows the high level overview of the system. The Business analyst interacts with the decision engine using the Jenkins or the PyCharm. Business analyst can make changes to the rules in rules repository, JSON in the functionality repository by Insert, Update and Delete from Jenkins/PyCharm Interface. The decision Engine is the core, where all the processing takes place. The process is invoked by the analyst using Jenkins/PyCharm, The engine dumps the output to CSV/Database which can be retrieved.

5.2 Rule Repository

The rule repository folder consists all the rules that is part of the decision engine.

5.2.1 The rule repository Folder Structure

The rule repository folder starts with the root directory by default and the nomenclature of the folder is "rules". It creates the folder is the folder is not existent

5.2.2 Rules

The rules are stored inside the rules folder. The rules are nothing but python modules stored in text files that. These modules are pre tested and conforms to

Single responsibility principle. The unit test for individual modules help to ensure the correctness of individual functionality. The Rule files are added to the folder or modified using Jenkins or PyCharm editor by the business analyst.

5.2.3 Rule Repository Base Class

The Rules Base class is a skeleton class which contains a standard structure for the Decision engine to use. The method of the class gets added at runtime from the rules present in the Rules folder. All the successful method loads are made available to the decision engine and the erroneous loads are also made note of in the log.

5.3 Functionality Repository

The Functionality repository is a descriptive interface just like the Rules folder which contains a standard structure for the Decision engine to know and run the sequence of execution. The each of the functionality is saved as a JSON inside a test file and read at runtime from the decision engine based on what functionality needs to be run on the set of data. All the successful rules runs are print on screen and the erroneous loads are also made note of in the log. The Functionality JSON are added to the folder or modified using Jenkins or PyCharm editor. The functionality JSON's are added to the folder or modified using Jenkins or PyCharm editor by the business analyst. The available Functionality is as follows:

1. Data Reconciliation: There are a few set of rules that needs to be applied on data stream. The set of rules are applied in a sequence or with multiple nested sequences. The output is generally in the nature of sanity test for individual records, sometimes it could also be updating of the records. This feature can also be used to detect anomalies in the data.
2. Contract Compliance : Where there is a contract between the supplier and consumer , there needs to be certain compliance maintained in terms of how much one consumes

5.4 Decision Engine

The decision engine is the central executor; it provides command arguments features to get decision. The basic input for the Engine is the API/CSV-file, the functionality that needs to be applied and output format.
The various features it provides currently are as follows:

- Input CSV file data and "Reconcile" parameter to perform reconciliation and get the output of the reconciled data as a csv.
- Input the data get the contract Minimum and contract maximum compliance as output.

The decision engine is written in pure python code. The decision engine can connect and interact with various other components of the application to retrieve the necessary data or send or even update as deemed necessary.

5.5 Authentication and Authorization

Authentication is the mechanism using which systems securely identify who their users are. The authentication scheme used in the system is token based authentication. Authorization is the mechanism using which a system gets to know what level of access a particular authenticated user should have access what kind of resources controlled by the entire system. Security is one major area where customization is very important the Platform provides all the necessary security system using OAuth. Token-based authentication provided here is the specialized version of Basic Authentication system provided in all authentication systems. The Authorization header tag consists of the authorization token as the username, and there is no password required. The script in platform assumes that user accounts are stored in user_accounts MongoDB collection. All the included API resources and methods in the platform is secured unless they are made explicitly public.

Summary

The aim of this dissertation work was to bring in flexibility and autonomous capability to the current system, so as to gain cost and efficiency. There were no matured open source decision engines built using python which satisfies applications current needs. Hence the moto was to build an effective engine that can automatically build, deploy and scale the existing stereotype rules and models to a more maintainable and configurable Decision engine.

The current platform has an easy interface and framework to model rules on the fly build, test the rules and also deploy it without any manual intervention. The decision engine platform is built after studying the existing decision engine environments and architectures, while keeping in mind the current application needs. Leveraging upon all open source tools and the engine’s efficiency gain is brought about by having features that interact with various components within the system and as well as the present application.

The current platform built during this dissertation course work has also been tweaked to seamlessly integrate, design, update and work with existing applications, but in a more sophisticate and configurable way. The two Key features supported during the current Phase of the platform is Data Reconciliation and Contract compliance. These features provide the business all the flexibility to gets valuable insights into the data curated from multiple sources.

Recommendations and Future works

The goal of the current platform was to bring in flexibility and configurability to existing stereotype nested if-then-else blocks. The current platform incorporates all the necessary functionality, flexibility, configurability with certain level of assumptions. The current platform assumes that the business analyst has a certain level of basic programing knowledge.

The current system has the rules modified and rule flow decided and configured by the business analyst with a very minimum programming knowledge. Future recommendation would be to completely eliminate even the smallest need of programing exposure so that the Business analyst can get exposure to one that has a fully independent body which uses natural language processing to configure, add, delete and modify the Rules. The system should have an extension where the rules are written in a language of the business analyst, but gets converted to syntactic python code that automatically gets built, tested and deployed.

Apart from that of a Natural language processing engine to create and decide the rules and rule process flow, there has to be a more sophisticated graphical user interface developed so that it is even more intuitive and flexible to make changes and also keep track of the changes.

References

[1] Data Source Wiki, https://en.wikipedia.org/wiki/Data_source

[2] Computer File Wiki, https://en.wikipedia.org/wiki/Computer_file

[3] Data Stream Wiki, https://en.wikipedia.org/wiki/Data_stream

[4] Definition, https://www.techopedia.com

[5] Data Reconciliation-
https://en.wikipedia.org/wiki/Data_validation_and_reconciliation

[6] Software Architectures,
https://en.wikipedia.org/wiki/Software_architecture

[7] Software Design Patterns,
https://en.wikipedia.org/wiki/Software_design_pattern

[8] Business Rules, https://en.wikipedia.org/wiki/Business_rule

[9] Rule Engine,
https://en.wikipedia.org/wiki/Business_rules_engine

[10] PyCharm, https://www.jetbrains.com/pycharm/features/

[11] Rules Design Pattern,
https://www.pluralsight.com/courses/patterns-library

[12] Single Responsibility Principle,
https://en.wikipedia.org/wiki/Single_responsibility_principle

[13] Open Closed Principle,
https://en.wikipedia.org/wiki/Open/closed_principle

[14] Drools, http://www.drools.org/

[15] Decisions, decisions.com

[16] InRule, http://www.inrule.com/

[17] Software Platform,
http://www.forbes.com/sites/adrianbridgwater

[18] Decisions Platform, http://www-
07.ibm.com/my/events/isbts/lotusiod/pdf/Collaborative_Decision-
Making_Platforms-
A_New_Way_to_Make_Decisions_Gartner_Ian_Bertram.pdf

[19] Python, https://www.python.org/

[20] Jenkins, https://jenkins.io/

COEXTRIX TECHNOLOGIES PVT. LTD.

No. 45/1, 3rd Floor, 100 Feet Ring Road, J.P. Nagar II Phase,
Bangalore - 560 078. Karnataka, India
Ph. : +91 80 2658 4009 /4042 / 0363 / 4120 9385
E-mail : interact@coextrix.com Website : www.coextrix.com

coextrix
Defining the future of excellence
CIN : U72200KA2009PTC050027

Place : Bangalore
Date : 13th,Oct 2016.

To Whom It May Concern

It has been brought to our notice that **Pramod Kumar N,** BITS ID. No: **2014HT13292** has enrolled for M.Tech in Software Systems course in your institution and would require him to get the source code of the project **Decision Engine Platform** that he was a part of, in order to complete the dissertation work.

We regret to inform you that, in accordance with our company policies, we do not authorize our employees to access/take the project artifacts like source code, etc. outside of our workplace.

[illegible]

(Signature)
Ganesh Guruswamy,
CTO,
Coextrix Technologies PVT. LTD.

COEXTRIX TECHNOLOGIES PVT. LTD.

Checklist of items for the Final Dissertation Report

This checklist is to be attached as the last page of the report.

This checklist is to be duly completed, verified and signed by the student.

1.	**Is the final report neatly formatted with all the elements required for a technical Report?**	[✓ Yes] / No
2.	Is the Cover page in proper format as given in Annexure A?	[✓ Yes] / No
3.	Is the Title page (Inner cover page) in proper format?	[✓ Yes] / No
4.	(a) Is the Certificate from the Supervisor in proper format? (b) Has it been signed by the Supervisor?	[✓ Yes] / No [✓ Yes] / No
5.	Is the Abstract included in the report properly written within one page? Have the technical keywords been specified properly?	[✓ Yes] / No [✓ Yes] / No
6.	Is the title of your report appropriate? **The title should be adequately descriptive, precise and must reflect scope of the actual work done.** Uncommon abbreviations / Acronyms should not be used in the title	[✓ Yes] / No
7.	Have you included the List of abbreviations / Acronyms?	[✓ Yes] / No
8.	Does the Report contain a summary of the literature survey?	[✓ Yes] / No
9.	Does the Table of Contents include page numbers? (i). Are the Pages numbered properly? (Ch. 1 should start on Page # 1) (ii). Are the Figures numbered properly? (Figure Numbers and Figure Titles should be at the bottom of the figures) (iii). Are the Tables numbered properly? (Table Numbers and Table Titles should be at the top of the tables) (iv). Are the Captions for the Figures and Tables proper? (v). Are the Appendices numbered properly? Are their titles appropriate	[✓ Yes] / No [✓ Yes] / No [✓ Yes] / No [✓ Yes] / No [✓ Yes] / No [✓ Yes] / No
10.	Is the conclusion of the Report based on discussion of the work?	[✓ Yes] / No
11.	Are References or Bibliography given at the end of the Report? Have the References been cited properly inside the text of the Report? Are all the references cited in the body of the report	[✓ Yes] / No [✓ Yes] / No [✓ Yes] / No
12.	Is the report format and content according to the guidelines? The report should not be a mere printout of a Power Point Presentation, or a user manual. Source code of software need not be included in the report.	[✓ Yes] / No

Declaration by Student:

I certify that I have properly verified all the items in this checklist and ensure that the report is in proper format as specified in the course handout.

[illegible]

Signature of the Student

Place: BANGALORE

Date: 13th Oct, 2016

Name: PRAMODKUMAR. N.

ID No.: 2014HT13292