Informed and Timely Business Decisions – A Data-driven Approach

Veera Tadikonda and Daniela Rosca
Computer Science and Software Engineering Department Monmouth University
West Long Branch, NJ, USA
{veera.tadikonda@gmail.com, drosca@monmouth.edu}

Abstract— **One of the main characteristics of business rules is their propensity for frequent change, due to internal or external factors to an enterprise. As these rules change, their immediate dissemination across people and systems in an enterprise becomes vital. The delay in dissemination can adversely impact the reputation of the enterprise, and cause significant loss of revenue. The current BRMS are often maintained by the IT group within a company, therefore the modifications of the BRs intended by executive management would not be instantaneous, since they have to be coded, and tested before being deployed. Moreover, the executives might not have the possibility to take the best decisions, without having the benefit of analyzing historical data, and quickly simulating what-if scenarios to visualize the effects of a set of rules on the business. Some of the systems that provide this functionality are prohibitively expensive. This paper addresses these challenges by using the power of Big Data analysis to source, clean and analyze historical data that is used for mining business rules, which can be visualized, tested on what-if scenarios, and immediately deployed without the intervention of the IT group. The proposed approach is instantiated in this paper by using open source components to mine stop loss rules for financial systems.**

Keywords: Business Rules, Decision Support, Big Data, Rule Mining

I. Introduction

Using a single piece of information to derive decisions could lead to polarized results and unprofitable decisions. To overcome this limitation, information from different sources, quantitative as well as qualitative has to be utilized. With today's overwhelming number of information sources, one cannot neglect the benefits they can bring to a business by integrating the information and extracting a meaning from it. Based on the recent advances in Big Data analytics, businesses can use historical and market data to gain competitive advantages. These methods prove to be beneficial to software engineering tasks as well, such as identifying and classifying requirements and mining sentiment analysis from App reviews [1], [2], automatic requirements elicitation from feedback comments [3], mining user comments for helping requirements evolution[4], or extracting features from product descriptions to recommend features implementation for software product lines [5].

Another major area of applying mining techniques in software engineering has been process mining [6] for tasks such as discovering processes from event logs, testing the conformance of processes, improvement of business processes, or mining user's intentions in intentional process models [7], [8]. One very important component of business processes has been the treatment of business rules (BRs). In today's highly agile world, where businesses need to quickly adapt to market changes, business processes have to dynamically react, and avoid locking processes in hard to change IT solutions. vonHalle [9] and Ross [10] have long been advocating the business rules approach, that claims that all BRs in an enterprise should be collected in a centralized business rules management system (BRMS). However, these BRs need to be created and verified by the business experts, who might not be familiar with formal languages, and hence their dependence on the IT staff. In order to alleviate this problem, a couple of solutions have been proposed, such as the OMG standard, Semantics of Business Vocabulary and Rules – Structured English (SBVR-SE) [11], a controlled natural language for business rules specifications (RuleCNL) [12], or the transformations of the SBVR specifications into UML/OCL [13], or BPMN metamodel [14]. However, Lucie performed a state of the art analysis of business rules languages in [15], and concluded that most of them are difficult to use by business experts.

To avoid the need of using formal or semi-formal languages for expressing business rules, we propose an approach where the business rules are mined from existing historical data and market data, and are presented to the business user in an intuitive format for verification and immediate deployment. This solution is close to [16], which advocates the induction of BRs from statistical data using decision trees learning. With the recent advances in Big Data and Machine Learning, the abilities to create rules have been considerably expanded. The approach proposed in this paper covers not only the creation and modification of BRs, but also supports the dynamic decision making needs of a business user, by presenting in a visual, easy to grasp way, the historical and current market data, and allowing the simulation of what-if scenarios to understand the impact of a set of rules on the business. Whenever the business user approves new rules or changes existing ones, they can be immediately deployed using an operational rule engine, as opposed to approaches mentioned above, which stop at the specification of business rules in a modeling language, without the benefit

DOI reference number: 10.18293/SEKE2016-216

of observing their effects on the business. This way, an agile reaction of the business to the market is made possible.

The remainder of the paper is structured as follows: Section II describes the process and architecture that support the proposed approach. Section III presents the method for mining business parameters out of historical data, while Section IV describes how these parameters are used for mining BRs conditions. Section V shows the steps for generating the rules, while Section VI presents the deployment, execution, and feedback of the rules' execution. Finally, the paper concludes with a summary, limitations and plans for extending this work.

II. Process and Solution Architecture

A. Process

The solution proposed in this paper allows the business user to interactively and dynamically create or change business rules, using historical and market data. The process (as shown in Fig.1) is started with the classification phase, where the variables used in the subsequent rules are identified, based on either a supervised or unsupervised learning algorithm, depending on whether the variables are well known for the business user, or they need to be discovered from a data set, or a new source of data needs to be mined. Using the variables identified in the previous step, and the trends from the available data set, the system identifies conditions that can be applied to the active data to create corresponding rules. The business user has the option to vary the range of the variables that are under control by the business, dynamically visualizing the new conditions. Once the conditions are decided upon, the system creates corresponding rules. The user has the ability to choose all or some of the generated rules for deployment. Alternatively, the user can create his own rules with autosuggestions from the system. The created rules are applied to the active data, to generate actionable decision rules, specific to the domain of application. Before deciding on the best rules, the business user can visualize the effects of applying a set of rules in what-if scenarios. Once the best rule set is identified, it is deployed for execution on the active data set, using a rule engine. The results of the execution are fed back into the system's persistent storage, to be used in later decision processes.

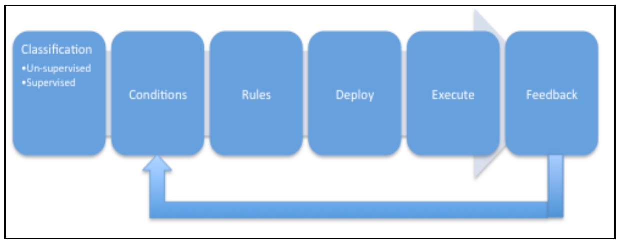


Figure 1. Process for dynamic BRs creation, modification and deployment

The domain of application for this paper is the stop-loss rules used in financial systems. Stop-loss is a technique of safeguarding the losses by the investor. The investor chooses to sell the stock if it falls to a certain price, such that the amount lost is limited to his risk capacity. The scenario that will be implemented in this paper refers to making recommendations for selling or buying a certain stock. It shadows the following procedure: using a standard stock movement prediction model, build an exposure chart, build a historical chart, create stop-loss rules and generate upside threshold messages for selling or buying. The fundamentals and market information from various sources are collated to predict an outcome for the stocks. These outputs/ predictions/ recommendations are converted to actionable rules. This scenario follows the steps of algorithmic trading.[17]

B. System Architecture

The approach proposed here relies on a generic architecture that is composed of four building blocks:

1. *Data collation and analysis* – to capture heterogeneous data from multiple sources and analyze it.
2. *Data transfer* – a typical Extract Transfer Load (ETL) component that extracts data from multiple sources and transforms it to be stored in an appropriate format in a NoSQL data store.
3. *Aggregation/Computation* – processes data from the NoSQL data store using Big Data tools, in order to extract business parameters and rule conditions.
4. *Rule management and deployment* – to support the creation, modification and deployment of the rules generated to a rules engine, in order to trigger the execution of relevant actions. It also helps to capture the feedback of the rules execution such that it can be used in future rule generation iterations.

An instantiation of this generic architecture is described next for mining stop loss rules for financial systems.

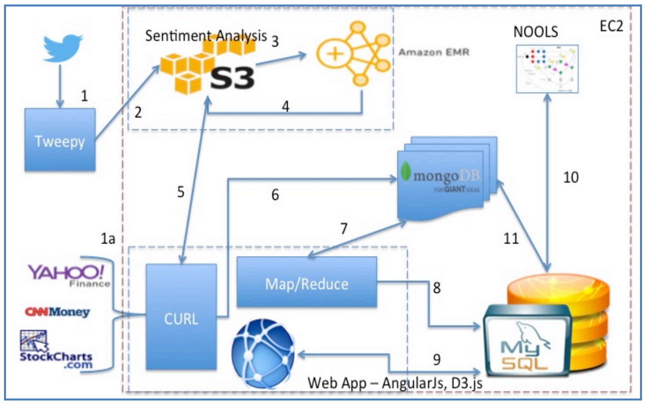


Figure 2. Instantiated Architecture Diagram

The architecture shown in Fig. 2 is using data from multiple sources, such as current portfolio, historical trades, Tweets (for determining sentiment and market information), Stock, Sector, and Index quotes from Yahoo, CNNMoney, and Stockcharts, respectively. Tweepy is used for extracting tweets pertaining to the stocks in the portfolio and provide them to Amazon EMR (Elastic Map-Reduce) to analyze the number of positive, negative, and neutral tweets. The results of the analysis are stored into Amazon S3 (simple storage service). Using CURL, the sentiment analysis information, as well as the stock prices and trade information, are stored into MongoDB. The information from this document storage is used in a Map-Reduce script to generate scores for stock, sector, and index quotes, as well as sentiment scores, necessary for mining conditions. The output from Map-

Reduce is stored on a MySql transactional database, as conditions, together with rules, historical trades and portfolio data. The trade rules are being deployed in real-time using Nools (the Rete algorithm based rule engine) and the current data set. The output from the execution engine is circled back to the system's data stores to improve the accuracy of its suggestion algorithm in future iterations. The interfaces between the architectural components are shown in Table 1.

Id	Source	Purpose
1	Twitter Feed to Tweepy	Fetch tweets from Twitter
1a	Stock, Sector and Index quotes from Yahoo, CNNMoney, Stockcharts	Fetch Stock, Sector and Index quotes
2	Tweets	Store tweets into Amazon S3 storage
3	Tweets from Amazon S3 to Amazon EMR	Process Tweets using Amazon EMR
4	Amazon EMR output	Sentiment analysis output from Amazon EMR
5	Amazon S3 Sentiment analysis output	Using CURL to fetch Sentiment analysis output from Amazon S3
6	CURL to store Stock and Sentiment	Store Stock and Sentiment data into MongoDB
7	Data for Map-Reduce function	Interchange data between MongoDB and Map-Reduce function to generate scores
8	Map-Reduce output	Push Map-Reduce output into MySQL
9	Web Application to MySQL	For data interchange between Web application and MySQL
10	MySQL to NOOLS	Send and receive Rules and Execution information between MySQL and NOOLS
11	MySQL to MongoDB	Send execution results from MySQL to MongoDB

Table 1. Interfaces between the architectural components.

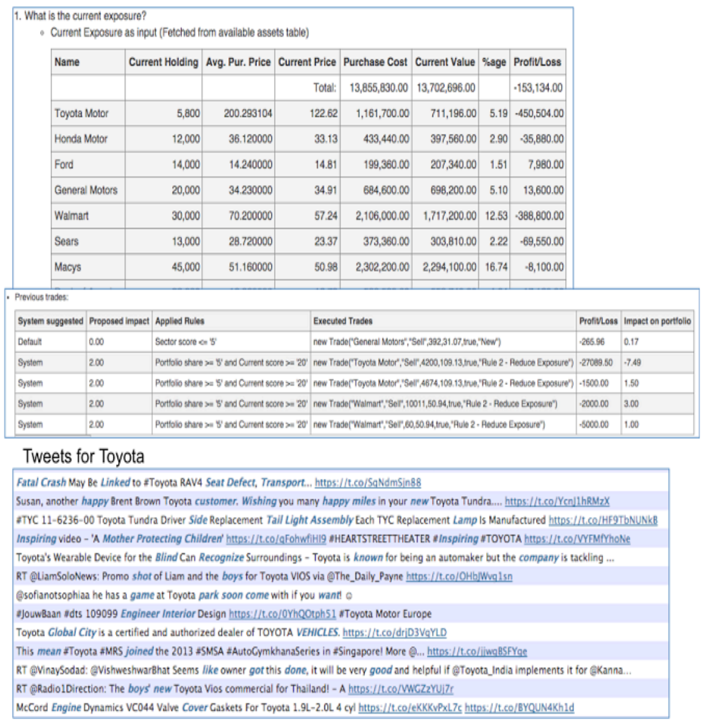
The entire architecture runs on the Amazon EC2 (Elastic Computer Cloud) platform, to allow the system to be used simultaneously by multiple users on different instances. The solution is built by integrating various open source applications in a web application. The design approach for this solution has kept customization to a minimal level, such that it can be applied across industries. Also, the usage of loosely coupled components in the architecture makes the solution implementable across different platforms and scenarios, and makes it possible that a part of the solution is in demilitarized zone, and the rest of the application, behind firewall. Next, we describe how the components of this architecture contribute to each phase of the process model shown in Figure 1.

III. CLASSIFICATION - MINING BUSINESS PARAMETERS

In this phase, related business parameters (variables) are determined from the historical trades data, current portfolio, and market data (see Figure 3), using a supervised or un-supervised machine-learning algorithm. For this application, since the variables are already known (stock price, sector score, index score, overall score and sentiment score), a linear regression supervised learning algorithm has been utilized. The variables are chosen to get a broad perspective of the stock price changes, since a stock price is generally dependent on the following factors:

- Company news and performance (reflected by the Twitter feed – Sentiment score)
- Industry performance (reflected by Sector score)
- Investor sentiment (Bull/bear market – reflected by Index score)

Based on these scores, the Overall Stock Score can be calculated as will be shown in Section V.



1. What is the current exposure?
   - Current Exposure as input (Fetched from available assets table)

Name	Current Holding	Avg. Pur. Price	Current Price	Purchase Cost	Current Value	%age	Profit/Loss
			Total:	13,855,830.00	13,702,696.00		-153,134.00
Toyota Motor	5,800	200.293104	122.62	1,161,700.00	711,196.00	5.19	-450,504.00
Honda Motor	12,000	36.120000	33.13	433,440.00	397,560.00	2.90	-35,880.00
Ford	14,000	14.240000	14.81	199,360.00	207,340.00	1.51	7,980.00
General Motors	20,000	34.230000	34.91	684,600.00	698,200.00	5.10	13,600.00
Walmart	30,000	70.200000	57.24	2,106,000.00	1,717,200.00	12.53	-388,800.00
Sears	13,000	28.720000	23.37	373,360.00	303,810.00	2.22	-69,550.00
Macys	45,000	51.160000	50.98	2,302,200.00	2,294,100.00	16.74	-8,100.00

- Previous trades:

System suggested	Proposed impact	Applied Rules	Executed Trades	Profit/Loss	Impact on portfolio
Default	0.00	Sector score <= '5'	new Trade("General Motors","Sell",392,31.07,true,"New")	-265.96	0.17
System	2.00	Portfolio share >= '5' and Current score >= '20'	new Trade("Toyota Motor","Sell",4200,109.13,true,"Rule 2 - Reduce Exposure")	-27089.50	-7.49
System	2.00	Portfolio share >= '5' and Current score >= '20'	new Trade("Toyota Motor","Sell",4674,109.13,true,"Rule 2 - Reduce Exposure")	-1500.00	1.50
System	2.00	Portfolio share >= '5' and Current score >= '20'	new Trade("Walmart","Sell",10011,50.94,true,"Rule 2 - Reduce Exposure")	-2000.00	3.00
System	2.00	Portfolio share >= '5' and Current score >= '20'	new Trade("Walmart","Sell",60,50.94,true,"Rule 2 - Reduce Exposure")	-5000.00	1.00

Tweets for Toyota

Fatal Crash May Be *Linked* to #Toyota RAV4 *Seat Defect*, *Transport*... https://t.co/SqNdmSjn88
Susan, another *happy* Brent Brown Toyota *customer*. *Wishing* you many *happy miles* in your *new* Toyota Tundra.... https://t.co/YcnJ1hRMzX
#TYC 11-6236-00 Toyota Tundra Driver *Side* Replacement *Tail Light Assembly* Each TYC Replacement *Lamp* Is Manufactured https://t.co/HF9TbNUNkB
Inspiring video - 'A *Mother Protecting Children*' https://t.co/qFohwfiHI9 #HEARTSTREETTHEATER *#Inspiring* #TOYOTA https://t.co/VYFMfYhoNe
Toyota's Wearable Device for the *Blind* Can *Recognize* Surroundings - Toyota is *known* for being an automaker but the *company* is tackling ...
RT @LiamSoloNews: Promo *shot* of Liam and the *boys* for Toyota VIOS via @The_Daily_Payne https://t.co/OHbJWvq1sn
@sofianotsophiaa he has a *game* at Toyota *park soon come* with if you *want*! ☺
#JouwBaan #dts 109099 *Engineer Interior* Design https://t.co/0YhQOtph51 #Toyota Motor Europe
Toyota *Global City* is a certified and authorized dealer of TOYOTA *VEHICLES*. https://t.co/drjD3VqYLD
This *mean* #Toyota #MRS *joined* the 2013 #SMSA #AutoGymkhanaSeries in #Singapore! More @... https://t.co/jjwqBSFYqe
RT @VinaySodad: @VishweshwarBhat Seems *like* owner *got* this *done*, it will be very *good* and helpful if @Toyota_India implements it for @Kanna...
RT @Radio1Direction: The *boys*' *new* Toyota Vios commercial for Thailand! - A https://t.co/VWGZzYUj7r
McCord *Engine* Dynamics VC044 Valve *Cover* Gaskets For Toyota 1.9L-2.0L 4 cyl https://t.co/eKKKvPxL7c https://t.co/BYQUN4Kh1d

Figure 3. A snapshot of current portfolio data, historical trades, and market data

The linear regression algorithm is used to get the best-fit line for each of the above scores. To get the best-fit line, the linear regression formula of y = a + bx is used, where "a" is the intercept and "b" is the slope. The r^2 provides the correlation coefficient, with values between -1 to +1. The inclination towards +1 denotes a strong correlation between the variables being tested, and thus for this application, a coefficient value above +0.5 is taken as a strong indicator of the relevance of the variables on the stock prices at that point in time. From the variables mentioned above, only the Sector score and Overall score show a strong correlation with the stock price, with r^2 values of 0.98 (as seen in Figure 4).

IV. DETERMINING RULES CONDITIONS

The relevant variables are used as inputs to another regression algorithm to find the best-fit line that will help identify the optimal values for them, values that will be used in determining the rules' conditions.

The pentagon chart (radar chart shown in Figure 4) plots the current scores of five axes that influence the risk determination for a stock. The five axes comprise of Alpha, Beta, Sharpe ratio, Risk tolerance, and Time horizon.

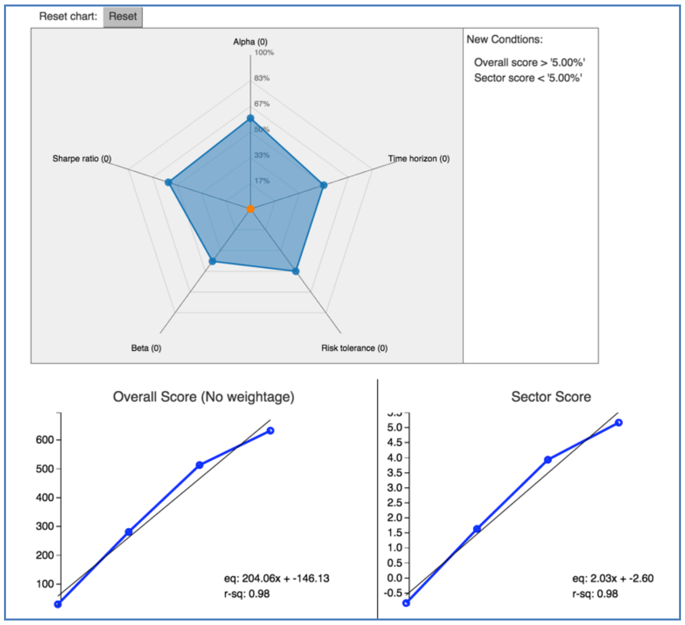


Figure 4. Radar chart visualization of new rules conditions

The "New Conditions" window shows the details of the newly mined conditions (for the sample set that we used, the mined conditions were "Overall Score > 5%" and "Sector Score <5%"). These conditions are the ones that are carried to the next step, the rules generation. The conditions resulting here are generated by the system based on the linear regression algorithm, and further modified by the user by dynamically adjusting the arm values on the radar chart.

Of the five axes of the radar chart, Alpha, Beta and Sharpe ratio are driven by the market conditions. The investor/portfolio manager controls only the Risk Tolerance and Time Horizon. A portfolio manager can reduce or increase Risk. In addition to the information on the other 3 axes, the change on this axis will determine the new conditions that could be applied to the portfolio. The intent is to optimize the portfolio and reduce the loss. Similarly, the Time Horizon can be adjusted to reduce loss and maximize returns. For example, a longer time horizon could flatten out the intermediate changes in a stock and yield better returns. But, if the sector is not performing well and is going through a bear phase, it might actually be riskier to have a longer time horizon.

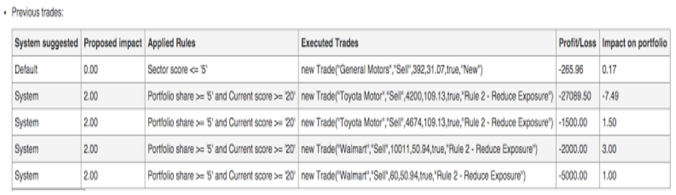
One additional step in helping the business user to make informed decisions is the evaluation of the impact of the identified conditions on the portfolio, based on previous trades. The impact is calculated as:

$$\textit{Stock PL} = \textit{Quantity} \times (\textit{Current Price} - \textit{Avg. Purchase Price}) \quad (1)$$

$$\textit{Impact} = (\textit{Stock PL} \;/\; \textit{Net PL of Portfolio}) \times 100 \quad (2)$$

$$\text{where } PL = \textit{Profit/Loss}$$

The system will discard the conditions with the least impact. Figure 5 shows the anticipated vs. actual impact, thus allowing the portfolio manager to see which of the conditions gave the maximum impact on the portfolio, based on past trades with similar conditions, vs the actual impact on the current portfolio.



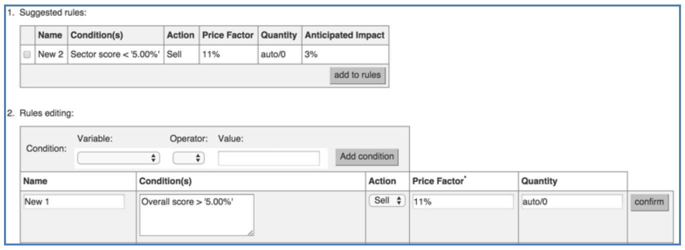
• Previous trades:

System suggested	Proposed impact	Applied Rules	Executed Trades	Profit/Loss	Impact on portfolio
Default	0.00	Sector score <= '5'	new Trade("General Motors","Sell",392,31.07,true,"New")	-265.96	0.17
System	2.00	Portfolio share >= '5' and Current score >= '20'	new Trade("Toyota Motor","Sell",4200,109.13,true,"Rule 2 - Reduce Exposure")	-27089.50	-7.49
System	2.00	Portfolio share >= '5' and Current score >= '20'	new Trade("Toyota Motor","Sell",4674,109.13,true,"Rule 2 - Reduce Exposure")	-1500.00	1.50
System	2.00	Portfolio share >= '5' and Current score >= '20'	new Trade("Walmart","Sell",10011,50.94,true,"Rule 2 - Reduce Exposure")	-2000.00	3.00
System	2.00	Portfolio share >= '5' and Current score >= '20'	new Trade("Walmart","Sell",60,50.94,true,"Rule 2 - Reduce Exposure")	-5000.00	1.00

Figure 5. Assessing the impact of conditions using previous trades

V. GENERATING RULES

Based on the conditions identified in the previous step, the web application suggests new rules that can be edited by the business user before deploying them.



1. Suggested rules:

	Name	Condition(s)	Action	Price Factor	Quantity	Anticipated Impact
☐	New 2	Sector score < '5.00%'	Sell	11%	auto/0	3%

add to rules

2. Rules editing:

Condition: Variable: Operator: Value: Add condition

Name	Condition(s)	Action	Price Factor*	Quantity	
New 1	Overall score > '5.00%'	Sell	11%	auto/0	confirm

Figure 6. Rules editing

The information associated with a rule, as shown in Figure 6, consists of *Name, Conditions, Action, Price Factor, Quantity, and Anticipated Impact*. The *Name* is used to easily identify each rule among all the existing rules. The combination of one or more *Conditions* of a rule would get matched to the active data during deployment. Each condition has the form *<variable> <operator> <value>,* where the variables are facts from the domain of application. The *Action* determines what kind of trade should occur; in a stock trading scenario, there are only 2 possible actions, either "Sell" or "Buy". In a different scenario or application, the options would differ accordingly. The *Price Factor* denotes the price at which either of the action should happen. The options for this solution are: default (11%), fixed amount ("10"), percentage ("5%"), incremental ("up 3%"). The system uses the price determination based on the *percentage stop-loss method*, where the selling price is determined based on a set percentage of the initial purchase price of the stock, and the current price. For this application, the default percentage is set to 11%, as the middle of the normal range of 5-15%, based on the risk tolerance of the investor. The options for expressing the Price Factor provide the user with extreme flexibility in creating the rule. The *Quantity* determines the number of shares to be bought or sold. The options available to the user are: as percentage, as fixed quantity, or system determined based on rules. When the quantity is set as "100%", the entire quantity held is sold, or if the action is "buy", the number of shares will be doubled. Whereas, if the quantity is set as "50", a fixed quantity of 50 shares will be bought or sold. If the quantity that is available for sale is less than the denoted quantity, the existing quantity is sold. The other option is "auto/0", this is the default option where the system determines the quantity based on the portfolio share. If the portfolio share is greater than 5%, the system will determine the necessary quantity to bring the portfolio share to 5% (normal range for each stock is 3-5% shares of the entire portfolio) [17]. Based on past trades with similar conditions, the system calculates the *Anticipated Impact* of the rule on the portfolio, according to equations (1) and (2). For the sample set we used, the value of the anticipated impact was 3%.

Each field of a rule can be edited, with the exception of the *Anticipated Impact*. To help the business user make informed decisions when editing or creating new rules, the system will support the visualization of the stock movement for the previous 4 weeks, for each stock in the portfolio, as a trend line chart.(see Figure 7).

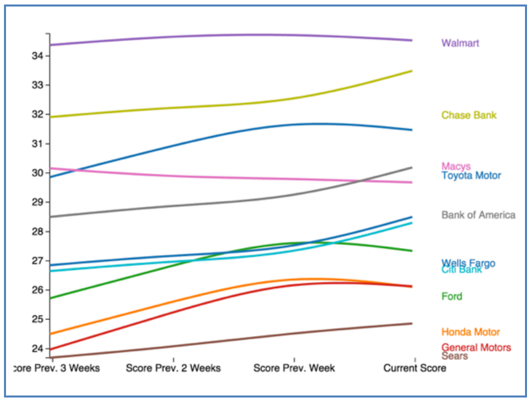


Figure 7. Stocks score trends

The stock scores in Figure 7 are calculated based on a weighted method that takes into consideration various variables for each stock, as follows:

1. Source the stock historical data - Averaged weekly
2. Source the sectorial data - Taken monthly
3. Perform sentiment analysis - Averaged Weekly
4. Calculate 50 day EMA (Exponential Moving Average) for each score (stock, sector, sentiment).
   EMA stands for Exponential Moving Average, a stock price average that applies more importance to the recent prices, reacting faster to the recent price changes, than a simple moving average.
5. Generate the Overall Score for each stock in the portfolio:

$$\text{Overall Score} = \text{StockEMA} * 0.5 + \text{SectorEMA}*.25 + \text{SentSc}*.25 \quad (3)$$

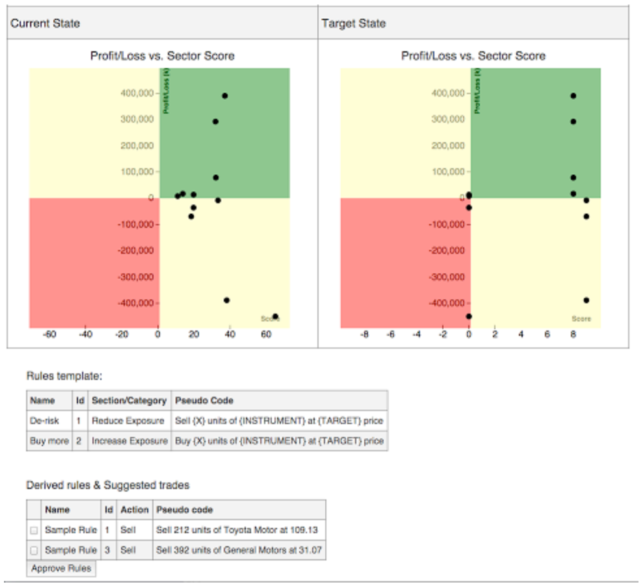
where *StockEMA* = 50 day Stock EMA
SectorEMA = 50 day Sector EMA
SentSc = Sentiment Score

The proportions for stock score, sector score and sentiment score, are defined by the portfolio manager as part of the application setup process. Higher overall scores are safe. The weights of the business parameters within a system can be adjusted by the business user, thus giving him the fine-grained control over the outcomes. In the current application the weights have been assumed as 0.5 for the Stock prices, 0.25 for Index and 0.25 for Sentiment analysis results, respectively. These weights can be different for a different application.

The calculations of scores for a particular stock (EMA, Sentiment, sector scores, overall score) are done using Map-Reduce on the data from MongoDB.

When editing rules, the system can also suggest conditions that have been used in the past, and have had a high impact on the portfolio. The best three conditions are shown as suggestions.

Before deciding on the deployment of a particular rule, the business user can visualize the effect of applying it on the portfolio. This kind of what-if scenario is visualized using a *Current* vs. *Target State* scatter plots, of profit/loss versus sector score (see Figure 8). The target state is plotted using the portfolio structure that would shape-up once the trade rules are executed.



Rules template:

Name	Id	Section/Category	Pseudo Code
De-risk	1	Reduce Exposure	Sell {X} units of {INSTRUMENT} at {TARGET} price
Buy more	2	Increase Exposure	Buy {X} units of {INSTRUMENT} at {TARGET} price

Derived rules & Suggested trades

	Name	Id	Action	Pseudo code
☐	Sample Rule	1	Sell	Sell 212 units of Toyota Motor at 109.13
☐	Sample Rule	3	Sell	Sell 392 units of General Motors at 31.07

Approve Rules

Figure 8. What-if scatter charts showing the risk of the stocks in a portfolio

A scatter chart is divided into 4 zones: left bottom quadrant is the high-risk zone, the top right quadrant is the ideal desired zone that is the safe zone, the left top and right bottom quadrants are the zones of moderate risk. The high-risk zone denotes area of loss and negative score. The safe zone denotes the area of profit and positive score. The moderate risk zones have either the Profit/Loss or the Sector Score as negative values. Stocks mapped in the high-risk zone are prone to higher risk; hence, stocks present in this zone are to be exited. The stocks present in the safe zone are the ones that would yield the best returns within the current portfolio. It would be a good idea to reduce the current holding of stocks present in a moderate risk zone, when the near term trend is negative.

All these actions help in reducing risk. The idea of the *Target State* is to move the portfolio into the top right quadrant zone, by adjusting the portfolio distribution and exposure to risk, hence resulting in minimized loss and better returns. This movement of stocks is based on the following algorithm that identifies the appropriate rules to do it:

1. Calculate a + /– range from the Overall score, based on the average movement of the sector index in the

preceding month (for example, if the sector index for Automotive had a low of 50 and a high of 70, the expected variations will be +/- 10).This range will determine the target state upper & lower limits.

2. Calculate exposure risk

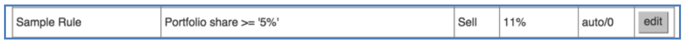
   *exposure risk = SectScore * AvgStockPr * OverallVal *100* (4)

   where *SectScore* = Sector wise Score
   AvgStockPr = Percentage of the individual average stock price for the week
   OverallVal = Percentage of the overall stock value held

3. Based on the exposure risk value, retrieve stop loss rules (where price is below purchase price, or where there has been a consistent negative trend)
4. Derive new trade rules based on the stop loss rules retrieved in step 3.

VI. Rules Deployment, Execution and Feedback

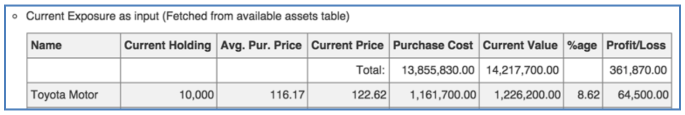
For each rule in the system, the conditions are run against the data of each stock in the portfolio. If there is a match between a rule condition and a stock data, then the action part of the rule is matched against the action parts of a set of rule templates. Assuming that a rule is matched to a rule template, a new trade rule is generated, and will be placed to the stock exchange (executed in Nools). For example, if we start with the rule in Figure 9,



Sample Rule	Portfolio share >= '5%'	Sell	11%	auto/0	edit

Figure 9. Sample rule

a possible match from the current portfolio, could be the Toyota Motor stock, because its portfolio share is 8.62% (as seen in Figure 10), and therefore is greater than 5%.

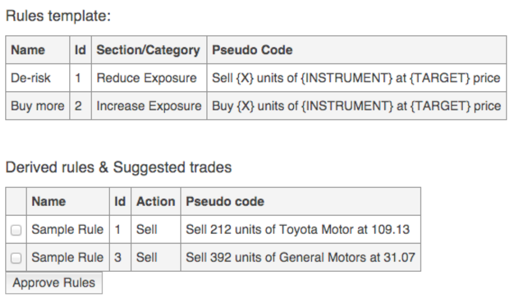


∘ Current Exposure as input (Fetched from available assets table)

Name	Current Holding	Avg. Pur. Price	Current Price	Purchase Cost	Current Value	%age	Profit/Loss
			Total:	13,855,830.00	14,217,700.00		361,870.00
Toyota Motor	10,000	116.17	122.62	1,161,700.00	1,226,200.00	8.62	64,500.00

Figure 10. Snapshot of the current portfolio showing Toyota Motor

A rule template that matches the action of the Sample rule (sell), could be the Reduce Risk rule template "Sell {X} units of {INSTRUMENT} at{TARGET} price", as in Fig. 11:



Rules template:

Name	Id	Section/Category	Pseudo Code
De-risk	1	Reduce Exposure	Sell {X} units of {INSTRUMENT} at {TARGET} price
Buy more	2	Increase Exposure	Buy {X} units of {INSTRUMENT} at {TARGET} price

Derived rules & Suggested trades

	Name	Id	Action	Pseudo code
☐	Sample Rule	1	Sell	Sell 212 units of Toyota Motor at 109.13
☐	Sample Rule	3	Sell	Sell 392 units of General Motors at 31.07

Approve Rules

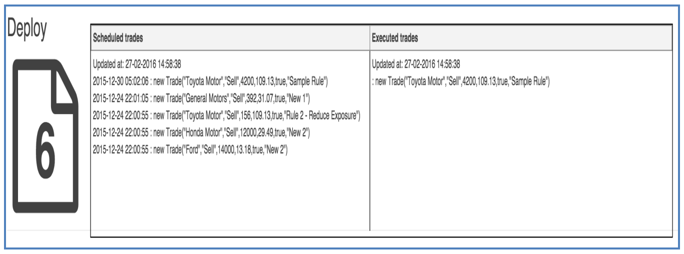
Figure 11. Rule template

In the template, the variables "{X}", "{INSTRUMENT}" and "{TARGET}" are bound to specific values for *quantity, stock name, and price* respectively, as follows: for quantity, as the portfolio share of Toyota Motor is 8.62, applying the rule, it should be brought down to 5% by selling the excess quantity of the shares, so to achieve that 10000 – (10000 * 5/8.62) results in 4200. The price is determined using the price factor set for the matched rule, therefore the current price being 122.62, applying a price factor of 11%, the resulting sell price is 122.62 – (122.62 * 11/100) = 109.13. The stock name is bound to Toyota Motor. Hence, the trade generated will be "Sell 4200 units of Toyota Motor at 109.13". This pseudo-code is later used to generate the actual trade order:

'*newTrade("ToyotaMotor","Sell",4200,109.13,true,"Sample Rule")*'.

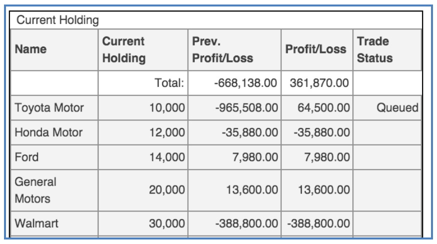
The mechanism in which the newly derived trades are deployed into the running system is handled as part of implementation. The implementation comprises validation of the new rules, adding them to the decision rules table and activating them. The decision rules table is stored in the MySQL database and loaded to the Nools rule engine for rule execution. This implementation provides the major benefit of implementing rules dynamically, without bringing down the system. Moreover, the business user is able to implement new rules much faster, without having to go through the entire IT development cycle, spanning over weeks to months.

Once a trade has been initiated, it shows up in the execution engine window, where the status of the execution engine and trades are shown (see Figure 12).



Scheduled trades	Executed trades
Updated at: 27-02-2016 14:58:38 2015-12-30 05:02:06 : new Trade("Toyota Motor","Sell",4200,109.13,true,"Sample Rule") 2015-12-24 22:01:05 : new Trade("General Motors","Sell",392,31.07,true,"New 1") 2015-12-24 22:00:55 : new Trade("Toyota Motor","Sell",156,109.13,true,"Rule 2 - Reduce Exposure") 2015-12-24 22:00:55 : new Trade("Honda Motor","Sell",12000,29.49,true,"New 2") 2015-12-24 22:00:55 : new Trade("Ford","Sell",14000,13.18,true,"New 2")	Updated at: 27-02-2016 14:58:38 : new Trade("Toyota Motor","Sell",4200,109.13,true,"Sample Rule")

Figure 12. Execution Engine Window



Current Holding

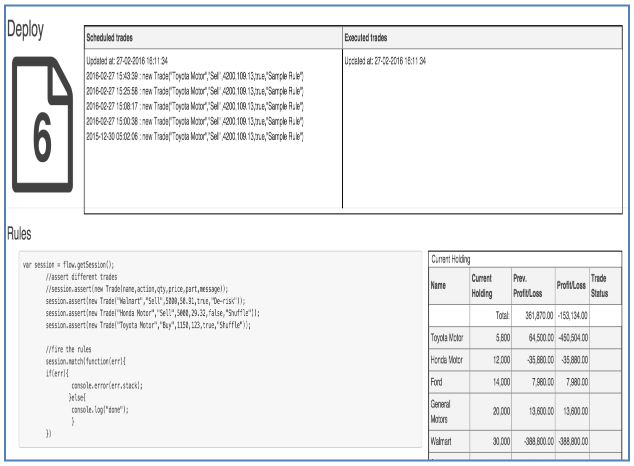
Name	Current Holding	Prev. Profit/Loss	Profit/Loss	Trade Status
	Total:	-668,138.00	361,870.00	
Toyota Motor	10,000	-965,508.00	64,500.00	Queued
Honda Motor	12,000	-35,880.00	-35,880.00	
Ford	14,000	7,980.00	7,980.00	
General Motors	20,000	13,600.00	13,600.00	
Walmart	30,000	-388,800.00	-388,800.00	

Figure 13. Deployment screen showing queued trading orders

Figure 13 shows the profit/loss table with the queued trade orders. As a trade is executed, the profit/loss table of the portfolio gets updated providing up-to date status. The resulting table is shown in Figure 14.

The executed trades move to the upper left section of the execution engine window in Figure 14, while the profit and loss table shows the post execution difference of previous profit & loss vs. current profit & loss side-by-side.

The resulting data is saved into the system's persistent storage for supporting the analytical process that is used for producing better recommendations of rules conditions in the future.



Scheduled trades	Executed trades
Updated at: 27-02-2016 16:11:34	Updated at: 27-02-2016 16:11:34

```
2016-02-27 15:43:39 : new Trade("Toyota Motor","Sell",4200,109.13,true,"Sample Rule")
2016-02-27 15:25:58 : new Trade("Toyota Motor","Sell",4200,109.13,true,"Sample Rule")
2016-02-27 15:08:17 : new Trade("Toyota Motor","Sell",4200,109.13,true,"Sample Rule")
2016-02-27 15:00:38 : new Trade("Toyota Motor","Sell",4200,109.13,true,"Sample Rule")
2015-12-30 05:02:06 : new Trade("Toyota Motor","Sell",4200,109.13,true,"Sample Rule")
```

```
var session = flow.getSession();
        //assert different trades
        //session.assert(new Trade(name,action,qty,price,part,message));
        session.assert(new Trade("Walmart","Sell",5000,50.91,true,"De-risk"));
        session.assert(new Trade("Honda Motor","Sell",5000,29.32,false,"Shuffle"));
        session.assert(new Trade("Toyota Motor","Buy",1150,123,true,"Shuffle"));

        //fire the rules
        session.match(function(err){
        if(err){
                console.error(err.stack);
               }else{
                console.log("done");
                }
        })
```

Current Holding

Name	Current Holding	Prev. Profit/Loss	Profit/Loss	Trade Status
	Total:	361,870.00	-153,134.00	
Toyota Motor	5,800	64,500.00	-450,504.00	
Honda Motor	12,000	-35,880.00	-35,880.00	
Ford	14,000	7,980.00	7,980.00	
General Motors	20,000	13,600.00	13,600.00	
Walmart	30,000	-388,800.00	-388,800.00	

Figure 14. Executed trades and updated profit/loss table

VII. Conclusions

The approach described in this paper allows the business user to interactively and dynamically create or change stop loss business rules using data from a multitude of sources, such as historical trades, market sources, and current portfolio. Before deciding on the best rules to apply on the current portfolio, the business user can visualize the effects of applying a set of rules in what-if scenarios. Once the best rule set is identified, it is immediately deployed for execution on the active data set. The results of the execution are fed back into the system's data storage, to be used in later decision processes. The main beneficiaries of this approach are the executives who feel too dependent on the IT staff for materializing their business ideas into working enterprise systems. With this solution, a business executive is in control of the business, because of the support provided for making informed and timely decisions.

This solution addressed the problem of reducing impending loss in financial trading using open source components to accomplish the task. The ability to use different sources of information, collate heterogeneous data into a common factor, and process it to come up with decision rules, provides an inexpensive solution for businesses of any size.

This paper demonstrates the proposed solution using rules that require simple conditions. In the near future, we intend to extend this work to rules with complex conditions. The rules derived in this paper refer only to Reduce or Increase Risk Exposure rules. However, we want to extend this work with other types of stop loss rules, such as Intra-sector Shift Exposure or Inter-sector Shift Exposure. Also, we intend to extend this solution to other industries, such as retail, or healthcare, and report on the extent of work needed to adapt to new industry models. In this paper we have used only a linear regression algorithm for processing the various types of data. We plan on experimenting with other Machine Learning algorithms, to see which one will provide the best performance for deriving business rules.

Disclaimer - The Company names and financial data are fictional, and used to solely demonstrate the proposed solution. In no way they resemble real company financial information.

References

[1] H. Yang, P.Liang, "Identification and Classification of Requirements from App User Reviews," *SEKE'15,*

[2] E. Guzman, W. Maalej," How DO Users Likes This Feature? A Fine Grained Sentiment Analysis of App Reviews", RE'14, p.153-162.

[3] H. Takahashi, H. Nakagawa, T. Ysuchiya, "Towards Automatic Requirements Elicitation from Feedback Comments: Extracting Requirements Topics Using LDA", SEKE'15

[4] L.V. Galvis Carreno, K. Winbladh, ",Analysis of Users Comments:An Approach for Software Requirements Evolution", ICSE'13, p.582-591, IEEE Press

[5] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B.Mobasher, C. Castro-Herrera, M.Mirakhorli, "On-Demand Feature Recommendations Derived from Mining Public Product Descriptions", ICSE'11, p.181-.

[6] W. Van der Aalst, "Process Mining: Discovery, Conformance, and Enhancement of Business Processes, Springer, 2011

[7] C. Rolland, "Capturing System Intentionality with Maps", in Conceptual Modelling in Information Systems Engineering, p. 141-158, Springer, 2007.

[8] G. Khodabandelou, C. Hug, C. Salinesi, " Mining Users' Intents from Logs", International Journal of Information System Modeling and Design, IGI Global, 2015, Special Issue from the 8th IEEE International Conference on Research Challenges in Information Science (RCIS): 2014, p. 43-71

[9] B. von Hale, "Business Ruels Applied: Building Better Systems Using Business Rules Approach",John Wiley & Sons, New York, 2001

[10] R.Ross, "Principles of the Business Ruels Approach", Addison-Wesley Professional, 2003

[11] Semantics of Business Vocabulary and Business Rules (SBVR), v1.0 OMG, http://www.omg.org/spec/SBVR/1.0/, 2008

[12] P. B. Feuto Njonko, S. Cardey, P. Greenfeld, W. El Abed, " RuleCNL: A Controlled Natural Language for Business Rules Specifications", Volume 8625 of the series Lecture Notes in Computer Science, p. 66-77

[13] L. Nemuraite, T. Skersys, A. Sukis, E. Sinkevicius, L. Ablonskis, " VEtis Tool for Efditing and Transforming SBVR Business Vocabularies and Business Rules into UML & OCL Models", 16th International Conference on Information & Software Technologies, pp. 377-384, 2010

[14] O.C. Tantan, J. Akoka, " Automated Transformation of Business Rules into Business Processes, From SBVR to BPMN", SEKE2014

[15] B. Lucie, " Report on State of the Art and Prospective Evolution of Formal Languages for Business Rules", Public Research Center Henri Tudor, Luxemburg, 2006

[16] D. Rosca, S. Greenspan, C. Wild, " Enterprise Modeling and Decision-Support for Automating the Business Rules Lifecycle", Automated Software Engineering, vol.9, p.361-404, 2002

[17] H.M. Markowitz, "Portfolio Selection". *The Journal of Finance* **7** (1): 77–91., March 1952.