CHAPTER 8

IMPACT ANALYZER

8.1 INTRODUCTION

Most of the tasks saddled on the business analyst have been automated by the Change Management Framework with the help of high level decision making component called Impact Analyzer. This automation has reduced the probability of human errors and the coordination time required by an enterprise and has alleviated the work of the analyst considerably. More importantly, enterprise managers can utilize their time efficiently by focusing on business decisions rather than task management. The final phase of the proposed model is impact analyzer in which the whole functionality is carried out by the component called prediction engine. The purpose of the prediction engine is to evaluate various predictive measures from the run time compatibility measures and impact analysis measures evaluated by the property evaluator and the change evaluator respectively. These predictive measures provide a considerable amount of knowledge, with the help of which the business analyst can make better decisions, and help them further to improve their business by analyzing the risk and estimating the growth benefits associated with the changes.

8.2 IMPACT ANALYZER

The distinguishing feature of this change management framework is the ability to analyze the impact of all the changes made, based on the current change reaction measures and the past incidents. This is accomplished by the impact analyzer component which predicts whether a particular change which is to be embedded to the services will increase or decrease the business growth rate in future and predicts the risk associated with the changes in addition. Thus the business analyst can decide if that particular change request can be incorporated or rejected. One of the special pattern analysis model which performs behavior and growth rate analysis as part of impact analysis is the cellular automata which has been used to predict land usage in GIS systems and to predict traffic in networks so far. Here, the cellular automaton is grabbed to predict the risk and business work flow growth rate by estimating the decidability, manageability and QoS Factor. Decidability determines whether the change request causes any adverse effects on the business logics. Manageability predicts whether a given change request
can be managed within the given time and within the available resources. The QoS Factor
determines whether the change request causes any effect on the quality attributes of the service
logic. The different factors are to be evaluated via the change evaluator and are to be transferred
to cellular automata which will dynamically change its structure and predict the business’ future
with respect to the incoming change request.

The Finite State Machine has several distinguishing and unique immanent features which
have eventually resulted in its accrued importance in the proposed framework. The FSM
exhibits high level of formalism since the business logic source maintains the rules, functions
and parameters. The state transition representation of the FSM with every state followed by a
transition to another state or end state contributes to the FSM being preferred for sequential
processes and the language representation has thus transmuted into logic representation.
Because of the default language handling capability which notifies every new member added to
the language, FSMs have rich applicable theoretical properties. The decidability of the problems
is possible in FSMs with standard algorithmic conversion and automated decision making
which decides if the current change is solvable or not and if the change made is tractable or not.
Further, FSMs map perfectly with the Cellular Automata since every FSM operation is capable
of being invoked by a transition rule in the cellular automata. So the backtracking from the
cellular automata to the FSM can be done efficiently.

Once the FSM is constructed, the business logic set is extracted and the rule or the
function or the parameter in which the change has to be made is found. Then the corresponding
BL schema is generated and dependency analysis is performed to identify all internal and
external dependencies associated with the logic followed by the property evaluation. Then the
change evaluation is done and based on the results from property evaluation and change
evaluation the change impact can be analyzed and the corresponding risk, behavior and business
growth can be predicted. Once the prediction engine decides to commit the change, the run
time manager will build the updated version of service from the BL schema and the service will
be redeployed in the service registry. It is to be noted that an additional support is provided to
the business analysts by calculating the change measures – the factors through which the analyst
can judge whether the change is incorporated correctly and whether it has less risk factor. This
predictive model is implemented via cellular automata and graphs are developed based on the
changes made.
8.2.1 Prediction Engine

Once the change factors and run time properties are evaluated the exact prediction of whether the change request can be managed as an emergency request (manageability) and whether that particular change request contributes to the business growth rate in a worthy manner is found (decidability). When the change request comes, the prediction engine tracks the BL set. The BL set consists of the extracted rules (R1, R2…..RN), functions (F1, F2….FN) and parameters (P1, P2….PN). For each rule, function and parameter, it generates the cell space for pattern generation using properties and change factors. By then the property evaluation process would have got over in the property evaluator, hence it tracks the outputs from the property evaluation engine for the all the 4 properties. Based on the outputs from each property’s execution, it forms a 4 tuple binary value input \{An, Tn, Cn, CYn\} for the manageability automaton and the other property based patterns and rule based patterns. The An is the accessibility, Tn is the traceability, CYn is the computability and Cn is the Configurability properties respectively in the 4 tuple input.

Once when the manageability is calibrated based on this input, next it performs the impact analysis process over the extracted logics based on the measures of the eight change factors. Impact analysis detects the effect due to a change over the logic. Decidability calibration is then performed based on the impact analysis done over the evaluated change factors. Here an eight tuple input is sent to the cellular automaton for deciding whether a change gives a positive impact over the business. When the two primary manageability and decidability calibration gets over, the framework automatically records the change incidents and periodically generates cellular automaton patterns in various dimensions in support of detecting the future cause on the event of change. Through these patterns, various analyst oriented performance measures such as behavior, risk and growth rate analysis are done.

8.3 CELLULAR AUTOMATON MODELLING FOR MANAGEABILITY

The core process behind the prediction engine is the cellular automaton calibration which involves Manageability and Decidability. Manageability is determined based on the result of four property values which are evaluated in the Property Evaluator.
When a cellular automaton is considered, four main components of it have to be modeled at the initial stage. So in our predictive model the

*Cell space* – is the web service domain in which the emergency change management takes place.

*Cell state* – is either decidability / manageability based on the purpose for which automaton is generated. Here it is manageability.

*Neighboring states* – the neighboring states are the property factors.

**Transition rule 1:**

If \( A_N \land C_N \land CY_N \land T_N = 1 \) then

Change Cell state \( M_N \leftarrow \text{BLUE} \)

\( M_N = 1 // \text{change is manageable} \)

Else

Change cell state \( M_N \leftarrow \text{RED} \)

\( M_N = 0 // \text{change is unmanageable} \)

End if

Von Neumann cellular automaton has 4 neighborhood cells. Since we have a 4 tuple input, Von Neumann automaton is considered for manageability. Transition rule 2 includestransition condition for manageability. It says when all the 4 key run time issues are compatible with the change request; the request is predicted to be manageable by the framework. Even if one among the 4 issues fails, then that particular change request is rejected and sent for exception handling.

The cellular automaton model for manageability is shown in the following Figure 8.1.

**Figure 8.1 Cellular Automata Model for Manageability**
The cellular automaton generation for manageability is shown where $M_N$ is the manageability for 4 change requests respectively.

## 8.4 MANAGEABILITY CALIBRATION ALGORITHM

The algorithm gives the steps through which the manageability is calibrated. Initially the BL set is retrieved by the prediction engine. From the configuration manager the output generative power before and after change is retrieved. If the new output generative power gives some constant, when compared to the old output generative power, then it is decided that the change can be managed with new configuration. Hence the 4th tuple $C_n$ in the 4 tuple input is formulated as 0. If there is a need of any new configuration of name spaces, drivers or driver managers then, the change is manageable for time being and the request with its BL set is forwarded to the exception handler. The exception handler adds the necessary new configurations and forwards the request to the property evaluator, where the property issues are rechecked and sent to the prediction engine. If the 4th tuple is 1, then the state allocated for Configurability changes Blue, says it is compatible for the change which is going to be implemented. Otherwise the neighborhood state remains in red color and the prediction starts tracking output of the next property.

The accessibility manager directly returns a binary signal regarding accessibility of the rules, functions and parameters involved in the extracted business logic. So based on that signal, the neighborhood state (which is allocated for accessibility) changes to blue or remains red.

The computability manager returns output in terms of polynomial time, which says the expected time within which logic is computable. If the returned polynomial time is less than the real expected time, then the computability factor is considered as true, else it is considered as false. Based on this, the computability neighborhood state is changed accordingly.

The traceability manager says whether the extracted logic is fully traceable or not. Based on that output, if the logic is fully traceable, and then the tuple input is given as ‘1’, else it is considered as ‘0’. When all the neighborhood states are calibrated, the transition rule runs over them and determines the cell state manageability $M_N$. If all the neighborhood states are blue then the cell state changes to blue, saying the change request is manageable; else the cell state remains red saying the change is not manageable. From this cellular structure, the analysts
will be known, the runtime incompatibilities that will occur to the service logic, because of a change, even before the change is applied. The pseudo code of the manageability algorithm is given below

```
Algorithm Manageability Prediction (CRN , BR,BF, P, F(x) , G(y) , PRESULT , CN, AN, PT, CYN , TN, MN )
// CRN – Nth change request, BR – Business Rules, BF – Business Functions
// P – Parameters
//G(x) – Generative power before change, G(y) – generative power after change
//PRESULT – output where property evaluation results are stored
// CN – allocated configurability Cell in CA, AN – allocated accessibility Cell in CA
// PT- expected time within which a logic has to finish it’s computation
//CYN - Allocated computability cell, TN – allocated traceability cell
// M_N - manageability
While CRN != NULL
Retrieve {BR_1, BR_2…BR_n}, {BF_1, BF_2…BF_n}, {P_1, P_2…P_n} from BL
Fetch G(x) & G(y) from configurability algorithm
//check for configurability
If F(x) \cap G(y) = C then // the change can be managed with same configuration
Set PRESULT[0] = 0
Set neighborhood state CN< - BLUE
// configurability CN< - 0
Else // New configuration is required
Set PRESULT[0] = 1
Set neighborhood state CN< - RED
//Configurability CN< - 1
Endif
Fetch the output signal form accessibility algorithm
If Signal returns 1 then //Accessibility is TRUE for all set of Access Point
Set PRESULT[1] = 1
Set neighborhood state AN< - BLUE
// Accessibility AN< 1
```
Else // Accessibility is FALSE for all or some or at least one set of Access Point
Set $P_{RESULT}[1] = 0$
Set Neighborhood State $A_N \leftarrow \text{RED}$
//Accessibility $A_N \leftarrow 0$

End if

Fetch polynomial time $P_T$ from computability algorithm
If BL is identifiable in all the three phases then
BL is totally computable
Set $P_{RESULT}[2] = 1$
Set neighborhood state $C_Y_N \leftarrow \text{BLUE}$
// Computability $C_Y_N \leftarrow 1$
Else
Set $P_{RESULT}[2] = 0$
Set Neighborhood State $C_Y_N \leftarrow \text{RED}$
//Computability $C_Y_N \leftarrow 0$
End if

Fetch $\text{Config}_{\text{out}}$ from traceability algorithm
If $\text{Config}_{\text{out}}$ is fully traceable then
Set $P_{RESULT}[3] = 1$
Set neighborhood state $T_N \leftarrow \text{BLUE}$
// traceability $T_N \leftarrow 1$
Else
Set $P_{RESULT}[3] = 0$
Set Neighborhood State $T_N \leftarrow \text{RED}$
//Computability $T_N \leftarrow 0$
End if

// cellular automaton calibration
If $A_N \&\& C_N \&\& C_Y_N \&\& T_N == 1$ then
Alike manageability, decidability also gets calibrated over the cellular automaton. Decidability is determined based on the 8 change factors which are evaluated in the change evaluator. Using these 8 change factors impact analysis process is done based on which the decidability for a single change request is determined. In manageability only 4 factors were considered for calibration, hence a cellular automaton with 4 neighborhood cells were considered. But here for decidability 8 factors are considered which makes the 4 tuple input to 8 tuple input, which increases the need for 4 more number of neighborhood states. Hence a cellular automaton called Moore automaton with 8 neighborhood cells is taken for decidability calibration. The 8 factors taken for consideration are listed below -

Again here,

**Cell space** –is the web service domain in which the emergency change management is taking place.

**Cell state** – is either decidability / manageability based on the purpose for which automaton is generated. Here it is decidability.

**Neighboring states** – the neighboring states are the 8 change measures

**Transition rules**

**Transition rule 2**

\[ D_N = 0 \]

If \(I^{VH} < T^{VH}\)

Change neighborhood change factor states of very higher impact \(\leftarrow\) **BLUE**
Else
Change neighborhood change factor states of very higher impact \(\leftarrow\) RED
And
\[\text{If } I^H < T^H\]
Change neighborhood change factor states of higher impact \(\leftarrow\) BLUE
Else
Change neighborhood change factor states of higher impact \(\leftarrow\) RED
And
\[\text{If } I^M < T^M\]
Change neighborhood change factor states of medium impact \(\leftarrow\) BLUE
Else
Change neighborhood change factor states of medium impact \(\leftarrow\) RED
And
\[\text{If } I^L < T^L\]
Change neighborhood change factor states of lower impact \(\leftarrow\) BLUE
Else
Change neighborhood change factor states of lower impact \(\leftarrow\) RED
And
\[\text{If } I^{VL} < T^{VL}\]
Change neighborhood change factor states of very lower impact \(\leftarrow\) BLUE
Else
Change neighborhood change factor states of very lower impact \(\leftarrow\) RED
If all neighborhood == BLUE
Then
\[D_N = 1.\]
Cell state \(\leftarrow\) BLUE
Else
\[D_N = 0.\]
Cell state \(\leftarrow\) RED
End
The transition rule is based on the impact analysis part of the decidability calibration. It is explained in the next section. The cellular automaton model for decidability is shown in the following Figure 8.3.

![Cellular Automaton Model for Decidability](image)

**Figure 8.3 Cellular Automaton Model for Decidability**

The cellular automaton generation for decidability is shown where \( D_N \) is the decidability for four change requests respectively.

### 8.6 DECIDABILITY CALIBRATION

Decidability is determined based on the impact analysis of 8 change factors which are evaluated in the change evaluation step. For every change request once the change factors are measured, they are considered for impact analysis. The amount of impact (Impact factor) of each change factor on the logics is determined in this step. The generalized formula for determining impact factor is

\[
I_F = \text{impactallowance} \times \sum \text{MaxCfv} \times I_v \times \frac{\text{ConsideredCr}}{\text{TotalCr}}
\]

where, 
- \( I_F \rightarrow \) Impact factor
- \( C_{fv} \rightarrow \) Change factor Value
- \( I_v \rightarrow \) Impact value
- \( C_r \rightarrow \) Change factor

The change factor values (\( C_{fv} \)) are taken from the change measures done in the change evaluator. The impact values (\( I_v \)) for each change factor is the rate of impact over the logic during a change. For every change factor rating is done from 1-5 after a comparative priority based ranking. For each impact value (1-5), a percentage of allowance is only allowed to act on
logic while making a change. For example, if a change factor’s impact value is 5, then it is of higher impact, hence only 40% of its impact is allowed to act on the logic where as if a change factor’s impact value is 1, it’s of lower impact hence 60% of its impact is allowed to act on the logic. This allowance of impact is termed as the threshold in the impact analysis part. As the impact values are classified based their effect on logics, the impact factor determination is also classified according to the rating. Hence the impact factor for the change factors having impact values 5 are calculated together Impact_very_high abbreviated as $I_{VH}$. The impact values determined in the step are compared with the threshold values and if all the change factors values are within the limit of allowance, then that particular change request’s decidability changes to ‘1’.

If the impact factor value is higher than the threshold then the states remain red, denoting that their impacts on logic are higher than the allowed %. If the impact is found to be reasonable for the higher impact level of change factors then, then the next level of impact analysis is done and the neighborhood states are changed accordingly. When all the levels of impact analysis gets done, the final execution of transition rule over the neighborhood states take place to determine the value of the decidability i.e. cell state.

When all the neighborhood states of the decidability Moore automaton are blue the cell state is changed to blue, denoting the change gives a positive impact over the service logics. Otherwise the cell state remains red denoting that the change is not acceptable. The decidability algorithm is given below.

```
Algorithm Decidability_Calibration(CR,CF,CFV, IV, IF, DYN)
For all CR_N do
Determine CFV where CFV includes C_OOE, C_CC, C_SM, C_SV, C_TC, C_SC, C_MF, C_BPE
Select CF with $I_{V} = 5$
//Change factor values with impact values 5 are selected
While CF != 0
Compute IF as $I_{VH}$
```
\[ I_{VH} = \{[(C_{CC} \cdot I_{CC}) + (C_{SM} \cdot I_{SM})]\} \cdot 2/8; \]

End While

If \( (I_{VH} < T_{VH}) \) Then
Select \( C_F \) with \( I_V = 4 \)
//Change factor values with impact values 4 are selected
While \( C_F != 0 \)
Compute \( I_F \) as \( I^H \)
\[ I_H = \{[(C_{BPE} \cdot I_{BPE}) + (C_{SV} \cdot I_{SV})]\} \cdot 2/8; \]
End While

If \( (I_H < T_H) \) Then
Select \( C_F \) with \( I_V = 3 \)
//Change factor values with impact values 3 are selected
While \( C_F != 0 \)
Compute \( I_F \) as \( I^M \)
\[ I_M = \{[(C_{TC} \cdot I_{TC}) + (C_{OOE} \cdot I_{OOE}) + (C_{SC} \cdot I_{SC})]\} \cdot 3/8; \]
End While

Compute \( I_F \) as \( I^L \)
\[ I_L = [(C_{MF} \cdot I_{MF})] \cdot 1/8 ; \]

End While

If \( (I_M < T_M) \) Then
Select \( C_F \) with \( I_V = 2 \)
//Change factor values with impact values 2 are selected
While \( C_F != 0 \)
If \( (I_L < T_L) \) Then
Select \( C_F \) with \( I_V = 1 \)
//Change factor values with impact values 1 are selected
While \( C_F != 0 \)
Compute \( I_F \) as \( I^{VL} \)
\[ I^{VL}: = I_F \text{ Formulae}; \]
End While
If (IVL < TVL) Then
DY_N := 1;
// Decidability is set to 1. Change is acceptable.
Else DY_N := 0;
// Decidability is set to 0. Change is not acceptable.
End If

8.7 QoS CALIBRATION
The cellular automaton structure for QoS factor is shown in the figure 8.5 where R_T represents
the reaction time, A_T represents the availability, U_F represents service usage factor and E_T
represents execution time.

![Cellular Automaton Pattern for QoS Factor](image)

Algorithm QoS Assessment prediction (CRN, BR, BF, P, RT, AT, UF, ET)

// R_T – Reaction time, A_T – Availability, U_F – Service usage factor, E_T – Execution time

Begin
Receive E_T 'From business analysts
Calculate E_T
\[ E_T \leftarrow \Sigma (E_T \text{ of all business functions}) \]

Call Monitoring Process ( )

\textbf{If} \[ E_T < E_T \text{ 'then} \]
Change neighborhood state \[ E_T \leftarrow \text{blue} \]
Receive \[ A_T \text{'From business analysts} \]
Calculate \[ A_T \]
\[ A_T \leftarrow \text{uptime (uptime + downtime)} \]
Call Monitoring Process ( )

\textbf{If} \[ A_T < A_T \text{ 'then} \]
Change neighborhood state \[ A_T \leftarrow \text{blue} \]
Receive \[ R_T \text{'From business analysts} \]
Calculate \[ R_T \]
\[ R_T \leftarrow \text{Current Response time of the logic} \]
Call Monitoring Process ( )

\textbf{If} \[ R_T < R_T \text{ 'then} \]
Change neighborhood state \[ R_T \leftarrow \text{blue} \]
Receive \[ U_T \text{'From business analysts} \]
Calculate \[ U_T \]
\[ U_T \leftarrow \Sigma (\text{Success rate per sessions, per page + hits}) \]
Call Monitoring Process ( )

\textbf{If} \[ U_T < U_T \text{'then} \]
Change neighborhood state \[ U_T \leftarrow \text{blue} \]
\textbf{Else}
Change neighborhood states \[ U_T \leftarrow \text{red} \]
\textbf{End if}

\textbf{Else}
Change neighborhood state \[ R_T \leftarrow \text{red} \]
\textbf{End if}
8.8 SUMMARY

This chapter has dealt with the Impact Analysis supported by the Business Logic Evaluation Model. The key component of the proposed model is the impact analyzer whose whole functionality is carried out by prediction engine. The purpose of the prediction engine is to evaluate various predictive measures from the run time compatibility measures and impact analysis measures done by the property evaluator and the change evaluator respectively. On evaluation of the properties, the manageability is checked and on evaluation of the change factors, the decidability is checked. These measures and the Change Management Framework appear lame in the absence of the predictions described here. Since an efficient Change Management aims at providing the ease of making consistent changes where any future conflicts and issues are foreseen before the change is committed, an efficient prediction engine which analyses the measured outcomes of the property evaluator and the change evaluator and evaluates the predictive measures is required to make the Change Management Framework meet the needs of the Analysts in a better fashion. These predictive measures give a good amount of knowledge transfer and help the analysts to improve their business by incorporating the continuous incoming changes efficiently.