MONITORING A CRICKET MATCH

IIT Madras
April 3, 2016

1. Introduction

Various sensors at Accenture used for monitoring Project Development Cycle will record and report the occurrences of various events like over budget, deadline missed, resources overused etc. These recordings can be both numeric and textual with which we try to monitor the project development and performance by a team at Accenture.

We believe this is similar to monitoring the performance of the team on strike in the second innings of a cricket match. The team has a target to chase, and at any given point, its resources, used and remaining, are wickets and number of overs. The simplest thing to do is to keep track of the required run rate, and perhaps make a recommendation - which could, for example, be about which batsman should come in next, or whether a player is doing well by “rotating strike”. Such observations and recommendations will also draw upon other information, for example historical strike rate of a player, current form of a player, the state of the pitch and whether the first team is above or below par (as decided by some commentator), the number of overs remaining for key bowlers etc. Such a system could be built given the numerical data associated with each ball as well as the historical data. A more sophisticated system, on the other hand, could extract more information from the text of the commentary, for example when catches are dropped, a ball is mis-fielded, an overthrow, or a change in field placement. Such a system needs data not only in the numeric form, but also in the form of spoken / written text about the match, as well as from the contextual information.

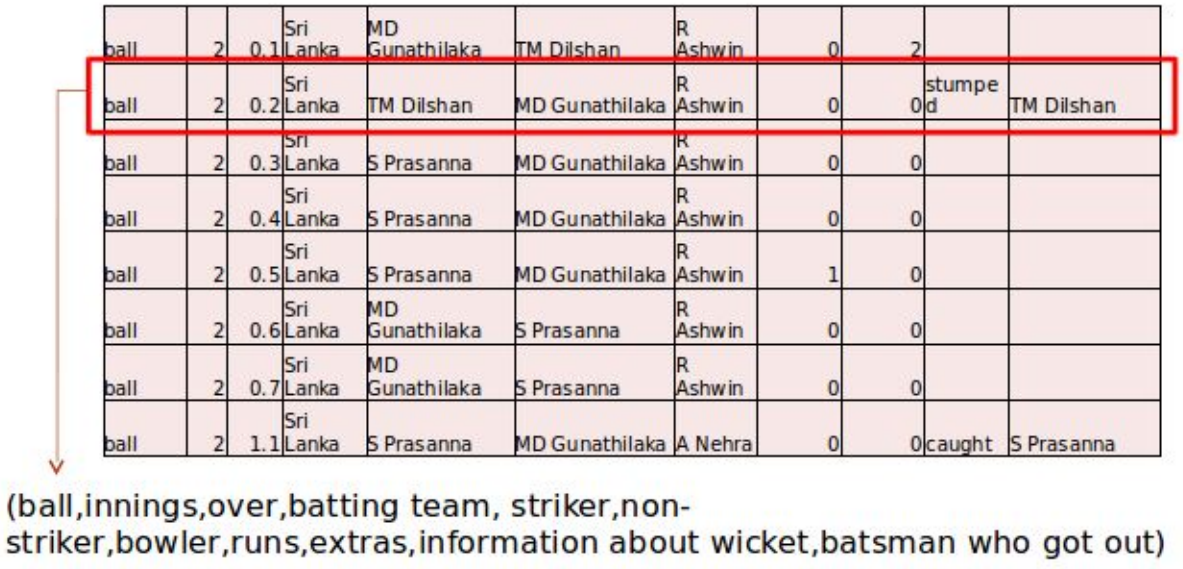
The output of a monitoring and recommender system could be of different kinds. One would be to aggregate numerical data into higher level patterns (like generating weather reports), for example the current run rate is too slow or good. The second would be to comment upon decisions like who is bowling, who is the next batsman, and if one can make out - whether a bowler is bowling to his field etc, and also make recommendations. The third would be to generate a post match summary - something that will draw upon the pitch report, past statistical data for players, identifying specific events like a player getting hurt, the performance of the first team, and in general a summarization of what happens in the 120 (plus extras) balls that constitute the innings of a T20 match. (This is similar to the idea drawn from the last write-up received from Accenture discussing about the various forms of text output expected from the system.) A more sophisticated system could draw upon more historical data about form and performance, and perhaps records being set, but that may be less relevant in the context of project health monitoring.

Following the Reiter and Dale proposed architecture for NLG system as described in the previous write-up, we discuss below further about the acquisition of the input data and about the first stage of document structuring.

2. Input Data Acquisition

Modelling a system that involves dynamic response, requires a time series data as the input. In the cricket domain, it is the data about the statistics and events that occur at each ball. Cricsheet.org is an online

repository that provides a structured ball-by-ball data for Test matches, ODIs and T-20s. In our system, we have used ball-by-ball data of several T-20 matches obtained from Cricsheet.org. An example snapshot of such data is shown below.



ball	2	0.1	Sri Lanka	MD Gunathilaka	TM Dilshan	R Ashwin	0	2		
ball	2	0.2	Sri Lanka	TM Dilshan	MD Gunathilaka	R Ashwin	0	0	stumped	TM Dilshan
ball	2	0.3	Sri Lanka	S Prasanna	MD Gunathilaka	R Ashwin	0	0		
ball	2	0.4	Sri Lanka	S Prasanna	MD Gunathilaka	R Ashwin	0	0		
ball	2	0.5	Sri Lanka	S Prasanna	MD Gunathilaka	R Ashwin	1	0		
ball	2	0.6	Sri Lanka	MD Gunathilaka	S Prasanna	R Ashwin	0	0		
ball	2	0.7	Sri Lanka	MD Gunathilaka	S Prasanna	R Ashwin	0	0		
ball	2	1.1	Sri Lanka	S Prasanna	MD Gunathilaka	A Nehra	0	0	caught	S Prasanna

(A snapshot of the data from cricsheet.org)

2.1 THE FUZZY INFERENCE SYSTEM

To model the alert generation for the cricket domain, we have implemented a fuzzy inference system that reflects the changes in the cricket match at each time stamp.

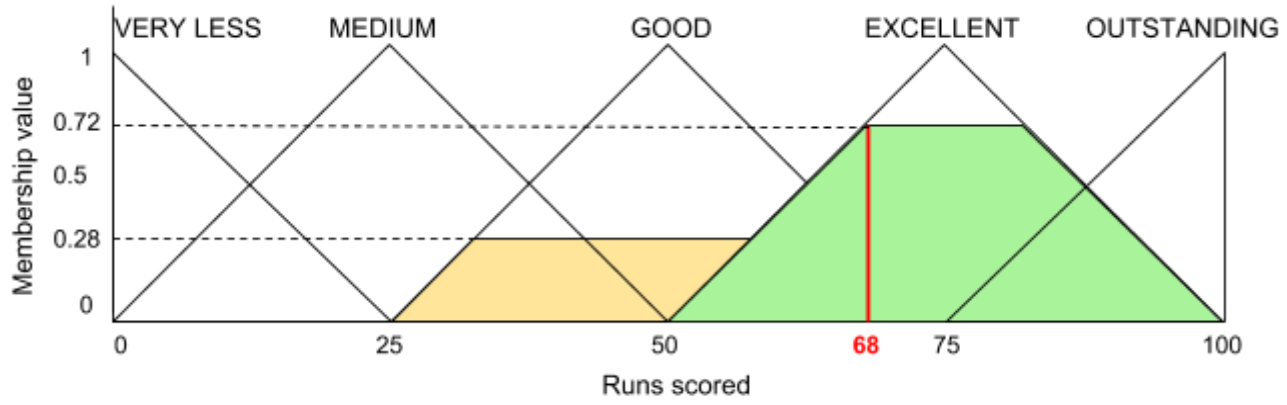
The input parameters to the fuzzy inference system are the strength of the team, the ratio of the required run rate to the current run rate (a measure that captures how well the team is performing), the number of wickets left and the number of balls left. The output of the system is a value that indicates the health of the team depending upon the resources left. The strength of the team is initially set to 100. With each wicket loss, the strength of the team is reduced by a factor proportional to how important that wicket was. The importance of the wicket is measured using another fuzzy inference system, that takes into account the runs scored by the batsman and the balls faced by the batsman and gives the importance of that wicket on a scale of [0..1]. The alert can be generated at those situations where the health of the team is falling below a threshold value.

Membership functions and rules for calculating the impact of wicket lost

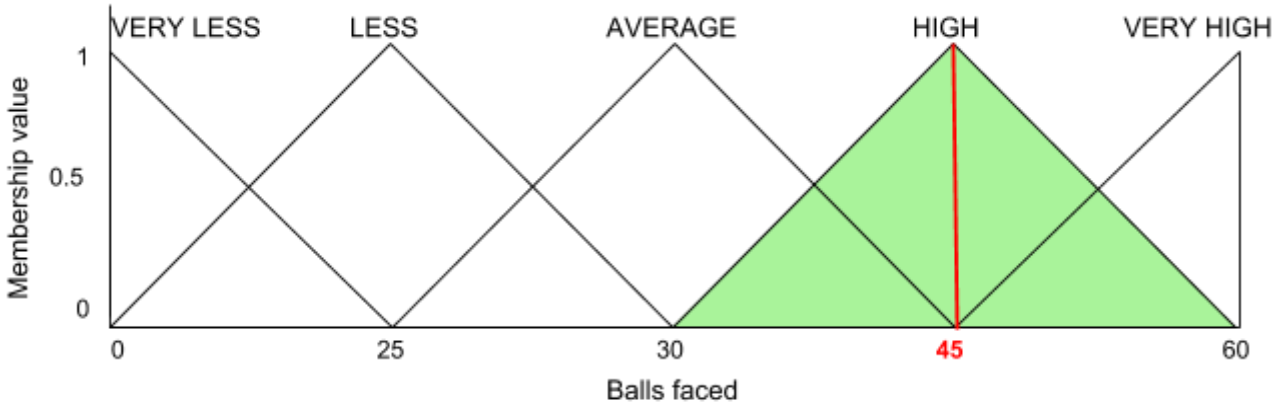
As mentioned earlier the impact of a wicket loss is computed on the basis of two input parameters, the runs scored by the batsman and the number of balls faced. These two parameters are modelled as fuzzy input variables and five membership functions have been defined for both the variables.

Based on these input variables, we have written nearly 25 fuzzy inference rules to compute the value of output variable(impact). An example of triggering of fuzzy rules and computation of value of impact is shown below:

Consider a wicket loss has happened. The batsman who got out scored 68 runs from 45 balls.



(Figure 2.1: The figure shows the triangular membership functions for the fuzzy input: Runs scored by the batsman. Here for the input 68, the membership value for EXCELLENT is 0.72 and for GOOD is 0.28)

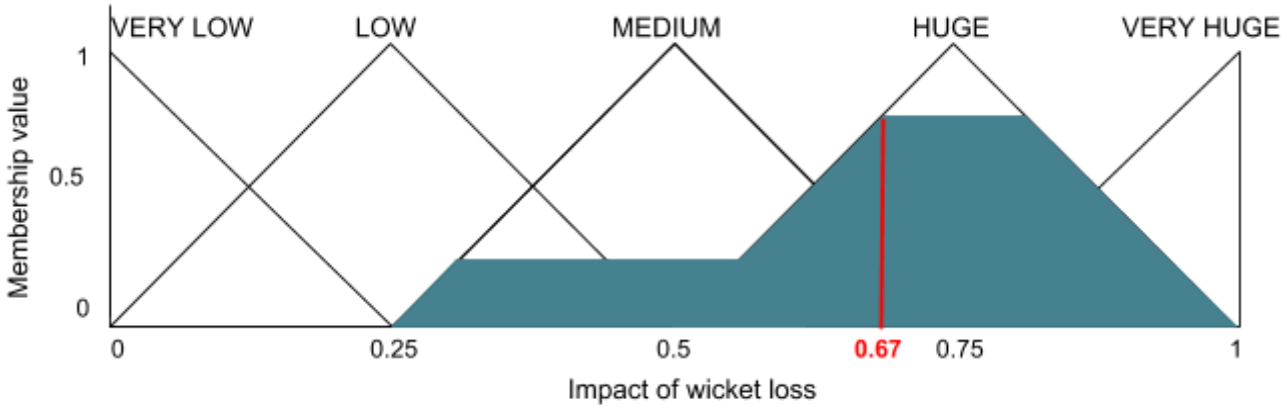


(Figure 2.2: The figure shows the triangular membership functions for the fuzzy input: Balls faced by the batsman. Here for the input 45, the membership value for HIGH is 1)

So.each input value gets transformed in a tuple of five values as in above examples of 45=(0,0,0,1,,0) and 68=(0,0,0.28,0.72,0) ,This process is called fuzzification.

For this particular input, from the set of rules in our system, the following two rules were triggered:

1. IF RUNS SCORED IS **GOOD** AND BALLS FACED IS **HIGH**, THEN IMPACT IS **MEDIUM**
2. IF RUNS SCORED IS **EXCELLENT** AND BALLS FACED IS **HIGH**, THEN IMPACT IS **HUGE**



(Figure 2.3: The figure shows the triangular membership functions for the fuzzy output: impact of wicket lost.Here the variable 'impact' has values in the membership functions MEDIUM and HUGE which is defuzzified(by centroid of the area method) to a crisp value of 0.67, which is the final output value)

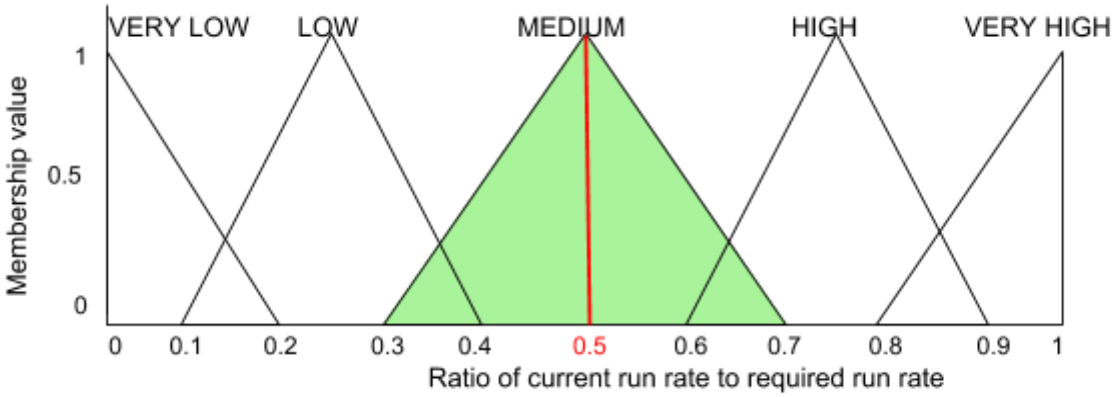
So the above fuzzy inference system outputs a value of 0.67 as impact of wicket loss for an input of runs scored by batsman equal to 68 and balls faced equal to 45.

Membership function and rules for calculating the health of the team

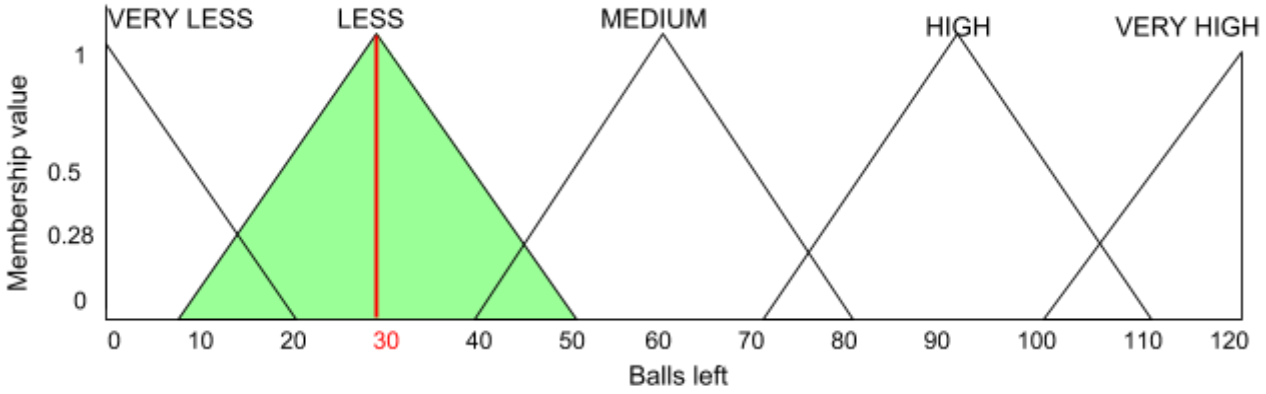
The health of the team is computed on the basis of four input parameters, the ratio of current run rate to required run rate, the number of balls remaining and the number of wickets left. The fourth parameter is a measure named the wicket strength that is used to capture the effect of wicket loss. Initially the wicket strength is set to a high fixed value, and it diminishes with every wicket loss. These four parameters are modelled as fuzzy input variables and corresponding membership functions have been defined for each of them.

Based on these input variables, we have written nearly 35 fuzzy inference rules to compute the value of output variable(health). An example of triggering of fuzzy rule and computation of value of health of team is shown below:

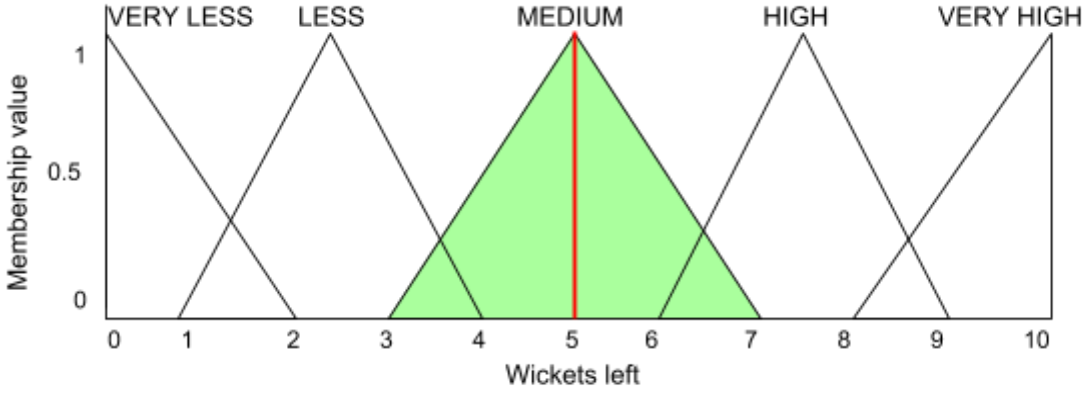
Consider a scenario where the health of the team is 75(out of 100), the ratio of current run rate to required run rate to current run rate is 0.5, the balls left is 30, and wickets left are 5.



(Figure 2.4: The figure shows the triangular membership functions for the fuzzy input: Ratio of current run rate to required run rate. Here for the input 0.5, the membership value for MEDIUM is 1)



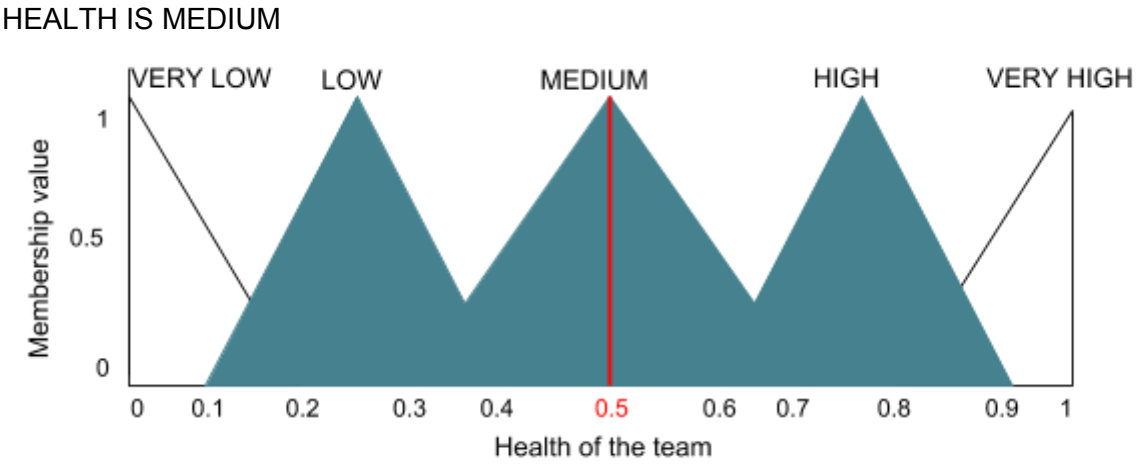
(Figure 2.5: The figure shows the triangular membership functions for the fuzzy input: Number of balls left)



(Figure 2.6: The figure shows the triangular membership functions for the fuzzy input: Number of wickets left)

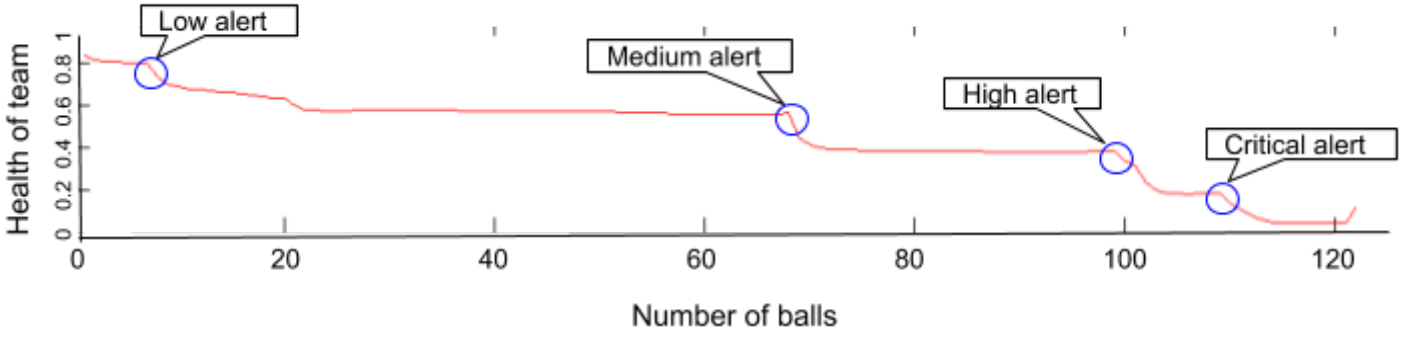
For this particular input, from the set of rules in our system, the following rule was triggered:

1. IF REQUIRED_RUN_RATE/CURRENT_RUN_RATE IS MEDIUM AND THE WICKETS LEFT IS MEDIUM AND THE BALLS LEFT IS MEDIUM AND THE WICKET_STRENGTH IS MEDIUM, THEN



(Figure 2.7: The figure shows the triangular membership functions for the fuzzy input: Health of the team. With the given inputs, the final defuzzified crisp value obtained was 0.5)

We also analysed how the output(health of the team) varies with the input graphically with the help of the following plot.



(Figure 2.8: The figure shows how the health of the team changes as the match progresses. The data used here is that of a team batting in the second innings ,losing the match. The alert points in the graph corresponds to those situations in the match where there has been a low current run rate to required run rate ratio or where there was consecutive wicket losses or loss of a good wicket)

3. Document Planning

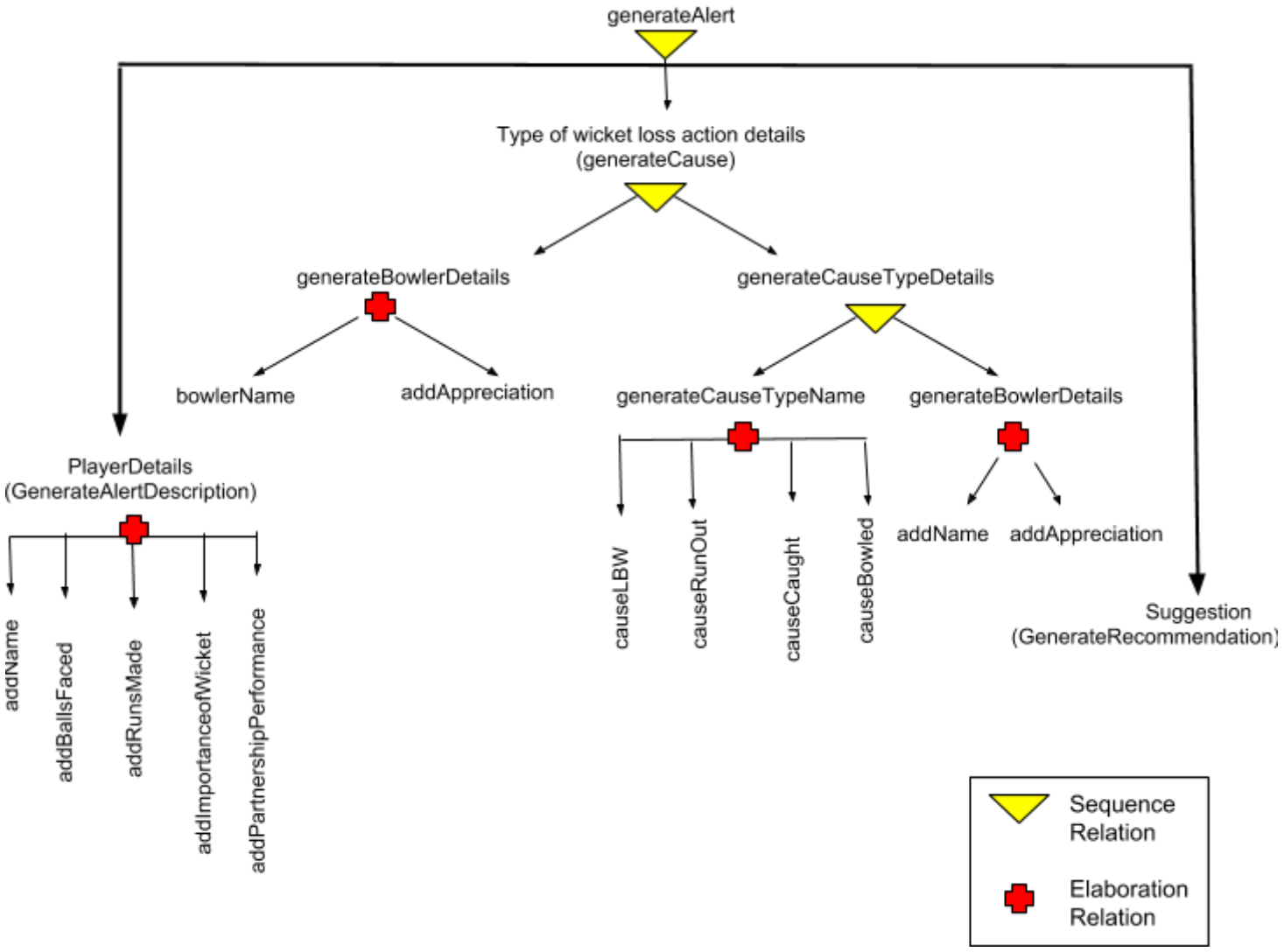
The system needs to decide what information needs to be communicated based on the communication goal and how this information should be presented in the form of alerts or a summary. This brings us to the document planning - the first stage of NLG system architecture, followed by Microplanning and Surface Realization. It consists of two major tasks: Content Selection and Document Structuring.

3.1 Content Selection:

Content selection is the task of deciding what chunks of information should be included in the generated alert, cause(s) and recommendation(s). Currently, the system generates an alert if there is a loss of wicket or if the current run rate is going below the required run rate. When the NLG system reads that the fuzzy value of the alert is above a threshold or is ‘high’, then, it compares the data with the previous ball data to conclude the cause of alert - ‘Wicket Loss’ or ‘Low Run Rate’. For the generation of alert, we already have a pre-defined pattern of generation which is <alertName><Cause><Recommendations> as read from the excel sheet obtained from Accenture. Accordingly, the content is fetched from the three sources of data available with us: (i) data recorded during the match, (ii) output from the fuzzy system, and (iii) a few textual attributes corresponding to the course of action based on the document structures for the events. These message values are set in the corresponding document structure values after recognising the type of alert, as discussed in the next section of document structuring.

3.2 Document Structuring:

Document Structuring is the task of building a structure that contains the content selected in the stage of content selection. We need to group the content which needs to be conveyed for which we specify the order in which the text appears. This is done by specifying discourse relations like Sequence, Narrative Sequence, Elaboration, Exemplification etc. Further, we come up with a tree whose leaf nodes specify the messages and the internal nodes specify the structural information such as how messages are grouped and the discourse relations that hold between messages or groups of messages. This will define the structure of the binding of the alert, its cause(s) and recommendation(s).



(Figure 3.1: A document plan corresponding to the alert caused due to a wicket loss.)

Further extension: We also have commentary scripts available for different events during the match from introduction of the pitch to declaring the man of the match. We plan to extract events like catch dropped, striker missed the shot etc. from the commentary scripts to generate a summary similar to the one which reads like a human-written summary. This is similar to the textual input which our system at Accenture can receive from the sensors. For example, a manager writing as a review, “Resources Overused.” So, we will then, be playing with both sets of input: numerical and textual because we want to build the closest analogous system which simulates the nature of the Accenture system.

4. Correspondence with Accenture

The system getting developed based on the Cricket information is similar to the one expected by Accenture. The monitoring of testing artifact rate or debugging rate is similar to monitoring the run rate. Since, only the second innings data is taken into consideration, we have a clearly defined target for the team on strike similar to the process of assigning a deadline to a project or a team. The various resources for cricket namely, wickets

left, runs scored, balls left etc. are like the various resources under consideration during the development of a software project. For example the alert of ‘over budget’ is similar to alert of ‘wicket-loss’ in our system. The corresponding recommendations like the deliverables plan to be rechecked are similar to the recommendations generated (*currently hard coded*) by the system like that of “run rate needs to be improved to achieve the far-fetched target”. We are planning to apply Case Based Reasoning in future for making the system to learn recommendations.