RISK BASED TESTING A Fuzzy Approach

Ochin Sharma
Faculty of Engineering and Technology-CSE
Manav Rachna International University
Faridabad , India
[1]ochin.fet@mriu.edu.in

Ameen Yasar Khan
Faculty of Engineering and Technology-CSE
Manav Rachna International University
Faridabad , India
[1]ameen.fet@mriu.edu.in

***Abstract*—Earlier Testers used to concentrate on only functionality, usability or performance sort of testing. Many of these derived by the customer's desire or need. Same thing was with the risk based testing. If an application used to be real time based or crucial in terms or national security or economic then it was considered to be a risk based testing candidate. But business of individual has equal importance for the people. Hence now risked based scenario and testing is a must do strategy. But based on the various factors how will you decide when an event or activity is under risk and should be treated on priority based. In real time projects many activities cannot be decided as 'Yes' or 'No' criteria. For example in a project if any one person is left out of 10 members' team. It cannot be treated as crucial, but if 5 members left within short span of time. This 50% breakdown in work force definitely be consider as critical, so here we can wave a red (priority) signal. So fuzzy approach can be consider as an important deal with respect of risk management**

***Keywords- Risk Based Testing , Fuzzy , Software Testing , Risk Management* , Risk Prioritization**

I. INTRODUCTION

Risk management is the process of measuring or assessing risk and then developing strategies to manage the risk. In general, the strategies employed include avoiding the risk, reducing the negative effect of the risk, minimizing the risks and accepting some or all of the consequences of a particular risk. Risk management in software project management begins with the business case for starting the project, which includes a cost-benefit analysis as well as a list of fall back options for project failure, called a contingency plan [1].

It is not necessary that to organize a test process always a risk is required to keep the main motive. Standard approaches to testing are implicitly designed to address risks. You may manage those risks just fine by organizing the tests around functions, requirements, structural components, or even a set of pre-defined tests that never change. This is especially true if the risks you face are already well-understood or the total risk of failure is not too high. If you want higher confidence that you are testing the right things at the right time, risk based testing can help. It focuses and justifies test effort in terms of the mission of testing itself. Use it when other methods of organizing your effort demand more time or resources than you can afford.

.

II. WHY RISK BASED TESTING

- Software projects are not estimated or planned with acceptable accuracy.
- Software project status reporting is often wrong and misleading.
- Software quality and reliability are often unacceptably poor.
- Executives add major new requirements in mid-development.
- Executives apply harmful schedule pressure that damages quality.
- Risks associated with incorrect and optimistic status reporting
- New requirements are added, but the original estimate cannot be changed.
- Estimating tools are not always utilized on major software projects.
- Pressure to agree to schedules which are completely impossible.
- Pressure to minimize inspections or testing in order to speed up development
- Pressure to add new features very late in the development cycle
- No historical data from similar projects is available for a "sanity check".

III .RISK MANAGEMENT LIFE CYCLE

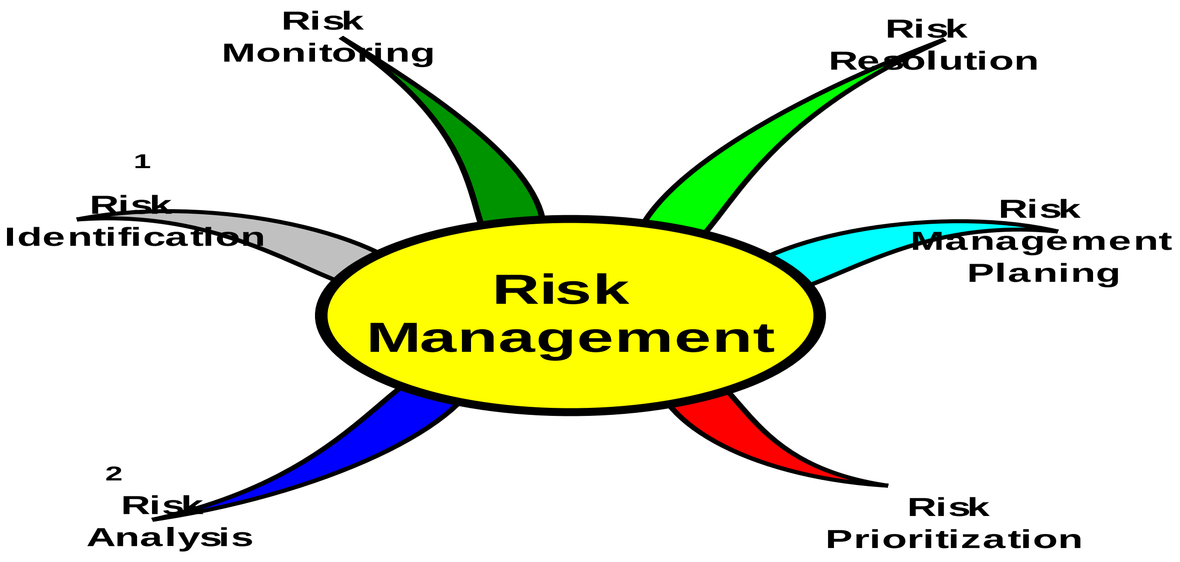

Fig. 1 Risk Management Cycle

A. *Risk Identification*

The Universal Risk Project [Universal, 2002] identifies two basic types of risk statements that can be used to identify whether a set of circumstances represents a risk to the project:

- IF-THEN Statements:

- o "IF technology is not available, THEN we will not meet the requirement"
- o "IF we cannot hire sufficient qualified software engineers, THEN we cannot meet the planned development schedule

- CONDITION-CONSEQUENCE Statements:

- o Given the "condition", there is a likelihood that the "consequence" will occur.
- o "Given that this specific test fails (the CONDITION), the CONSEQUENCE is that the planned schedule will slip".

Subdivided into Development Process; Development System; Management Process; Management Methods; and Work Environment.

B. *Risk Analysis*

After risks are identified, they should be partitioned into categories such as technical, cost, schedule, management, etc. Note that some risks may fall into multiple categories.

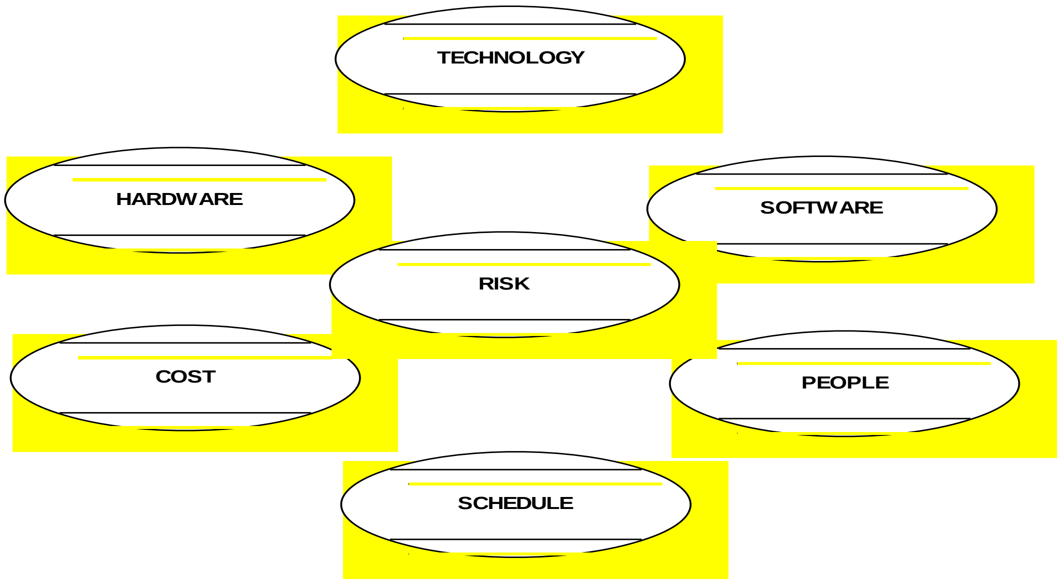

Fig. 2 Risk Analysis Partitioning

Why do risks need to be partitioned? First of all, some risks are more important than others. Also, different stakeholders may be concerned about different risks, or different personnel may bear responsibility for tracking/monitoring different risks. Finally, different risk types may require different mitigation strategies.

The initial activity in risk analysis is to identify contributing factors, then establish a hierarchy of those contributing factors. Figure 2 illustrates a hierarchy of how a project might fail, given the contributing factors of Staffing, Funding, Performance Failures, and so on. The Staffing factor is further broken down to show, first, how staffing may become a contributing factor to project failure and second, what the contributing factors might be that result in insufficient staffing (subsequently leading to project failure). All contributing factors defined within the hierarchy would be broken down to a correspondingly meaningful level of detail.

Similarly, for positive risk, a hierarchy of contributing factors could also be created, this time highlighting those elements for which risk is being undertaken in order to leverage a perceived opportunity for the project, such as "Schedule Completion Will Be Early".

There are any number of ways that risk can be partitioned, analyzed and quantified. The approach taken and method(s) used should always be tailored to meet the needs of the business, the customer and the project.

Graphical tools such as Project Risk Maps or Project Opportunity Maps [Kahkonen, 2001] can also be used to highlight whether specific risks represent a potential negative (risk) or positive (opportunity) impact on the project. These maps highlight (1) where attempts should be made to move a "negative risk" out of the red-shaded risk area (by decreasing their probability of occurrence and/or their degree of impact) and (2) which "positive risks" should be pursued, if possible, into the green-shaded opportunity area (by increasing their probability of occurrence and/or their degree of impact) (see Figure 1). Note that for "positive risks", a.k.a., opportunities, risk mitigation techniques should be defined to maximize the probability that the opportunity will occur (rather than the traditional minimizing the probability that a loss will occur). For our previously discussed positive risk example of using a new software tool to deliver a product earlier than required, a mitigation technique could be to partially outsource the software development effort to someone who is familiar with the tool in question (i.e., no learning curve required), thereby increasing the probability of achieving the opportunity.

Once you have successfully partitioned and quantified project risk, prioritization of risk becomes the next logical activity.

C. *Risk Prioritization*

Risk prioritization is a critical characteristic of the formal risk management process, as it provides the opportunity to apply what are typically limited project resources to those risks having the largest potential impact on the project. For many risk prioritization approaches, risks are ranked and prioritized based on some combination of probability (i.e., how likely is it that the risk will occur) and impact (i.e., what are the consequences if the risk does occur). This can be done qualitatively in a risk prioritization matrix or quantitatively using some type of composite probability-impact score. Both of these basic approaches are illustrated in Figure 1.

D. *Risk Management Planning*

Risk management planning provides the basis for the identification of the monitoring procedures that should be put in place for each risk, including how to tell if a risk is going to manifest as a real problem, and how frequently each identified risk should be monitored. Risk planning also takes into account risk aversion planning (i.e., what actions will be taken to mitigate risk before it occurs) and contingency planning (i.e., how to react if a risk actually manifests).

IEEE Standard 1540-2001 [IEEE, 2001] was introduced earlier in this document. It contains a number of templates that can be used in support of formal risk management planning, including

- Risk Management Plan Outline
- Risk Action Request Outline
- Risk Treatment Plan Outline

The purpose of a Risk Management Plan is to define how risk management activities are implemented and supported during a project. It is a key output of the planning process, and serves as the mechanism for implementing software risk management.

The Risk Action Request serves as a mechanism by which risk information can be captured and communicated to the stakeholders. An effective risk management process requires the creation of risk action requests for any risks that exceed their quantified risk threshold value.

Finally, the Risk Treatment Plan is used to define how risks that are found to be unacceptable are to be treated. It serves as the mechanism for implementing a selected recommended alternative defined within a risk action request. Note that "treatment" and "mitigation" are considered synonymous.

The document "Risk Management Survey of Department of the Navy Programs" [Navy, 1997] contains samples of Risk Management SOW Contract Clauses (Section A); a Risk Status Report format (Section B - illustrated); a product/process combination risk management process (Section B); a technical risk assessment form (Section B); a technical risk status form (Section B); a risk management process and analysis methodology (Section B); an event-driven risk mitigation report (Section C); and risk management survey references (Section D).

E. *Risk Resolution*

The "Software Acquisition Risk Management Guidebook" [Gallagher, 1997] contains a software risk planning decision flowchart (Figure 1) that, among other things, highlights the major options to be considered when developing response plans for threats, namely "Keep", "Delegate" and "Transfer" if personal risk responsibility is appropriate; and "Research", "Accept", "Mitigate" and "Watch" (or "Monitor") if further involvement in risk resolution is warranted.

F. *Risk Monitoring*

There are a number of measurements/metrics and tools that can be used to monitor and report status of project risks. Table 9 presents some of the basic measurements and metrics that can be considered to meet these needs.

The number of identified and active risks provides a picture of how many risks have been mitigated, and how many remain as open items. Prioritized risks can be categorized as the number considered "high", "medium" and "low". Combining risk probability and impact into a total composite score is another way to identify, track and monitor high priority risks. Identifying an "average" probability impact (PI) score subdivides the risk population into "greater than" and "less than" average risks, providing a very general guideline on which risks to address first
There are several indexes that can be used to measure risk, including project criticality, project cruciality, and risk exposure. The Criticality Index identifies which resources (or resource types) are most constrained and how those constraints may impact the project schedule. Activities with a high criticality index are likely to determine the overall project duration. The Cruciality Index, which is similar to the Criticality Index, factors in how critical the resources are to meeting the project schedule constraints. Activities with a high cruciality index have a direct bearing on the variability of the overall schedule duration. The relative Risk Exposure Index, which we discussed earlier, is the probability of unexpected loss (or risk) and the size of the loss. Finally, the Risk Reduction Leverage refers to the corresponding reduction in cost impact or cost of response as a result of reducing the risk level in other areas, such as performance, reliability, schedule, etc. Its value is determined using the equation:

Risk Reduction Leverage = (Risk Exposure Before – Risk Exposure After) divided by Risk Reduction Cost

There are also a number of risk management audit measurements/metrics that are appropriate to monitor and track progress on controlling project risk and relate well to SEI CMM concepts. These include factors such as (1) the number and ratio of scheduled and actual risk management audits, (2) the effort required to support risk management audits, (3) the total number of risk-related problem reports (and the number of open and closed reports) that are measured in each risk management audit, and (4) the number and ratio of actual and deferred corrections that are reported in each risk management audit. This number of corrections can be further broken down into categories of major and minor corrections and, as with the number of actual corrections, can be assessed against the amount of effort required to implement them.

Some general comments about risk monitoring and tracking:

- The frequency with which risks are measured should be dependent on the risk priority (higher priority means measure more often)
- At a minimum, risk measurements should be made/communicated at all key project milestones
- Resist the temptation to perform measurements too precisely or too frequently
- Track all actions to closure (and with demonstrated verification, if possible)

One of the better tools to use in the risk management process is Risk Status Reports. They represent a best practice in that they allow the communication of appropriate risk metrics to all stakeholders on the project. Risk Status Reports can include such things as a listing of the top ten risk items, the number of risks resolved as of the report date, the number of new risk items introduced since the last report, the total number of current unresolved risk items, the presence of unresolved risk items that lie on the project critical path, and assessment of the probable cost of unresolved risk in comparison to the amount of risk reserve. Figure 2 provides an example of how the Navy has used a Risk Status Report architecture to convey the level of risk associated with various elements of a product development process.

II. FUZZY APPROACH

Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory to deal with reasoning that is robust and approximate rather than brittle and exact. In contrast with "crisp logic", where binary sets have two-valued logic, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions.

Fuzzy logic emerged as a consequence of proposal of fuzzy set theory by LotfiZadeh in 1965.fuzzy logic has been applied to many fields, from control theory to artificial intelligence. Ü denotes the set of rational numbers in [0,1]. A fuzzy subset s : $S \longrightarrow [0,1]$ of a set S is recursively enumerable if a recursive map h : $S \times N \longrightarrow Ü$ exists such that, for every x in S, the function h(x,n) is increasing with respect to n and s(x) = limh(x,n). We say that s is decidable if both s and its complement –sare recursively enumerable. An extension of such a theory to the general case of the L-subsets is proposed in Gerla 2006. The proposed definitions are well related with fuzzy logic.

III. PROBABILITY V/S FUZZY APPROACH

Fuzzy logic and probabilistic logic are mathematically similar – both have truth values ranging between 0 and 1 – but conceptually distinct, due to different interpretations—see interpretations of probability theory. Fuzzy logic corresponds to "degrees of truth", while probabilistic logic corresponds to "probability, likelihood" Fuzzy logic and probability are different ways of expressing uncertainty. While both fuzzy logic and probability theory can be used to represent subjective belief, fuzzy set theory uses the concept of fuzzy set membership (i.e., how much a variable is in a set), probability theory uses the concept of subjective probability (i.e., how probable do I think that a variable is in a set). While this distinction is mostly philosophical, the fuzzy-logic-derived possibility measure is inherently different from the probability measure, hence they are not directly equivalent. Bart Kosko argues that probability is a subtheory of fuzzy logic, as probability only handles one kind of uncertainty. He also claims to have proven a derivation of Bayes' theorem from the concept of fuzzy subsethood. Lotfi Zadeh argues that fuzzy logic is different in character from probability, and is not a replacement for it. He fuzzified probability to fuzzy probability and also generalized it to what is called possibility theory.

IV. FUZZY LOGIC IN SOFTWARE TESTING

We identify three aspects of test planning where Uncertainty is present: the artifacts under test, the test activities planned, and the plans themselves. Software systems under test include, among others: Requirements specifications, produced by
requirements elicitation and analysis. Design representations, produced by architectural and detailed design. Source code, produced by coding and debugging. According to MUSE, uncertainty permeates these processes and products. Plans to test these artifacts, therefore, will carry their uncertainties forward. Software testing, like other development activities, is
human intensive and thus introduces uncertainties. These uncertainties may affect the development effort and should therefore be accounted for in the test plan. In particular, many testing activities, such as test result checking, are highly routine and repetitious and thus are likely to be error-prone if
done manually, which introduces additional uncertainty. Test planning activities are carried out by humans at an early stage of development, thereby introducing uncertainties into the resulting test plan. Also, test plans are likely to reflect uncertainties that are, as described above, inherent in software artifacts and activities. Test enactment includes test selection,
test execution, and test result checking. Testenactment is inherently uncertain, since only exhaustive testing in an ideal environment guarantees absolute confidence in the testing process and its results. This ideal testing scenario is infeasible for all but the most trivial software systems. Instead, multiple factors exist, discussed next, that introduce uncertainties to test enactment activities. Test selection is the activity of choosing a finite set of elements (e.g., requirements, functions, paths, data) to be tested out of a typically infinite number of
elements. Test selection is often based on an adequacy or coverage criterion that is met by the elements selected for testing. The fact that only a finite subset of elements is selected inevitably introduces a degree of uncertainty regarding whether all defects in the system can be detected. One can therefore associate a probability value with a testing
criterion that represents one's belief in its ability to detect defects.

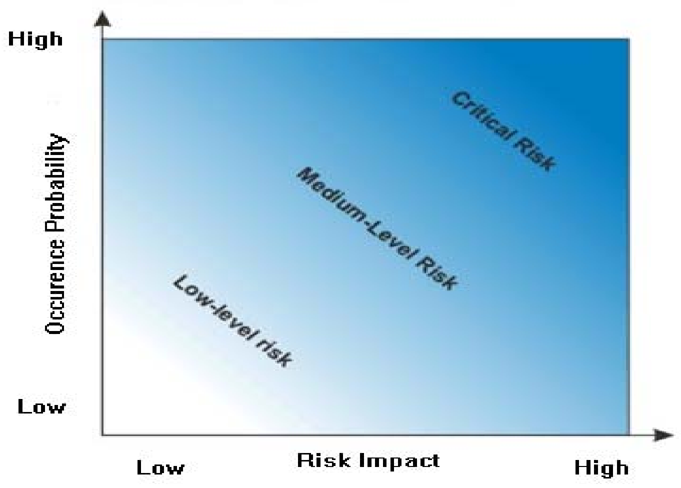

Fig. 3 Associated Probability values with Testing

- **Low impact/Low probability** – Risks in the bottom left corner are low level, and you can often ignore them.
- **Low impact/High probability** – Risks in the top left corner are of moderate importance – if these things happen, you can cope with them and move on. However, you should try to reduce the likelihood that they'll occur.
- **High impact/Low probability** – Risks in the bottom right corner are of high importance if they do occur, but they're very unlikely to happen. For these, however, you should do what you can to reduce the impact they'll have if they do occur, and you should have contingency plans in place just in case they do.
- **High impact/High probability** – Risks towards the top right corner are of critical importance. These are your top priorities, and are risks that you must pay close attention to.

An example of assigning confidence values to path selection criteria is given below.

TABLE I: RISK BASED PRECTION TABLE
A SPECIFIC CASE DEPICTING

RISK CATEGORIES	RISK FACTORS	VALUE	LOW	MEDIUM	HIGH
Personnel Shortfalls	Staffing with top talent,	PS: st < 0.4			H
	job matching,	PS: jm > 0.3	L		
	team building,	PS: tb = 0.5		M	
	key personnel agreements,	PS: kp < 0.8			H
	training,	PS: tg > 0.6	L		
	prescheduling key people	PS: pk = 0.5		M	
Unrealistic Schedules and Budgets	Detailed multisource cost and	USB:dm > 0.3	L		
	schedule estimaton,	USB:se = 0.3		M	
	cost to design,	USB:cd > 0.3	L		
	software reuse,	USB:sr < 0.2			H
	requirement scrubbing	USB:rs < 0..1			H
Developing the Wrong software functions	Organization Analysis,	DWS:oa < 0.2			H
	OOPS Concept formulation,	DWS:oc > 0.6	L		
	user surveys,	DWS:us > 0.7	L		
	prototyping	DWS:pt < 0.4			H
	early user's manuals	DWS:um > 0.9	L		
Developing the Wrong User Interface	Prototyping,	DWI:pt > 0.5	L		
	Scenarios,	DWI:s > 0.3	L		
	Task Analysis,	DWI:ta < 0.8			H
	User Characterization,	DWI:uc > 0.6	L		
	functionality,	DWI:fy =0.5		M	
	style,	DWI:se >0.8	L		
	layout	DWI:lt < 0.3			H
Gold Plating	Requirements Scrubbing,	GP:rs > 0.6	L		
	post benefit Analysis	GP:pba <0.8			H
Continuing Stream of Requirements Changes	High Change Threshold,	CSR:hc <0.7			H
	Information Hiding,	CSR:ih >0.6	L		
	Incremental development	CSR:id =0.8		M	
Shortfalls in Externally furnished components	Benchmarking,	SEC:bm < 0.6			H
	Inspections,	SEC:in =0.5		M	
	Reference Checking	SEC:rc < 0.4			H
	, Compatibility Analysis	SEC:ca >0.2	L		
shortfalls of externally performed tasks	Reference checking,	SEP:rc >0.1	L		
	Teambuilding,	SEP:tb =0.2		M	
Real Time performance shortfalls	Simulation,	RTP:sm >0.7	L		
	Modeling,	RTP:mg < 0.9			H
	Instrumentation,	RTP:im=0.5		M	
	Tunning	RTP:tg< 0.4			H
straining computer science capabilities	Technical Analysis,	SCC:ta > 0.3	L		
	Cost Benefit Analysis,	RTP:ba=0.7		M	
	Prototyping	RTP:pt =0.7		M	

V. CONCLUSION

As uncertainty is inherent and inevitable in software development processes and products. So, software uncertainties should be modeled and managed explicitly. Give its ability to represent differing levels of uncertainty for inputs and outputs whilst still basing inference on the same model, fuzzy logic is well suited to a life-cycle approach to software
development process. The ensuing consistency, communicatability, and economy make this an attractive modeling technique for such applications as effort estimation, requirement engineering and testing. The other advantages of data-free (or datapoor) model building, more robust models, and improved communication further enhance the
opportunities from using fuzzy logic for software metrics.

REFERENCE:

[1] Kahkonen, 2001, Deferring Elimination of Design Alternatives in Object- Oriented Methods, in: Concurrency Practice and Experience, 2000. [2] M. D. Hakel, "How Often is Often", American Psychologist, pp. 533-534, 1968
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] Lorensen, "Object-Oriented Modeling & Design", Prentice-Hall, 1991.
[4] Navy, 1997, Reducing Quantization Error and Contextual Bias problems in Software Development Processes by Applying Fuzzy App.
[5] F. Marcelloni and M. Aksit, Improving Object- *Oriented Methods by Using Fuzzy Logic*.1994
[6] B. Tekinerdogan and M. Aksit, 1998 "Modeling Heuristic Rules of Methods", University of Twente..
[7] B. Tekinerdogan ,2000, Synthesis Based Software Architecture Design, Ph.D. thesis, University of Twente, The Netherlands.
[8] E. Yourdon, 1989 "Modern Structured Analysis", Englewood Cliffs, New Jersey: Yourdon Press,

AUTHORS PROFILE

Ochin Sharma : Assistant Professor Computer Science and Engineering at Manav Rachna International University , has more than 5 years of experience and has software testing as mojor field of interest.

Ameen Yasar Khan : Assistant Professor Computer Science and Engineering at Manav Rachna International University , has more than 3 years of experience and has few other publications in software Testing due to immense area of interest in Software Testing Approaches.