Understanding Software Requirements Using Pathfinder Networks

Udai K. Kudikyala and Dr. Rayford B. Vaughn Jr.
Mississippi State University

Thursday, 22 April 2004
Track 6: 10:15 - 11:00
Room 150 D - F

Understanding and communicating user requirements early in the software development life cycle is essential for satisfying user needs as well as reducing defects, cost, and schedule. This article reports on a technique that uses pathfinder networks to discover and evaluate mental models that represent stakeholders' perception of software requirements. The results obtained by applying this technique on multiple projects are also described.

During the initial phase of any system development activity, software developers are challenged to uncover, understand, and specify user requirements. It is important to have a common understanding among users/customers, project managers, and developers (collectively known as stakeholders) regarding requirements of the software system being developed. This is often considered a major risk factor in software development projects [1]. The sooner misunderstandings are resolved, the more likely developers will build a product correctly [2, 3, 4].

You might consider early requirements in the context of a mental model – or a representation of how a customer or developer thinks about a set of requirements and visualizes them as a whole. Mental models not only model an understanding of the system but also misconceptions that the stakeholder may have [5]. They are also used to understand and communicate what a user actually thinks [6]. Empirical evidence we have obtained indicates that such models may also be useful in identifying misunderstood, duplicate, and ambiguous requirements.

For the past three years, we have experimented with a technique known as pathfinder networks (PFNETs), which are more fully described in the next section. This technique comes from the field of artificial intelligence and one of its properties is the ability to represent knowledge structures as they exist in the minds of humans. This representation is formulated through a facilitator working with one or more subjects who take concepts and group them in terms of their relatedness.

In our research, we used software requirements as *concepts* and applied the PFNET technique to a requirements document separately for both the developer and customer. In both cases, the resulting PFNET was considered a *mental model*, and the two mental models were compared mathematically to determine how close they were. We also discovered this technique was useful in identifying redundant, ambiguous, and misunderstood requirements.

While a complete discussion of the mathematics of the process is beyond the scope of this article, we do provide references and contact information for the interested reader. We have implemented the required mathematics in working software packages and have used the technique on two industrial experiments for real-world projects. In all cases, the PFNET technique was successful in identifying requirements misunderstandings and contributed to a better understanding between the developer and customer early in the life cycle.

> *"Mental models not only model an understanding of the system but also misconceptions that the stakeholder may have. They are also used to understand and communicate what a user actually thinks."*

The techniques we have developed can generally be learned and used in less than eight hours of training. We have not estimated the cost of this training, but believe it to be minimal. We have validated the utility of this technique for small to medium-size software development activities and now intend to publish the results and assist in transferring this technique to industrial use. We are especially grateful to AmerInd[1] Inc. and Nortel for their assistance in using and validating our PFNET work within their software development organizations.

PFNETs

The PFNETs [7] have been used widely to represent knowledge structures. They have been used to model the knowledge of experts and novices in the computer-programming domain [8] and to assess students' knowledge when compared to that of experts [9]. We have successfully applied this technique to the software-engineering domain and, in particular, to the problem of requirements verification and validation. The technique is briefly described in this article, but more detail can be found in [10, 11].

Essentially, we generate a PFNET for the customer/user community and a separate network for the developer community. These networks represent the current *mental model* of the requirements for both communities. By calculating the correlation between the two network models, we can then estimate the degree of common understanding. With additional analysis, we are also able to identify redundant and ambiguous requirements.

The PFNETs' original objective was to generate a network model from *psychological proximity data*, which is the subjective estimate of the closeness or relatedness between requirements as perceived by a stakeholder. The primary goal is to arrive at network representations with nodes representing requirements and links representing relations between requirements. Weights on the links represent the dissimilarity between the requirements. These edge-weights are calculated based on the categorization of information about the requirements that are collected from each stakeholder.

The process of categorization basically means that stakeholders are asked to place individual requirements into categories based on their perceived relatedness. The pathfinder algorithm is applied to concepts (requirements) from the computer science discrete structures domain. Figure 1 (see next page) reveals how the categorization of information about the

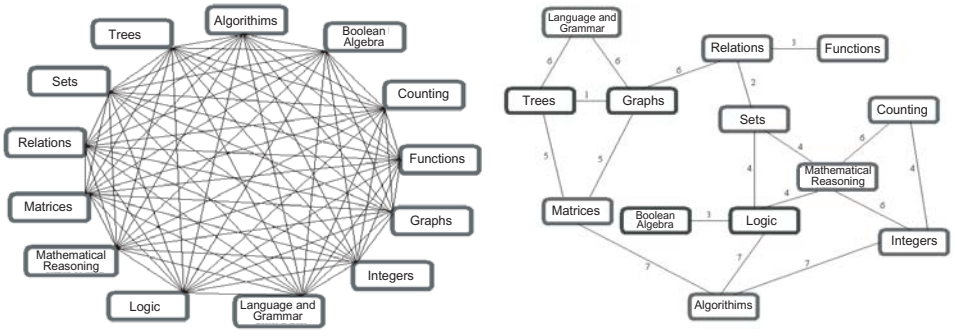


Figure 1: *A Fully Connected Graph With 13 Concepts and the Resulting PFNET With Only 18 Links*

concepts collected from a group of experts results in the generation of a conceptual structure represented by a PFNET.

Essentially, a link exists between a pair of nodes if and only if there is no shorter alternate path between that pair. The initial conceptual network is assumed to be a fully connected graph since no information is available about how the stakeholders relate the list of concepts. Therefore in Figure 1, the original conceptual structure (on the left) is reduced to a more understandable structure (on the right) by removing all links between concepts except for the shortest path or least dissimilar relationship.

Our assumption is that a stakeholder's requirements knowledge consists of the requirements and the interrelationships among those requirements. Figure 2 shows a portion of a PFNET that was generated in one of our experimental projects by the developer. The edge-weights represent the dissimilarity between the pairs of requirements. The lower the edge-weight the lower the dissimilarity and higher the similarity between the two requirements as perceived by the stakeholder.

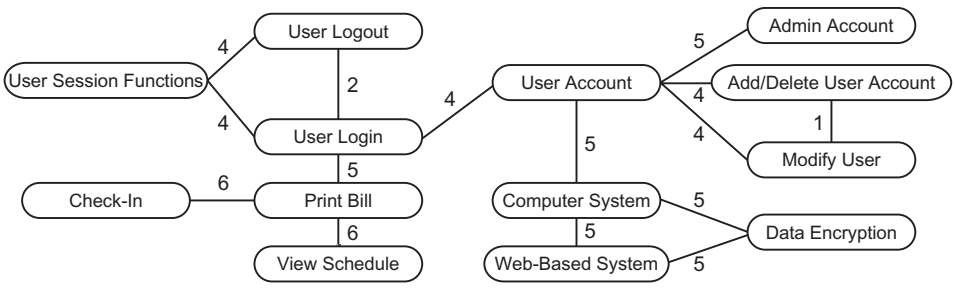
The similarity count for a pair of requirements is determined by the number of times that pair shows up together in the categorization of information collected from the stakeholders. A dissimilarity matrix is computed by subtracting each similarity count from the highest similarity count plus one. The pathfinder algorithm is then applied to the dissimilarity matrix to generate the PFNET. While details of these steps are not provided here, they are implemented in our software. Specifics can be provided to the reader on request.

In Figure 2, most of the developers perceived *Add/Delete User Account* and *Modify User* as closely related. We know this because the link weight is the minimum (least dissimilar), between the two requirements. A number of developers also perceived *User Account* and *Add/Delete User Account* to be related but to a lesser degree. The absence of a link between requirements could mean that very few or none of the developers perceived those requirements as being related. For example, *Admin Account* and *Add/Delete User Account* are not directly connected because the developer perceived *Admin Account* to be more closely related to *User Account* than to *Add/Delete User Account*.

In summary, the primary benefit of the PFNET is the ability to model aspects of human semantic memory. In our case, the ability to model the stakeholders' perception of requirements as graphs is useful. It provides the ability to mathematically evaluate and compare these networks for similarities and dissimilarities to reveal potential misunderstandings.

While this article represents only an overview of the PFNET technique, it is important to note that it is not new – it has been used in other applications outside software engineering for many years. Our work applies this technique to software engineering requirements analysis for the first time.

Figure 2: *Example of a PFNET (Partial) Generated for a Group of Developers*



We generated two PFNETs – one for the customer and one for developers. The categorization of information collected from each developer is combined, and a single consensus PFNET for that group is generated. A similar procedure is used for the end user. We then compare and analyze the two networks to determine the consistency of understanding between the two, as well as determine which individual requirements may not be well understood. Specific results of our application of this technique in projects are presented in the following sections.

Initial Experiment Results

Experiments were first conducted at Mississippi State University (MSU) using students taking a software engineering class [12]. Four real customers agreed to work with the class in order to obtain a needed software product. Students developed a software requirements specification (SRS) for each system, which was the basis for generating all subsequent PFNETs. The following procedures were used (implemented today in software):

- Requirements were individually extracted from each SRS. All participants were asked to read the SRS to become familiar with the requirements. Each student and customer then categorized requirements into groups based on their perceived relatedness. This was achieved by using index cards with requirements printed on each card. The set of index cards and a copy of the SRS were distributed to the customer and the developers working on that project. Instructions were given on the procedure for categorization; none of the participants were allowed to consult with each other during the categorization activity. The categorization activity consisted of grouping the index cards into piles of related categories. The relatedness decision was entirely up to the participant and was very subjective. We refer to the groups of index cards collected from each participant as the categorization of information.
- The categorization of information was then collected and a similarity matrix (N rows by N columns, where N is the number of requirements) was generated for each participant. When a pair of requirements appears together, the count in that cell of the similarity matrix is incremented by one. We refer to the similarity count as a co-occurrence count for that pair of requirements. To compute a consensus (group) PFNET, the corresponding elements of the similarity matrices for that group are simply added

to generate a consensus similarity matrix.
- Dissimilarity matrices were generated by subtracting each co-occurrence count in the similarity matrix from the maximum co-occurrence count plus one (to avoid a zero dissimilarity count in any cell).
- The dissimilarity matrices for the group of developers and the customer were then used as input to the Pathfinder generation program resulting in the generation of two PFNETs.
- The resulting PFNETs are then correlated with each other, producing a correlation coeffcient (cc) that is used to measure similarity between the mental models.

Adding substance to the preceding steps and realizing that several new terms were introduced to the reader, we provide a short example in the online version of this article at <www.stsc.hill.af.mil/crosstalk/2004/05/0405kudikyala.html>.

The cc ranges from -1 through +1, where -1 represents no similarity and +1 represents perfect similarity between the two networks [13]. For all projects, the following heuristics were applied:
- A cc of a network/node below 0.4 indicates little or no similarity.
- A cc from 0.4 through 0.7 indicates a moderate degree of similarity.
- A cc of more than 0.7 indicates very good to strong similarity.

The boundary values we assigned above are subjective and were selected based on empirical evidence. Table 1 shows the overall ccs between the developer and user PFNETs for the four projects in our experiment. In Table 2, each row shows the percentage of requirements with different ccs between developer and user PFNETs for each system developed.

Thus, the higher the value of the cc, the more similar the mental models of the customer and developers appeared to be at the early stages of product development. From Table 2, we can see that severe misunderstanding exists for System 2 and further requirements work is needed. In fact, this observation was validated when, at the end of the actual system development, the user was not satisfied with the final product.

We also seeded the SRS with duplicate requirements to determine if such requirements could be identified using PFNETs. Table 3 shows the ccs based on path distances for each of the original and seeded requirements.

The ccs of the original and seeded requirements were very high when PFNETs for both groups were compared. Further analysis of the PFNETs for each group also revealed that the original and duplicate requirements were directly linked since they were perceived to be closely related. This provided us with evidence that PFNETs may be useful in uncovering duplicate requirements. In addition to the first four experiments, we ran a fifth similar classroom experiment the following academic year and achieved essentially the same results. Together, these five experimental results encouraged us to validate the PFNET approach in industrial settings.

Industrial Experimentation at Nortel

Our experiments were continued at Nortel, Inc., Dallas, Texas, [14] with their assistance over two semesters (about eight months). Our results again indicated that PFNETs were useful to identify misunderstood and duplicate requirements.

The procedure used to apply this technique was very similar to that outlined for our classroom experiments. A Web-based tool was introduced to collect the categorization data from each stakeholder. This tool interface consisted of check boxes to aid the process of categorization as shown in Figure 3 (see next page). Each Web page consisted of a requirement with the description provided at the top of the page. The remaining requirements were displayed on the same page. The stakeholder then checked the boxes of the requirements that

Software Systems	Overall Correlation Coefficient
System 1	0.77
System 2	0.46
System 3	0.91
System 4	0.87

Table 1: *Overall Correlation Coefficients Between Developers and User PFNETs*

Table 2: *Percentage of Requirements By Range of Correlation Coefficients*

Correlation Coefficient (cc) / Systems / Percentage of Requirements	cc >=0.9	cc < 0.9 and cc > = 0.7	cc < 0.7
System 1	43.75	21.88	34.38
System 2	0.00	0.00	100.00
System 3	50.00	50.00	0.00
System 4	50.00	50.00	0.00

Table 3: *Correlation Coefficients of Original and Seeded Duplicate Requirements*

Software Systems	Original Requirement	Seeded Requirement	Correlation Coefficient
System 1	1-13: E-mailing a Student Resumé to a Company	1-30: Sending a Student Resumé to a Company by E-mail	1.00
	1-16: Faxing a Student Resumé to a Company	1-30: Sending a Student Resumé to a Company by E-mail	0.96
	1-12: E-mailing a Student Transcript to a Company	1-31: Sending a Student Transcript to a Company by E-mail	1.00
	1-15: Faxing a Student Transcript to a Company	1-31: Sending a Student Transcript to a Company by E-mail	0.96
	1-11: E-mailing a Letter of Recommendation to a Company	1-32: Sending a Letter of Recommendation to a Company by E-mail	1.00
	1-14: Faxing a Letter of Recommendation to a Company	1-32: Sending a Letter of Recommendation to a Company by E-mail	0.96
System 2	2-2: Add Appointment	2-31: Make Appointment	0.93
	2-3: Change Appointment	2-32: Modify Appointment	1.00
	2-13: Delete Appointment	2-32: Modify Appointment	1.00
	2-20: Modify User	2-33: Add And Delete User	1.00
System 3	3-8: Add a Major	3-39: Modify a Major	0.98
	3-9: Edit a Major	3-39: Modify a Major	1.00
	3-10: Delete a Major	3-39: Modify a Major	0.98
	3-4: Add a College	3-40: Modify a College	1.00
	3-5: Edit a College	3-40: Modify a College	1.00
	3-6: Delete a College	3-40: Modify a College	1.00
System 4	4-23: Display Available Vehicle	4-30: Check Available Vehicle	1.00
	4-5: Delete Reservation	4-31: Cancel Reservation	1.00
	4-9: Make Reservation	4-32: Add Reservation	0.99

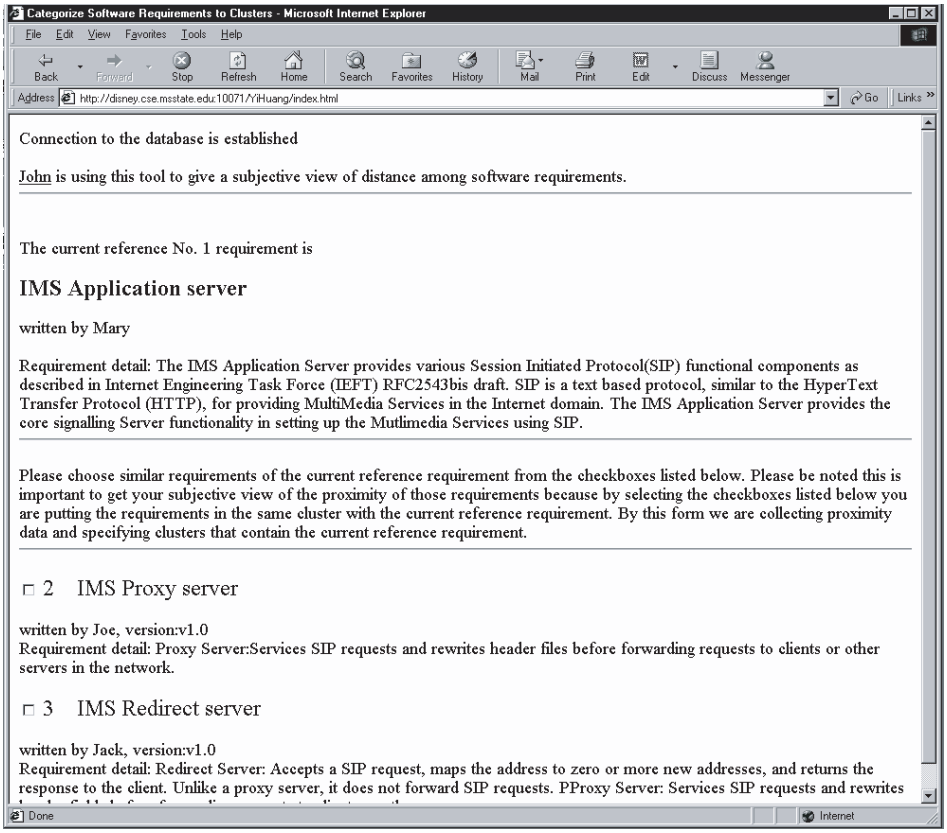


Figure 3: *Web-Based Check Box Interface at Nortel, Inc.*

were related to the requirement provided at the top of the page. After categorizing all the requirements, the information for each stakeholder was submitted over the Internet.

A comparison of customer and developer PFNETs revealed that two requirements (Nos. 6 and 14) had very low ccs, even though the overall cc between the customer and developer networks was 0.88. The shaded rows in Table 4 show the ccs for these two requirements. This demonstrates that the PFNETs in Table 4 were found to be duplicates after facilitating a session between the two groups. In this case, very high ccs (1.00 and 0.99) led to identifying this duplication. High values of correlation may also mean, however, that requirements might well be understood between the stakeholders. Only after further analyzing these requirements can we understand if duplication is present.

Table 4: *Correlation Coefficients of Customer and Developer of Individual Requirements for the Interactive Multi-Media Server (IMS)*

Requirements	Correlation Coefficients
1. IMS application server	0.95
2. IMS proxy server	0.87
3. IMS redirect server	0.99
4. IMS registrar server	0.94
5. IMS location server	1.00
6. IMS external interface	0.05
7. IMS hardware platform	1.00
8. IMS database interface	0.78
9. IMS SIP interface	0.67
10. IMS H.323 client interface	0.74
11. IMS PSTN gateway interface	0.67
12. IMS media gateway	0.79
13. IMS media server	1.00
14. IMS multi-domain support	0.05
15. IMS performance and capacity	1.00
16. IMS BBUA component	0.75
17. IMS application server security	1.00
18. IMS discriminator service	0.98
19. IMS arbitrator service	1.00
20. IMS network handling	0.69
21. IMS call transfer service	0.99
22. IMS call conference service	1.00
23. IMS accounting	0.77

Conclusion

Our initial research work, especially for the student projects at MSU and the project at Nortel, Inc. had generated encouraging results regarding the ability of the pathfinder technique to predict misunderstandings about requirements among customers and developers during the requirements analysis phase.

Further research conducted at AmerInd Inc., validated the usefulness of the technique in a typical industrial setting. The research also showed that this technique is scalable to a medium-scale project like New Material Acquisition. (A full version of this article, including the AmerInd Inc. and New Material Acquisition examples, appears on the CROSSTALK Web site at <www.stsc.hill.af.mil/crosstalk/2004/05/0405kudikyala.html>).

The ccs generated from the PFNETs of the stakeholders identify potentially misunderstood requirements. The values of ccs enable the groups to focus on potentially misunderstood requirements during facilitation sessions. Furthermore, duplicate and ambiguous requirements were also identified. Additional information concerning tools, processes, and experimental results are available from the authors.◆

Acknowledgements

The authors wish to acknowledge the support of the National Science Foundation (Grant #CCR-0303554); the James Worth Bagley College of Engineering at Mississippi State University; Nortel Inc. of Dallas, Texas; and AmerInd Inc. of Alexandria, Va. Without such support, this research and results would not have been possible. We are also grateful to the editorial staff of CROSSTALK for their patience and helpful suggestions to make this report better and more readable.

References

1. Keil, M., et al. "A Framework for Identifying Software Project Risks." Communications of the ACM 41.11 (1998): 76-83.
2. Davis, Alan, et al. "Identifying and Measuring Quality in a Software Requirement Specification." Software Requirements Engineering. Eds. R.H. Thayer and M. Dorfman. Los Alamitos: IEEE Computer Society Press, 1997: 164-75.
3. Boehm, B., and Victor R. Basili. "Software Defect Reduction Top 10 List." Computer 34.1 (2001): 135-37.
4. Faulk, S.R. "Software Requirements: A Tutorial in Software Requirements Engineering." Software Requirement Engineering. Eds. R.H. Thayer and M. Dorfman. Los Alamitos: IEEE Computer Society Press, 1997: 7-22.
5. Caroll, M.J., and J.R. Olson. "Mental Models in Human-Computer Interaction." Handbook of Human-Computer Interaction. Ed. M. Helander. North-Holland: Elsevier Science Publishers B.V., 1989: 45-60.
6. Faro, A., and D. Giordano. From User's Mental Models to Information System's Specification and Vice Versa by Extended Visual Notation. Proc. of the International Professional Communication Conference (IPCC '95), Savannah, GA., Sept. 1995.
7. Dearholt, D.W., and R.W. Schvaneveldt. "Properties of Pathfinder Networks." Pathfinder Associative Networks:

Studies in Knowledge Organization. Ed. R. Schvaneveldt. Norwood, N.J.: Ablex Publishing Corp., 1990: 1-30.
8. Cooke, N.M., and R.W. Schvaneveldt. "Effects of Computer Programming Experience on Network Representations of Abstract Programming Concepts." International Journal of Man-Machine Studies 29 (1988): 407-27.
9. Goldsmith, T.E., and P.J. Johnson. "A Structural Assessment of Classroom Learning." Pathfinder Associative Networks: Studies in Knowledge Organization. Ed. R. Schvaneveldt. Norwood, N.J.: Ablex Publishing Corp., 1990. 241-53.
10. Kudikyala, Udai K., and Rayford B. Vaughn Jr. "Software Requirements Understanding Using Pathfinder Networks: Discovering and Evaluating Mental Models." Journal of Systems and Software (to be published in 2004).
11. Kudikyala, Udai K., and Rayford B. Vaughn Jr. Software Requirements Understanding Using Pathfinder Networks as Mental Models. Proc. of the 2004 ASEE Southeastern Section Conference, Auburn, AL., Apr. 2004.
12. Lu, X. "Using Pathfinder Networks to Analyze and Categorize Software Requirements." Master's Thesis. Mississippi State University, 2000.
13. Goldsmith, T.E., and D.M. Davenport. "Assessing Structural Similarity of Graphs." Pathfinder Associative Networks: Studies in Knowledge Organization. Ed. R. Schvaneveldt. Norwood, N.J.: Ablex Publishing Corp., 1990: 75-87.
14. Yi, H. "Automated Web-Based Tool for Software Requirement Refinement Using Pathfinder Networks." Master's Project. Mississippi State University, 2001.

Note

1. AmerInd Inc., <www.amerind.com> is a medium-sized computer services company located in Alexandria, Va.

About the Authors

Udai K. Kudikyala is a Ph.D. candidate at Mississippi State University and an active research assistant. He has two years of teaching experience. His research interests include requirements engineering, especially modeling and evaluation of requirements understanding using artificial intelligence techniques, and network routing and parallel computing. Kudikyala received a Bachelor of Engineering in electronics and communication engineering from S.R.K.R. Engineering College, Andhra University, Bhimavaram, India, and a Master of Science in computer science from Mississippi State University.

Department of Computer Science and Engineering
Center for Computer Security Research
P.O. Box 9637
Mississippi State University
Mississippi State, MS 39762
Phone: (662) 325-7503
Fax: (662) 325-8997
E-mail: kumar@cse.msstate.edu

Rayford B. Vaughn Jr., Ph.D., is professor of computer science at Mississippi State University. A retired Army colonel, he served 26 years, including commanding the Army's largest software development organization and creating the Pentagon Single Agency Manager organization to centrally manage all Pentagon information technology support. After retiring from the Army, he was vice president of Integration Services, Electronic Data Systems Government Systems. Vaughn has more than 40 publications and actively contributes to software engineering and information security conferences and journals. He has a doctorate degree in computer science from Kansas State University.

Department of Computer Science and Engineering
Center for Computer Security Research
P.O. Box 9637
Mississippi State University
Mississippi State, MS 39762
Phone: (662) 325-7450
Fax: (662) 325-8997
E-mail: vaughn@cse.msstate.edu