ISSN:0975-9646



# Roads Paved for Reuse in Software Engineering.

[#1].**N.Krishna Chythanya**, [#2.] Prof. **Dr.Lakshmi Rajamani**

[#1.]*Asst.Prof.,C.S.E.,G.R.I.E.T.-Hyd,India.*
[#2.] *Retired Professor,C.S.E.,O.U.-Hyd,India*

**Abstract: This is a research paper which through some light on research work carried out by different researchers in the area of RESUABLE COMPONENTS of Software Engineering. As The time has come for institutionalizing of software reuse as an enduring part, Reuse needs to be treated as an integral part of engineering and acquisition activities. A few reusable component repositories have been discussed and also Identification process followed by a set of researchers have been discussed.**

**Keywords: Reuse, component, Reusable component Repository, Software Engineering.**

## 1.INTRODUCTION:

As per the literature , we can consider the discipline of Software engineering to be born when "Software Engineering Crisis" was coined in 1968 at the NATO conference, Germany. The main concern being effective development of very large and highly qualitative software systems.

The time has come for a shift in paradigm from current practices of software engineering and development to a process of software engineering in which institutionalizing of software reuse becomes an enduring part . Reuse needs to be treated as an integral part of engineering and acquisition activities.

A software component is defined as an independent object or the characteristic object that can be deployed or integrated in an application so that the development process will be improved. The collection of all reusable components are stored in a library or repository.

There is no indispensible meaning for a module to be considered reusable . The programming language used has a high influence on the reusability of the component. In general people overlook the need of factor of code that is used for making module robust with proper Exceptional Handling in case of rare input and behavior.

Based on reusability Software Engineering is divided in to Domain Engineering and Application Engineering. Creation and storing of component for reuse is taken care in Domain Engineering where as during Application engineering required component is selected from the reuse library and is used according to the needs or requirements. Different aspects of software reuse explored by research studies and reports over the years are: how does reuse happen, artifacts reused, affects on development cost, quality, languages support, impact of training developers to reuse.

The remaining part of parper is categorized as *Section 2* summarizes Related work done in the direction understanding reuse activity and its goals, *Section 3* deals with processes implemented for constructing different Repositories with a few example repositories and *Section 4* emphasizes work done in component identification and experience of developers in using reusable components. *Section 5* deals with Retrieval mechanisms followed by the researchers and *Section 6* concludes our work.

## 2.RESEARCH CARRIED OUT PREVIOUSLY:

Reuse is a process of creating a solution to a problem based on existing solutions of its sub problems. The reuse activity can be divided in to following major steps performed at different phases in preparation for the next phase.[29].

1. Reuse Strategy Development after analyzing the problem and available solutions of its sub problems.
2. Make out a solution structure for the problem following the reuse plan.
3. Reconfigure the solution structure to the possibility of using predefined components available at the next phase.
4. Obtaining, instantiating and altering predefined components.
   Integrating the components into products for this phase and evaluating the products.

The specific cost effectiveness and productivity goals are met with Reuse libraries ,which are organized of tools, personnel procedures, components of software that facilitates software reuse .Typical libraries has the following capabilities:

- Browsing , probing and retrieval can be carried on using GUI of a Automated Library System.
- A framework for standard components.
- Each Domain would be effectively classified.
- Detailed documentation of system and component.

It is also desirable to have a support to transform, adapt or specialize the components. Also , for the system to be used, it should be readily available for the developers and must support access from a variety of platforms.

In order to avoid frustration and delay for the user , the library should implement a way to categorize reusable components, irrespective of the tool being used.

Standard component frameworks help ease the process of comprehension and comparison of similar components, and include data such as relative numeric measures for reusability, reliability, maintainability and portability [5].

Besides the above, The system development and acquisition process must integrate library capabilities and procedures and Specific requirements of security and integrity should be identified and supported and also among diverse library systems the intercommunication and interoperability must exists.

The library must gather and analyze the usefulness of its reusable components, their accuracy, and library's general responsiveness as per needs of users, In order to measure reuse access.

### 3. *REUSABLE SOFTWARE COMPONENT REPOSITORIES[33]:*

Lets take a look at a few commercial repositories..

**Reuse Library Toolset (RLT)::** In 1994 , the commercial release of RLT was announced by EVB Software Engineering Inc.. Independent of development process, design method and programming languages used reusable assets can be managed and created using this system. To represent all life-cycle assets RLT employs the Extended Faceted Classification System, controlled keyword, attribute value (frames), and asset interdependencies. It provides ability to exchange library information across multiple platforms and databases, and also provides library metrics, client-server architecture. The multiple platforms supported includes DEC Alpha, Sybase,HP/UX,Oracle,Informix,OSF1 and in 1995 additional platforms like Windows 3.1/NT and OS/2 were also supported.

**The Universal Repository::** This system is based on the object oriented principles and was developed by UNISYS. At the core of this repository is the Repository Services Model (RSM) - which can encompass representations of all tools, database management systems (DBMSs), programming languages, business rules, and data. Based on the structures provided in the RSM, customers can add their own models to extend the repository. Customers can develop, validate, and verify a component for use in one product. Customers can quickly adapt with this new technology with the support and training available in this system. The repository promotes reuse by providing a shared catalogue of all software components. A single change to correct a defect in a reused component is reflected in all tools using that component. Such consistency among products ensures their integration and interoperability when you port them different operating systems.

**+1 Reuse Repository:** Presently running on SUN Workstation platforms, it was developed by +1 Software Engineering Co. in California. Its GUI is based on Open Windows, Motif and CDE ,and Solaris operating system. User-Defined Reuse library, Filtered reuse library and selective Reuse are three forms of reuse supported by this system. Project wise "Filtered components" are maintained by it. Selective reuse significantly improves a user's ability to reuse all source code and documentation from all previous projects and at any granularity. This system supports at all levels of software development life cycle including Design, Code, Test cases , Shell scripts of tests , modeling information and even expected results.

**AIRS::** E.J. Ostertag, J.A. Hendler,R. Prieto-Diaz, C. Braun [7] developed an AI-based library system for software reuse- called as AIRS. In this system a component is described by a set of (feature, term) pairs. A feature represents a classification criterion, and is defined by a set of related terms . It also allows representation of packages with features. Degree of similarity between their descriptions and a given target description decides the candidate component from library for reuse in this system. A non-negative magnitude called distance represents the expected effort required to obtain the target given a candidate is used as a quantifier for similarity. Subsumption, closeness, and package comparators are three functions used to compute Distances in this system.

**Software Asset Library Management System (SALMS)::** SALMS is a system for classifying, describing, and querying reusable assets [6]. The lack of visibility of reusable assets is the common inhibiting issue among the developer community. Such a problem can be solved using a central repository. The gap between development for and with reuse respectively is filled by the SALMS. It is easily accessible by all developers as it can be distributed over network or unix workstations. It has a webtechnologies based GUI.

**Automated Software Reuse Repository (ASRR)::** It is a searchable repository of reuse information. The administration tool and Reuse repository are the main components of this. The administration portion of the tool performs user administrative functionality including: the ability to add, delete, or change users and their attributes. The attributes include the following: security levels, group and security permissions to add, edit and delete modules. The reuse repository allows the user to upload modules and store them in a searchable repository. The following functions are provided by ASRR:: Easy access to reuse items, reuse information readily available for users, Program Control, Protection & Security.

**HSTX Reuse Repository:-** The HSTX reuse repository was developed by Hughes STX Corporation. The mechanisms are designed so that users can search/browse the contents of the Reuse Repository for what they need and submit contributions to the reuse repository librarian through WWW pages.

Apart from the above certain Government Repositories include:Defense Software Repository System (DSRS), Library Interoperability Demonstration (LID), Integrated - Computer Aided Software Engineering (I-CASE). Multimedia Oriented Repository Environment (MORE). Asset Source for Software Engineering Technology(SAIC/ASSET).The Public Ada Library (PAL).The Ada Library and the Reuse Library at the Defense Information Systems Agency (DISA),CAPS Software Reusable Component Repository.

### 4.WORK RELATED TO COMPONENT IDENTIFICATION AND EXPERIENCES OF REUSE:

Providing a best quality product with in the least time is the biggest challenge that organizations are facing today. The cost increased very high, when, to deliver high quality product in less time, if more developers and experts are

hired. The making of decision in component selection, to start from a particular point to get the best result is the most complex thing.

Customers need to visit all repositories in order to select the best qualifying component from the different repositories available for reusable components .[35] SRSCS's main purpose is to provide a single point of the access to the customers from where they can select their required component instead of visiting all the repositories one by one. Functional requirements of the component as described by the owner organization are taken as basis for selection of reusable component in SRSCS. SRSCs process consists of extraction, transformation, Loading and component selection steps.

TeraData is chosen for implementation in [35] because of the reason that heterogeneous sources like flat files, RDBMS are used to extract information and then ,after transformation ,the data is loaded in to single repository. The work is based on ETL i.e. Extraction, Transformation, Loading.

William W. Agresti [31] has conducted a survey of 128 developers to explore their experiences and perceptions about reuse of code and to emphasis on reuse as a source of cost savings in software development. They stress that availability, awareness, accessibility and acceptability are the four conditions for any organization to obtain benefits of reusing code.

The motivation for [31] is that if we improve our understanding of developers' experiences and perceptions regarding reuse, we may identify ways to increase reuse, and, by doing so, further reduce the cost of software development. "It is imperative to understand the behavior of software developers. The key to successful reuse programs obviously depends on its acceptance by the users."[1]. Where as [6], analyzes data from a development environment that classified modules by the reuse categories of: Verbatim, Adapted, Rebuilt and Newly Developed. [31] followed an approach of "4A" model based on sequence of conditions that must meet for reuse to occur. Availability, Awareness, Accessibility, Acceptability. Reuse cannot occur if any of the above 4 conditions are not met. In the other way they act as impediments for the occurrence of software reuse.

Standish and Thomas [8] presented a paper on, "An Essay on Software Reuse". This paper explored basic software reuse concept and discussed briefly what economic incentives were used for software system generation so that the software reusability will be improved.

Arnold [18] [17] mentioned a number of heuristics that can be used for locating reusable components in the Ada source code.

The work of Mikael et al [40]is based on the component metadata automatically retrieved from development tools. Idea of improving test phase using meta data is extended to cover the whole component development phase. Resusing of tests as specifications and results in metadata instead of executable test cases is done. But the work is not applicable where inheritance relations of Object Oriented approaches may affect the storage structure.

## 5.RETRIEVING REUSABLE COMPONENT:

To enhance retrieval an automation retrieval of software components was proposed by Luqi and Guo [9] through their paper "Toward Automated Retrieval for a Software Component Repository". The paper discussed the improvement of over existing software component system using signature matching and provide profile based match for effective development to the system is obtained.

In [36] a component retrieval system for reuse process is proposed with integration of facet attributes for fetching process. Metadata repository integrates expert knowledge of correlative domains and generalizes crucial concepts and relations among concept in this domains.

The precision of software component retrieval is poor as a result of subjective factor in faceted classification retrieval in case of software component retrieval based on faceted classification.

A Survey For Effective Search And Retrieval Of Components From Software -Repositories [30 ] paper presents a survey about the main research on effective search and retrieval of components and on various software repositories.

Various retrieval techniques are enumerated classification, facets, frame based classification; free-text indexing and relational databases have been employed to address the problem of finding relevant components[21]. But issues involving how effective repositories are built and populated have received considerable less attention in the recent past.

This work in [21] aims at summarizing the state of the art in software reuse repositories research by proposing answers to the following questions: requirements of software reuse repository, design aspects of repository, approaches for constructing effective repository , challenges faced while searching components and challenges faced while retrieving the components. The work throws light on Codefinder-PEEL repository that was outlined by Scott henninger in 1996. And also Code Broker which is an active and adaptive reuse repository system developed by Yunwen Ye in 2001 to assist java developers. It also speaks about John Grundy's aspects based indexing work in 2001 named as ASPECT BASED COMPONENT REPOSITORY, which is primarily a facet based approach. The work also mentions about Jiyun Lee, et al worked out in 2003 to build a component repository for facilitating EJB. The repository was CRECOR( Component Repository for Facilitating EJB Component Reuse) Component Repository.Portability, Flexibility , Understandability and Confidence are the four listed reusability factors by the ESPRIR-2 project called REBOOT( Reuse Based on Object-Oriented Techniques).

Paper by Subedha[38] reported on going work of them which adopted quality hierarchy method for determining reusability factor to make quality assessment more effective. Time , reuse frequency, usefulness quality factors based context specific level model of reusability was proposed. It proposed a dynamic approach to analyze the component for reusability.

The modeling for reusability has been addressed by only a few researchers in the past The contribution of metrics to

collectively determine reusability is still a big research area.

Function based software systems' reusability prediction using Expectation Maximization based clustering was proposed by Himani Goel and Gurbhej Sing[8]. In this coupling, Cyclomatic complexity , volume, regularity, reuse frequency are the metrics used for measuring reusability of component.

Parvinder & Shalini [10] proposed particle swarm optimization technique along with the four variations of conjugate gradient algorithm to train the feed forward network. The performance of the trained neural network is tested to evaluate the reusability level of the procedure based software system.

P.Shirisha et al.. worked on code level optimization with creating a function module in ABAP to check for optimization of any code written in ABAP.

Where as [11][12][18][19][32][34][37][39] work in the area of applying neural networks, artificial intelligence Genetic algorithms etc in reusability identification and component identification from repository.[27] implements their work in the form of a suffix tree based component representation in a tree model. The pruning over the keyword list is performed at each level and the suffix tree consists of various levels.

To get the components or code from the library, it is required to know architecture of the software thoroughly. But to build such architectural overview the first requirement is to retrieve the component information from the software product. This information is collectively called the software ontology.

Lucredio et al. [10] presented a paper, "Component Retrieval Using Metrix Indexing". If the software repositories have a large number of stored components, then efficient retrieval of these software components from the development modeling to the system. To make the retrieval process more efficient, the scenario specific search mechanisms are defined.

Qualitative and Empirical are basically two approaches to evaluate software. Module designed factors and module implementation factors are two categories of factors identified by selby in his recent experiments that characterize successful reuse based software development of large systems.

The module design factors that characterize module reuse without revision were: low coupling, high cohesion, few input-output parameters, few reads and writes and many comments. Where as Source lines, low cyclomatric complexity are the module implementation factors. It was observed that modules used without revision had the fewest faults and subsequently lowest fault correction effort.

Chen and Lee[16] found that lower the value of the software complexity metrics, the higher the programmer productivity from the controlled experiment they conducted on 130 reusable C++ components developed by them, in order to relate level of reuse in a program to software productivity and quality. The software metrics collected included Halstead size, volume of program, program level, effort and estimated difficulty.

Vijayan et. al. have given some different touch to component selection and they used domain model and object libraries to identify software components [20]. But unfortunately this method doesnot provide any functional information for decision making about component selection. It implements keyword, based approach, retrieved from the user's query written in Natural Language.

The above approaches does not provide the solution that how to select best qualifying component among the available different repositories and how to compare components with each other if there are more than one component developed for the same purpose by different organizations.

The effort needed to modify a component as reflected by the number or percent of operations to add or modify was suggested as a reusability metric by Woodfield,Embley and scott after conducting an experiment to asses the reusability of an ADT in 21 different languages by 51 developers.

Program size, structure of program, documentation and language used to code and the reuse experience are the five metrics identified by Prieto-Diaz and Freeman for evaluating reusability and the process is an Empirical method.

Caldiera and Baili state that the basic reusability attributes depends on qualities of correctness, readability, testability, ease of modification, and performance as we cannot calculate these directly, they propose 4 candidate measures of reusability such as: Halstead's program volume, Mc Cabe's Cyclomatic complexity, Regularity and Reuse frequency.

Function, form and similarity are the three approaches discussed by Hislop[26]. The prototype tool developed "Softkin" consisted of a data collector and a data analyzer. The collector parses the software and calculates measures of form for each module. The analyser computes the similarity measures based on a variety of form metrics such as McCabe Complexity and structure profile metrics.

As most of the methods focus specially on internal characteristics of components and environmental factors are ignored ,Poulin[29].. challenged reusability researchers for domain attributes to be incorporated in to software metrics. A critically important factor to find reusability of a component is its application domain, this point is focused in their work. There was a requirement of finding domain knowledge and required artifacts in an economical way. The success of component is the possible best measure for a component's reuse..

The study done by Lin and Clancy showed that a standard layout can be used by a potential user to quickly scan the important aspects of a component such as implementation information, text description, illustrations and pseudo code

In order to know where to make changes to improve reuse the following things can be helpful: Detailed comments, better prologs, and better online and off line tools and access to persona with required knowledge.

## 6. CONCLUSION:

The paper is a part of ongoing research by us in Identification of Reusable component using neural networks and in this we summarized the previous work that was done in the area of Reuse, what is reuse, how people worked on to identify reusable components etc. It is evident that still there is a lot of scope for improvements in the mechanisms being implemented for reusable components creation, identification and maintainance. This work does not emphasis on application of neural networks, which we would like to present as a separate paper.

## REFERENCES:


1) Wahlster W., "Einführung in die Methoden der Künstli-chen Intelligenz," University of Saarbrücken, Germany, Lecture Notes 2002.
2) Wachsmuth I.,"The Concept of Intelligence in AI," in Prerational Intelligence-Adaptive Behavior and Intel-ligent Systems without Symbols and Logic , vol. 1, The Nether-lands: Kluwer Academic Publishers, 2000, pp. 43-55.
3) Winston P. H., Artificial intelligence , 3rd (repr. with corrections 1993) ed. Reading, Mass.: Addison-Wesley, ISBN: 0-201-53377-4, 1993.
4) Pozewaunig H., Mining Component Behavior to Sup-port Software Retrieval. PhD Thesis. Institut für Infor-matik-Systeme der Fakultät für Wirtschaftswissen-schaften und Informatik, Universität Klagenfurt, Kla-genfurt, 2001.
5) J. Penix and P. Alexander, ``Design representation for automating software component reuse," in Proceedings of the first international workshop on Knowledge-Based systems for the (re)Use of Program libraries, Nov. 1995.
6) Elisabetta Morandin, "SALMS v5.1: A System for Classifying, Describing, and Querying about Reusable Software Assets", The Proceedings of ICSR '98.
7) G. Arango and R. Prieto-Diaz, "Domain Analysis Concepts and Research Directions", Domain Analysis and Software System Modeling,sIEEE Computer Society, 1991.
8) Standish,T., and Thomas,A., "An Essay on Software Reuse", IEEE Transactionsonsoftware engineering,Vo l. SE-10, No. 5, pp. 494-497, 1984.Luqi.,
9) Retrieval for a Soft ware Component Repository", Proceedings of IEEE International Conference and Workshop on ECBS, pp. 99-105, 1999 .
10) Lucredio,D., Gavio li,A., Prado,A.F., and Biajiz,M., "Component Retrieval Using Metrix Indexing" ,IRI, Proceedings of IEEE International Conference ,pp. 79-84, 2004.
11) C. Veras,R., and Silvio,L., " Comparative Study of Clustering Techniques for the Organization of Software Repositories",,Vo l. 1,pp. 210 -214, 2007.
12) Dixit,A., and Saxena,P.C., " Software Co mponent Retrieva l Using Genetic Algorithms" International Conference on Co mputer and Automation Engineering © IEEE, ISBN: 978-0-7695-3569-2, pp. 151-155, 2009.
13) Ichii,M., Hayase,Y., Yoko mo ri,R., Yamamoto,T., and Inoue, K., "Software Component Recommendation Using Collaborative Filtering", SUITE, ISBN: 978-1-4244-3740-5, pp.17-20, 2009
14) Viana, T.B., Nobrega, H.I., Ribe iro, T., and Silveira, G.,"A Search Service for Software Components Based on a Semi-Structured Data Representation Model", 6th International Conference on Information Technology: New Generations © IEEE, ISBN: 978-1-4244-3770-2, pp. 1479 -1484, 2009.
15) Aboud,N.A., Arevalo,G., Fa lleri,J-R.,Huchard,M., Tibermacine,C., Urtado,C., and Vauttier,S., "Automated Architectural Co mponent Classificat ion using Concept Lattices", Software Architecture & European Conference on Software Architecture WICSA/ ECSA @2009 IEEE, ISBN: 978-1-4244-4984-2, pp. 21-30, 2009.
16) Chen, Y.F.Nishimoto, M.Y.andRamamoorty, C.V. "The CInformationAbstractionSystem",IEEE Trans. on Software Engineering, 16, No. 3, March 1990.
17) "Software Reusability and Efficiency: A Scientific And Technological Study", Undertaken by Parallab, Bergen Center for Computational Science, University of Bergen (Norway) for the Enacts Network, Sectoral Report,FinalVersion-April2004, http://www.enacts.org.
18) Maxym Sjachyn, Ljerka Beus-Dukic, "Semantic Component Selection – SemaCS", 2006
19) Vijayan Sugumaran, Veda C. Storey, "A Semantic-Based Approach to Component Retrieval", 2003.
20) Vijayan Sugumaran, Mohan Tanniru and Veda C.Storey, "Identifying software components from process requirements using domain model and object libraries", 1999 .
21) Scott Henninger, "Supporting the Construction and Evolution of Component Repsitories", IEEE, 1996, pp-280-286.
22) Pennell, James P., "An Assessment of Software Portability and Reusability for the WAM Program," Institute for Defense Analysis, Alexandria, VA, October 1990 .
23) Halstead, Maurice H. Elements of Software Science. Elsevier North-Holland, New York,1977.
24) Zhuo, Fang, Bruce Lowther, Paul Oman, and Jack Hagemeister, "Constructing and Testing Software Maintainability Assessment Models,"Proceedings of the IEEE Computer Society International Soft-ware Metrics Symposium,Baltimore, MD, 21-22 May 1993, pp. 61-70.
25) Mayobre, Guillermo, "Using Code Reusability Analysis to Identify Reusable Components from the SoftwareRelatedto anApplication Domain,"(WISR'91), Reston, VA, 18-22 November 1991.
26) Hislop, Gregory W., "Using Existing Software in a Software Reuse Initiative," (WISR'93), 2-4 November 1993, Owego, New York.
27) Kanwaljeet Sandhu, Trilok Gaba,-A Novel Technique for Components Retrieval from Repositories -- COMPUSOFT, An international journal of advanced computer technology, 3 (6), June-2014 (Volume-III, Issue-VI).
28) Ajay Kumar, -MEASURING SOFTWARE REUSABILITY USING SVM BASED CLASSIFIER APPROACH --- International Journal of Information Technology and Knowledge Management January-June 2012, Volume 5, No. 1, pp. 205-209.
29) Jeffrey S. Poulin, Loral Federal Systems–Owego -Measuring Software Reusability-. Proceedings of theThird International Conference on Software Reuse, Rio de Janeiro, Brazil, 1-4 November 1994.
30) Amandeep Bakshi, Seema Bawa,--ASurvey For Effective Search And Retrieval Of Components From Software -Repositories,IJERTVol.2 Issue4,April– 2013.
31) William W. Agresti- Software Reuse: Developers' Experiences and Perceptions-. Journal of Software EngineeringandApplications,2011,1,4858.Published OnlineJanuary2010 (http://www.scirp.org/journal/jsea).
32) Artificial Intelligence and Software Engineering: Status and Future Trends --Jörg Rech, Klaus-Dieter Althoff
33) Jiang Guo, Luqi --A Survey of Software Reuse Repositories--.
34) G. Boetticher, K. Srinivas, D. Eichmann --A Neural Net-Based Approach to Software Metrics -.
35) SINGLE REPOSITORY FOR SOFTWARE COMPONENT SELECTION (SRSCS): A REUSABLE SOFTWARE COMPONENT SELECTION TECHNIQUE --YOUNAS WAHAB, MUHAMMAD IMRAN BABAR,SHAHBAZ AHMED
36) Optimal Component Software Development based on Meta Data Repositories -Tamanna Sood, Dr.Rajiv Mahajan - IJ ARCSand software Engineering. Volume 4, Issue 2, February 2014. ISSN: 2277 128X.
37) Identification of Object Oriented Reusable Components Using Multilayer Perceptron Based Approach --Shamsher Singh, Pushpinder Singh, and Neeraj Mohan .-ICCEMT'2012 Sept8-9, 2012 Bangkok
38) PROCESS MODEL FOR REUSABILITY IN CONTEXT-SPECIFIC REUSABLE SOFTWARE COMPONENTS V. Subedha, Dr. S. Sridhar. IJCSE, Vol. 3 No. 1 Feb-Mar 2012. ISSN : 0976-5166
39) Integrating Matlab Neural Networks Toolbox functionality in a fully reusable software component library—Arturo, Emilio,Lado,Jacinto.Manuael—Springer,Neural Computational and Applications,2007.
40) A model for reuse and optimization of Embedded SoftwareComponents-Mikeal,Joakim,Kristian-proceedings of the ITI 2007 29th International Conference on Information Technology Interfaces, June 25-28, 2007.