Recommendations for the Papers

Aspect-oriented Requirements Engineering for component-based software systems
J. Grundy

When you start developing reusable components then you quickly will notice that specifying such components is different from specifying software for a given purpose for some dedicated customer. Stakeholders are no longer clearly identifiable. The context where a component will be used is still open at the time of specification. On the other hand, a component typically has a context of other components it relies on. So, how can you adequately specify requirements under these conditions? If you are looking for answers, you should read this paper.
-Martin Glinz

Bridging the Gap between Past and Future RE: A Scenario Based Approach
P. Haumer, P. Heymans, M. Jarke, K. Pohl

We all know that it is important to understand the existing system before designing a new one. But all too often that understanding is not used effectively when the requirements and design of a new system begin to take shape. In this paper, Peter Haumer and colleagues demonstrate that a set of scenarios can be used to help validate requirements by exploiting stakeholders' familiarity with the old system. Along the way, they show how to integrate informal chunks of information (video clips of 'real world scenarios') with a formal requirements modeling language (Albert-II) to improve traceability and comprehension of the formal model. Not only is there a tool, but there's plenty of method too. The authors draw on their industrial experience to provide a number of 'method chunks' that illustrate how this approach addresses real problems during requirements modelling. If you've ever wondered what use video clips are when it comes to modelling, this is the paper for you.

— Steve Easterbrook

An Empirical Investigation of Multiple Viewpoint Reasoning in Requirements Engineering
T. Menzies, S.Easterbrook, B. Nuseibeh, S. Waugh

This paper describes experiments that test two widely-held beliefs about using multiple worlds to reason about inconsistent requirements: multiple-world reasoning is (1) impractical because of the proliferation of worlds, but (2) necessary because no single model comes close to describing all of the desired behaviours. The results indicate that neither hypothesis is generally true.

- J o Atlee

Experience With Goal-Scenario Coupling In Requirements Engineering
C. Rolland, K. Grosz, R. Kla

The merits and benefits of both scenario-based and goal-based analysis in requirements engineering are well documented. In fact, it is impossible to overlook the recent flurry of activity and research in both of these approaches. However, many of us are still struggling with how to use scenarios and goals in a complementary fashion. This paper advances the current state of research by taking an integrative approach to goal and scenario-oriented requirements analysis. Using concrete examples from a BPR project, the authors provide a balanced assessment of the strengths and weaknesses of their goal-scenario coupling approach.

-Annie Anton

Requirements Engineering, Expectations Management, and The Two Cultures
B. Boehm, M. Abi-Antoun, J. Kwan, A. Lynch, D. Port

Would you like to know what is going to cause your development project to fail? How about knowing what will help it succeed? The technique is as simple as it is effective. First, make a list of all those things (architectures, restrictive requirements, etc.) that complicated your prior projects. Second, make a list of all those things that simplified your prior projects. Now, predicate the applicability of these lists by application type and you have complicators and simplifiers. Application of these lists during requirements analysis of new projects will significantly reduce your project failure.

-William N. Robinson

Events in Linear-Time Properties
M. Chechik, D. Paun

Automated verification offers substantial benefits to developers of systems with critical properties (like safety). Experience shows that it is ultimately cheaper and more effective to build in correctness than attempt to test it in. Part of building in correctness is showing that the requirements are correct with respect to critical system properties like safety constraints. While model checkers offer the potential to automate the proof of such properties, their practical application has been hampered by the inability to deal with large systems due to the size of the state space. This paper provides a nice treatment of the subject in addressing such a problem with the use of Linear Temporal Logic. The authors provide techniques that support the use of more expressive constructs (the "next" operator) while retaining the ability to simplify the state space to manageable size (through showing that LTL formulae are "closed under stuttering"). For those interested in issues in model checking Linear Temporal Logic or understanding the surrounding issues, this paper gives a clear and interesting presentation.

— Stuart Faulk

Formal Modeling of Space Shuttle Software Change Requests using SCR
V. Wiels, S. Easterbrook

This paper describes the authors' experiences using SCR to model NASA-style requirements and change requests and to automate some of the tedious manual consistency checking when analyzing change requests. The appeal of this experience report is that the case study is an industrial-strength specification and the authors have no stake in the formal method.

- Jo Atlee

Generating User Interface Prototypes from Scenarios
M. Elkoutbi, I. Khriss, R.K. Keller

The mention of user interface prototypes often conjures images of attractive dialog screens that are intended to demonstrate the look and feel of a new or altered application. Creation of such prototypes is often a matter of composing from a rich palette of graphics and behaviors, and using the result as a communication vehicle between customer and designer. Beyond this, though, lies the need to explore the requirements that the system's various interactors impose on the user interface, and the need to determine what a user interface might look like if it were to incorporate the needs of the multiple interactors. This paper presents a no-nonsense, constructive method of creating annotated UML diagrams and generating such user interface prototypes from them. The method uses collaboration scenarios as the basis of UI prototype generation, and the resultant UI prototypes reflect the requirements indicated by the combined scenarios of system use.

-Mark Feblowitz

Guidance for Parallel Requirements Acquisition and COTS Software Selection
C. Ncube and N.A.M Maiden

As COTS (Commercial off-the-shelf) technology becomes more popular and attractive, more practitioners and researchers have the same question "How can we find the appropriate COTS software?". One of the answers is found here! If you take into account which COTS is appropriate for your development during eliciting and specifying customers' requirements, the requirements specification may make your selection of COTS more effective and easier. The authors provide a guide informing you how to acquire the requirements that can imply the selection of COTS software. In the authors' approach, called PORE, the guide is defined as process descriptions consisting of three parts. The first is the process model as goals for selecting COTS Software and as the sequence of activities to achieve the goals. The second is the methods and techniques that assist in your achieving the goals. The last one is the product model for COTS Software. The authors assessed their process description by applying it to the requirements acquisition and the selection of an electronic mail system from three commercial ones. If you have great interest in COTS and are puzzling the above question, then you will want to explore this work.

— Motoshi Saeki

Human Error and System Requirements
A. Sutcliffe, J. Galliers, S. Minocha

This papers focuses on issues that have not been widely recognized in the RE community despite their importance. The paper discusses how errors can be made with using the system and how different designs of system functionality, user interface design and task structure may impose sources of error during system use. The proposed method suggests a way to identify and analyze factors that can induce errors during the system use and a quantified model that helps "walk through" various parts of the system and its use and how error prone they can be. The model is useful reading to anybody interested in designing robust, user-friendly systems based on a systematic analysis of system requirements from the view point of system usability.

— Kalle Lyytinen

An Integrated Scenario Management Strategy
T. Alspaugh, A. Anton, A. Alexander, T. Barnes, B. Mott

So, now that we are convinced that the concept of a scenario is useful during software requirements elicitation, what comes next? This paper argues that we need better scenario management mechanisms that would allow us to correlate and organize a database of scenaria. Towards this end, the authors propose a mechanism for discovering common scenaria subsequences ("episodes") and another mechanism for measuring similarity between scenaria. This is an early example of the next phase of research on this important topic. Hopefully this research will build a technology and a methodology that brings scenaria into mainstream practice.

-John Mylopoulos

Integrating Non-Functional Requirements into Data Modeling
L.M. Cysneiros, J.C.S.P. Leite

Are you really serious about the quality of the system? Do you ever carry out any non-functional analysis then? Or do you still analyze only the functional aspect of the system despite your claim on customer satisfaction? Are you interested in gaining mileage on cost reduction but stuck with functional concepts (and icons) only? This paper describes a serious effort towards an honest non-functional analysis. It uses "the NFR Framework" for the non-functional analysis of a real system and lends a significant insight as to the effectiveness of such an analysis. You get not only improved quality but substantial reduction in software development cost! This paper is a must for any software practitioner who is serious about software quality, production time and cost.

-Lawrence Chung

Introducing Measurable Quality Requirements: A Case Study
S. Jacobs

This paper is chock full of juicy nuggets of experimental wisdom. It describes a practical attempt to improve requirements engineering in an industrial project by the application of Gilb's method. It focuses on the all important communication between designers and members of the client organization and on the behavior of these people. The paper describes many specific problems and the lessons learned from them, both good and bad. These lessons are directly applicable to other projects in other companies.

— Daniel Berry

Prioritisation of System Changes using Cost-Benefit and Risk Assessments
D. Greer, D.W. Bustard, T. Sunazuka, M. Yoshimura

New requirements introduce additional risks by introducing changes. This paper describes an approach of applying risk management techniques to analyse these changes in an evolutionary development context. The risk analysis allows to quantify the impact of changes from a business perspective and allows for effective prioritisation of proposed changes. The prioritisation is based an analysis of the current system, the target system and the development process. It concerns benefits introduced by the changes, cost associated with the changes and risk exposure in the development process. The relative importance of these factors can be adjusted. Tool support exists for applying the analysis and prioritisation steps of the approach. Elements of an industrial case study are described.

— Hans-Jtirgen Kugler

ScenIC: A Strategy for Inquiry-Driven Requirements Determination
c. Potts

Nowadays there are numerous case studies and reviews which demonstrate that scenarios are effective for eliciting and validating system requirements. Both researchers and practitioners now need to progress forward and determine how best to use scenarios for different requirements tasks. The ScenIC method reported in the paper is a substantial contribution to this progress. ScenIC offers prescriptive guidance for acquiring and validating system requirements with scenarios which describe future system use. In particular, it addresses what might be called the 'scenario-requirements gap' which represents the failure of scenario-based approaches to address adequately the emergence of requirements from scenarios. To bridge this gap, guidelines are presented in the form of: (i) issues which arise from scenarios, and: (ii) means of resolving these issues, in the form of requirements themselves or methods and techniques for discovering or refining requirements. As such, it highlights the importance of requirement and scenario content, rather than representation of requirements and scenarios. I hope that readers will find this paper as insightful and useful as I did.

-Neil Maiden

Social analysis in the requirements engineering process: from ethnography to method
S. Viller, I. Sommerville

If you want to find out how to really make the most social analysis and learn from Computer Scientists who have developed a tried and tested method then this is the paper for you. Ian Sommerville and his colleagues have been working on social level RE for several years- see his books with Kotyna (RE processes and techniques) and Sawyer (RE: a good practice guide). This paper summarises their experience and gives ethnographically based method that shows how design recommendations can be derived from these studies. The paper describes the Coherence method that builds on their earlier VORD (Viewpoint-oriented requirements definition) method and extends it to social viewpoints. The analysis links through to specification in UML (Unified Modelling Language) and is illustrated with an Air traffic control case study. This paper gives you the most authoritative review of ethnographic approaches to RE.

— Alistair Sutcliffe

Use Case Authoring Guidance: Results of an Empirical Study
C. Ben Achour, C. Rolland, N.A.M. Maiden, C. Souveyet

It is widely accepted that use cases, a popular term for scenarios, are valuable for requirements acquisition and validation. The CREWS research project on scenarios has proposed some "style guidelines" based on current practices and "content guidelines" based on linguistic constructs (case grammars). The question addressed by this paper is: "Do these guidelines help?" The paper proposes several hypotheses that predict that the use of the guidelines will lead to better (i.e., more complete, consistent, unambiguous, etc.) descriptions. Well, do they? Read the paper to find out. You will find a very lucid and honest evaluation of the results. For a given hypothesis, something is learned whether the hypothesis is confirmed or not.

— Sol Greenspan

Use Case Maps for the Capture and Validation of Distributed Systems Requirements
D. Amyot, L. Logrippo, R.J.A. Buhr, T. Gray

Use case maps (UCMs) are a simple, semiformal technique to represent requirements on systems in the form of scenarios. A UCM represents a scenario as a causal sequence of tasks. UCMs also allow the representation of parts of scenarios to different components in a distributed system. They are more abstract than message sequence charts, because they abstract from the particular sequence in which messages are exchanged by system components. This paper shows how UCMs can be used to represent distributed system features in the form of scenarios, and how the integration of different UCMs allows the designer to avoid many unwanted interactions between these features. The paper goes on to show how the resulting UCM model can be transformed into a formal Lotos specification, which can then be tested using a Lotos toolkit. These tests may yield further interactions, that could not have been discovered using the semiformal UCM technique. However, by making the UCMs first, many unwanted feature interactions were avoided in the first place. The paper is a good example of the advantage of combining formal and semiformal techniques in software design.

—Roel Wieringa

Viewpoint-oriented Software Development by Distributed Graph Transformation: Towards Living with Inconsistencies in Integrating Multiple Perspectives
M. Goedicke, T. Meyer, G. Taentzer

Dealing with multiple viewpoints is an important area in RE. Many models and representations are needed during systems development and evolution. Having good frameworks and tools to support multiple viewpoints and perspectives, to relate them and to manage correspondences and inconsistencies is therefore highly desirable. Viewpoint issues have been addresses by a number of approaches. A particularly significant one was the Viewpoints (TM) approach. This paper offers a fresh look at viewpoint issues by introducing a distributed graph transformation approach. The proposal provides for a uniform way of expressing operations on Viewpoints (TM). An example illustrates the linking of a software architecture design model to a performance evaluation model.

-Eric Yu