Online Exclusive from PHARMACEUTICAL ENGINEERING®
The Official Magazine of ISPE
January/February 2012, Vol. 32 No. 1
www.PharmaceuticalEngineering.org

This article presents the case that fault tree analysis is the better risk analysis method to apply early in software development projects.

Applying Fault Tree Analysis (FTA) as a Top Level Risk Management Tool in Software Development

by Paul Noble, PhD

Introduction

With the introduction of GAMP® 5, "A Risk-Based Approach to Compliant GxP Computerized Systems,"[1] in 2008, risk assessment is to be included in all life cycle phases of a computerized system. Conceived was "an iterative process used throughout the entire computerized system life cycle." Typically, this has been interpreted by the application of an initial risk identification followed by use of the popular FMEA method for determining the testing scope of software features.

Recently, it has been recognized in the Quality Risk Management (QRM) approach[2] that selection and exclusive use of a single risk management tool, such as FMEA, may limit the usefulness of QRM. The same limitation also can be expected when using risk management in a software development project. When the selection process in the referenced article is followed for risk assessment of undesirable events arising from software use, particularly during the early phases of a development project, the Fault Tree Analysis (FTA) method is suggested as one of the methods of choice.

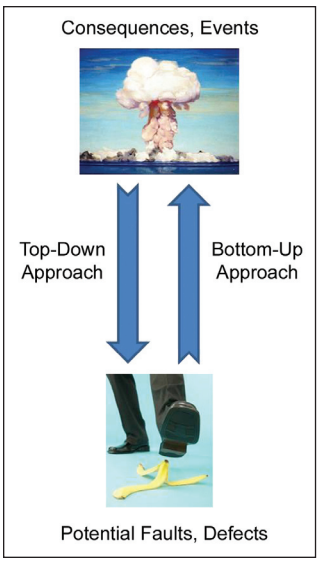

Figure 1. Risk assessment approaches.

FTA is a top-down type of analysis to be explained later. This article presents the case that it has advantages for the conceptual and design lifecycle phases. The distinction between top-down and bottom-up assessment methods has been largely ignored by the regulators, which leaves it to the project team to recognize the benefits of these two fundamentally different approaches.

The automation of business processes is targeted as an area for application of top-down methods, because potential human errors are a great source of risks for these processes, and such errors need to be addressed early in the life cycle of the system.

Unacceptable operation of computerized systems can arise both from human use and hard-/software defects. Human error has commonly not received the same attention in the past, while computers were replacing manual, error-prone operations. Where full automation is not practical, restriction of authorized use has been commonly relied upon to address the potential of human error, but this tactic is limited for automated business processes, such that they still commonly have a high potential for human error.

Although it is generally recognized that software defects and human errors are difficult to predict, and quantification of risks arising from them cannot be based upon failure rates, it may not be commonly recognized that they have different statistical dependencies. In tandem with the examination of a top-down approach to risk assessment, attention will be brought to the reader of the statistical nature of user errors, borrowing upon the statistical concepts of the QbD approach.[3]

Design Spaces Applicable to a Business Process

It is obvious that a business process cannot be controlled like a physical process can, even when the process is computerized. Active participants include both the users and the business players who participate in the process. Whereas a physical process can be controlled, such that it becomes robust and reproducible, a business process is not necessarily reproducible and is biased by the behavior of the active participants. Also, a physical process is governed by physical laws, whereas a computerized business process is partly governed by program logic, which can be in error.

In the QbD concept for physical (pharmaceutical) processes,[2] a design space is defined as, "The multidimensional combination and interaction of input variables (e.g., material attributes) and process parameters that have been demonstrated to provide assurance of quality." In physical terms, it is defined by the control parameters and their limits, which are needed to keep the process within a pre-defined quality level for the dependent variables, i.e., a desired event space. Of course, real systems have variability which cannot be completely removed and statistical methods are at the forefront of QbD. Typically, a physical process has several degrees of freedom, leading to multiple control variables, and the potential event space is quite large and usually considered mathematically to be infinite.

Conceptually, there is clearly a need for a business process to stay within a design space. Although a design space for program logic may not be a useful concept to employ, a design space which limits the human inputs to the system is. Inputs from other systems or devices can, as a useful simplification, be ignored because they are more reliable (assuming that the computerized system will be correctly specified and tested). It should become evident in this article that the separate consideration of user inputs has advantages in the design and review of the system.

The goal during the design of a computerized business process should be to limit the user inputs to the extent necessary for achieving the quality objectives. Users generally are not keen on limiting their freedom in use of the system, but experienced developers know that this must be done in order to create a robust product. Typical programming methods include selection lists, required fields, and the cancel button.

The problem of defining a design space for user input to a computerized business process may seem intractable, because so many possible inputs are involved. The data within a computerized system is still limited and digitized, such that at least we can think of a finite limit to the possible event space of user inputs. In this article, the computerized business process is considered statistically to be a finite system with a finite limit to the number of combinations of inputs.

Simplification is achieved by breaking the process down into individual steps within a process (as done with process modeling), and to consider inputs of individual steps at first independently. Further, the user event space can be further simplified by classifying user input to be one of three basic possible events:

- user makes no input (e.g., optional field, function not initiated)
- user makes incorrect input
- user makes correct input (to meet quality objectives)

Even with this simplification, the number of possible combinations of user inputs is usually large. For example, during the design of an entry screen, it may be planned to have m required fields and n optional fields, leading to a total of $m+n$ independent variables within this screen. The total number of possible combinations of inputs (user event space) is 2^m3^n. For a modest screen entry of three required fields and three optional fields, this number is 216, which can be employed as the statistical event space. The design space includes only eight members (which includes all correct combinations of optional fields).

A typical screen for material master data maintenance has about 20 data entry fields, for which circa three are typically required fields and the rest optional. The user event space for data entry in a typical screen is then $2^{3}*3^{17} \approx 10^{9}$, for which the design space is still large, ($1^{3}*2^{17} \approx 10^{5}$) because of the large number of optional fields. Material master data maintenance typically requires about a dozen such screens, such that in practice, very large event spaces are tolerated. The tricks to tolerance include extensive user training and experience, coupled with limited access and heavy reliance upon input restrictions and checks.

Still with such large event spaces, false inputs from users are inevitable, thus degrading the quality of the system data and performance. Recognition of the large potential for user error during design review could help balance the desire of users for optional fields, multiple selections, and fine granularity in data acquisition. As we all know, such desires are not deterred by cost factors. During the design, a limit should be set for the maximum event space of a user interaction (based upon experience).

Quality Risk Management (QRM) Methods for Design of Computer Systems

ICH Q9[4] describes a number of acceptable risk analysis methods for which the Failure Mode Effects Analysis (FMEA) is the most popular for identifying potential failures of a computer system, so that testing can be planned. FMEA is a bottom-up-analysis which starts with single component failures and yields estimations of their impact upon the system. Because it requires as a basis the specification of those components, it has limited usefulness when applied early in a project (iteratively throughout the lifecycle, as suggested in GAMP 5[1]). Risk analyses focused upon single component failures tend to miss the big picture, and usually are formulated by the solution provider. Risks caused by users typically receive scant attention.

Often the only risk management documentation available early in a project consists of a GxP assessment of the system or process. Although such assessments are useful for projects, they cannot substitute for a recognized QRM method. The only risk-based decisions obtainable from such an assess-

ment determine the scope of compliance documentation, e.g., validation documentation.

Fault Tree Analysis (FTA)[5] is a top-down analysis which starts with top undesirable conditions which should be conceivable early in the design phase. FTA is not commonly used in software development and the distinctions top-down and bottom-up are also not commonly known so that some explanation can be helpful here.

In the bottom-up FMEA analysis, one starts with an initiating event or fault, typically a software defect, and estimates its impact (consequences). The defect is a potential root cause for an undesirable event. Potential software defects are identified by examination of the software, and the FMEA is useful for risk ranking these potential defects based upon their probabilities and potential consequences. Although FMEA usually yields Risk Priority Numbers (RPN), the ISO standard[5] also recognizes qualitative approaches, i.e., simple rankings with this method.

By contrast FTA is useful when the potential initiating defects are not easy to identify, such as combinations of user errors. Here one starts with unacceptable top events and attempts to identify potential initiating events which can lead to them. It is thereby a top-down analysis. Where data is available, such as with mechanical systems, probabilities also can be associated with the defects, as with the FMEA method, and probabilities for the top events can be estimated. This is clearly not feasible for user errors. The ISO standard[5] also recognizes qualitative approaches using the FTA method.

Figure 1 illustrates conceptually the differences between the risk assessment approaches in terms of consequences and faults.

In the early design stages, the business requirements and a conceptual software solution are available, from which a top-down analysis can be started. The analysis leads to the identification of defects or errors which can lead to a top event. FTA allows analysis of multiple errors, which certainly need to be considered where multiple user inputs are involved. Attention should be given to potential human error before a design is completed, such that the design can be intelligently reviewed, and the future users of the system can be informed of what needs to be addressed in training.

For automation projects having little direct human interaction, an early FTA application can still identify critical software modules or functions to be targeted for a risk-based approach to qualification. It could replace the typical project GxP assessments with the advantage that the potential impacts, i.e., top events, are also identified and associated with the software components.

In summary, utility is seen for an early application of FTA to identify primary risks, particularly for business processes, in order to improve the design of the user interface. Critical software modules identified during the top-down analysis can later be targeted for a bottom-up analysis. Risks associated with SW defects, which can only be fully appraised when the specifications are available, may be best analyzed via a bottom-up analysis later in the project so that risk ranking can be assigned.

Example: FTA Applied to a Complaint Handling Process

Non-compliant complaint handling is frequently cited by the FDA in Warning Letters[6] to pharmaceutical and medical device manufacturers. Particularly for medical device manufacturers, the letters also frequently cite a failure to report in a timely fashion injuries or potential injuries resulting from the malfunction or misuse of a medical device, (in the form of Medical Device Reports (MDRs)). The complaint handling process is clearly a business process, which typically involves use of software for registering and processing the complaints. Commercial (Off-the-Shelf (OTS)) software exists to support complaint handling, such that a hypothetical case study can be presented and suggested as reference.

Figure 2 provides a typical view of an entry screen that might be employed for complaint handling. Almost all of these fields are optional for creation of a complaint record, resulting in a very large user event space. It is clear that the standard configuration must be configured to limit user error. Clearly, user roles which limit access must be considered.

A precondition for applying FTA as a top-down risk assessment is an initial definition of the business process and software solution. Business processes are defined in this article by means of object-oriented process models, as is typically done in BPM.[7] A minimal definition of the system requires knowledge of system goals and the process workflow, including user roles and user inputs. Figure 3 provides a basic workflow process model of complaint handling, from the point of receiving the call and ending with the closure of the complaint. Figure 4 provides a more detailed model of the process chain, which includes actors and user inputs for individual process steps.

Taken from a balanced scorecard[8] or other information, a brief list of project goals for a complaint handling process for a medical device manufacturer commonly includes:

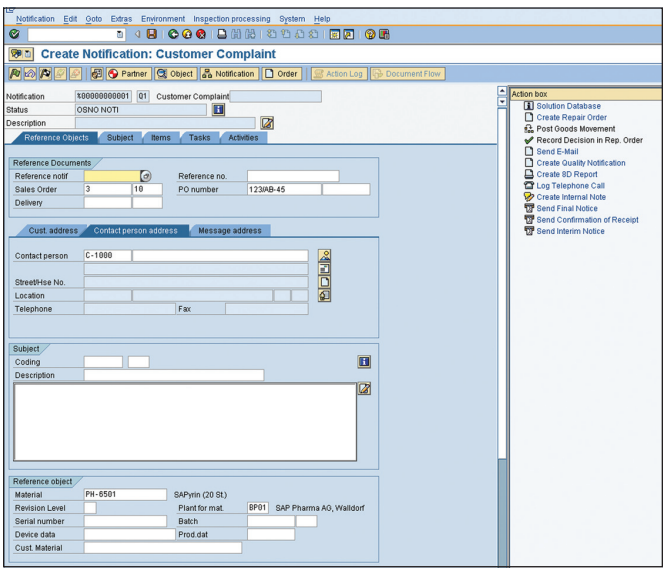

Figure 2. Typical entry screen for a new complaint record.

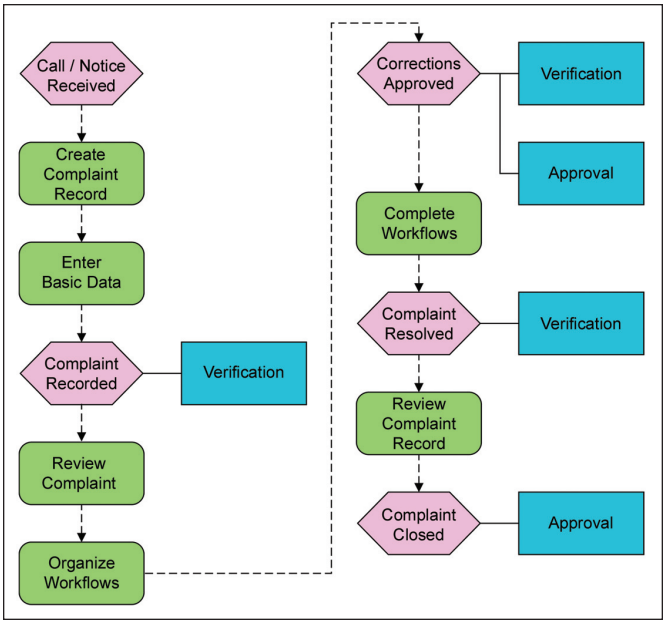

Figure 3. Complaint handling process.

- support of filing MDRs in a timely manner, as needed
- customer assistance with use of device (help desk)
- registration of product defects and/or malfunctions from the field
- registration of patient injuries or potential injuries
- support of filing internal Corrective and Preventive Actions (CAPAs)

If top events (potential impacts) of the system are not known or there is inexperience in recognizing them, they can be perceived by taking a goal (quality objective), and formulating a negative hypothesis. Another approach is to identify the compliance-relevant electronic records, which are processed by the system. Top events should include major errors in that processing, e.g., loss of integrity.

An example of FTA is provided in Figure 5, starting with the top event, "MDR not filed on time." Possible user input errors which can lead to the top event are listed with the relevant data element. Combinations of errors which lead to the top event are joined with the logical functions OR or AND. The identified data elements can be considered to belong to the key process parameters for the process.

The FTA diagram does not include possible software defects, which also can lead to the top event. The added complexity to the diagram would probably inhibit a useful review by the user group, and it should be clear that user errors can be considered separately from software defects. It is anticipated that such top-level analyses would be primarily reviewed by the process owners and users, who are not expected to have much knowledge of the software solution during the early phases of the project.

From an initial inspection of the model, the following characteristics of the process can be inferred:

- Multiple pathways can lead to this failure, (i.e., the model has breadth).
- Simple combinations of user errors can lead to failure, (i.e., the model has little depth).
- At least two user errors can directly cause the failure (i.e., there is a significant probability of failure).

Keeping in mind that call centers are often outsourced, and thereby not always closely managed, a mitigation strategy based solely upon user training and limited user access will not usually result in a highly reliable process. The two errors which can directly result in failure originate from the person taking the call: to open a complaint record; and to select from the system the correct record type. Mitigation strategies involving software enhancements that could eliminate or inhibit some of the branches are certainly conceivable by the reader and should be available as options during the early project phases.

The relevant process parameters for the user errors modeled in Figure 5 should be included in the list of key process parameters for the system. Design review should focus upon user entry of these parameters and consider:

- user access to the entry field
- selection option list
- possible plausibility checks

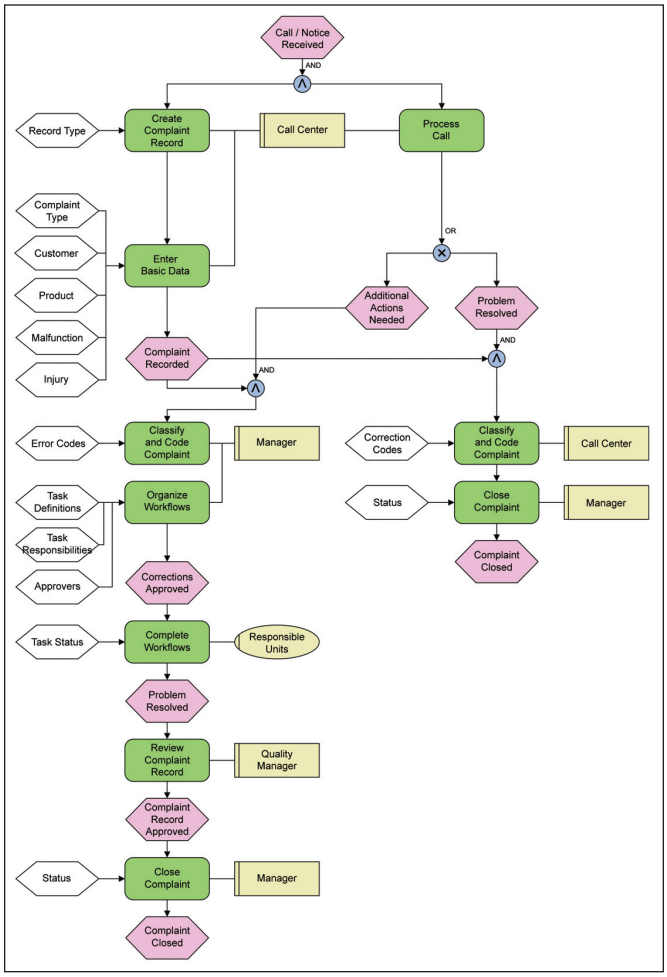

Figure 4. Complaint handling workflow.

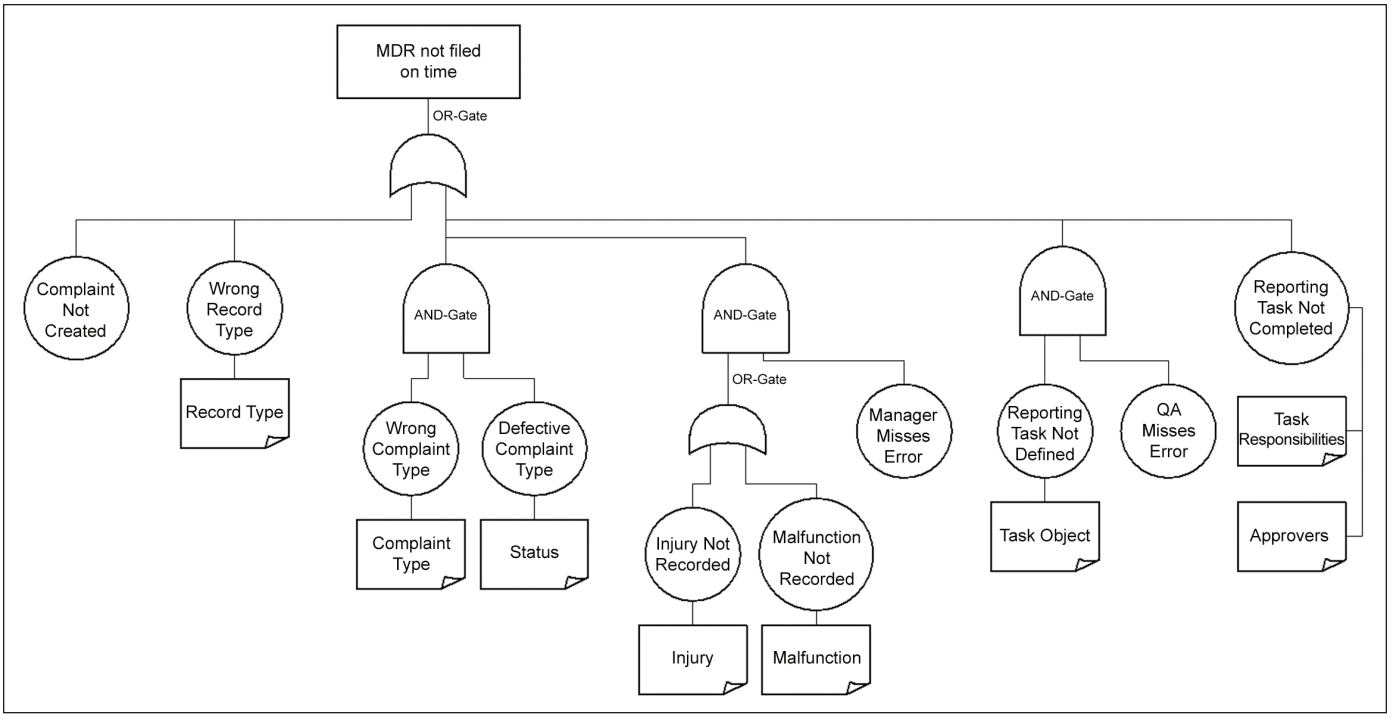

Figure 5. Fault Tree Analysis of user errors for top event, "MDR not filed on time."

To complete the top-down analysis, other failure scenarios would need to be analyzed similarly, starting from the project goals. For example, separate analyses also could be started from the top events "MDR is incorrect" and "Customer not helped." The number of such analyses can be limited by the number of goals set for the project and basically document the concerns addressed in the top-down analysis.

Although FTA is best for early analysis of combinations of errors, Figure 6 illustrates how critical software functions could be separately identified for the top event "MDR is incorrect." No detailed analysis of software is advisable at this level of detail, but FTA does directly associate basic functions with a top event and implicitly gives them a high ranking. Combinations of defects leading to this failure are not explored in this diagram. Such combinations would be expected to have a lower probability and thereby a lower ranking. Such an analysis could provide an orientation for detailed functional risk analyses later in the project.

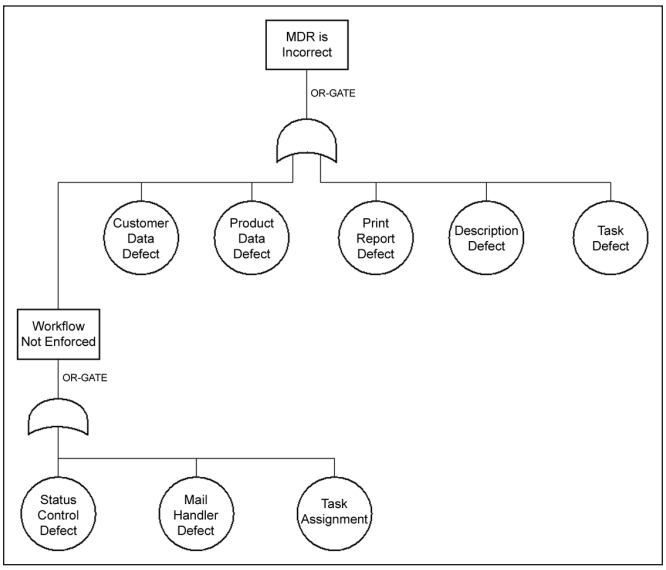

Figure 6. FTA of SW defects for top event "MDR is incorrect."

Upon completion, the design project has identified at an early stage the major risks of the system and cataloged the user errors which contribute to those risks. This catalog along with the failure scenarios would provide an excellent start for preparing training documentation and for subsequent functional risk analyses if used to identify critical software modules.

Conclusion

A top-level analysis should be conducted at the beginning of a project and helps to orient that project to address the major risks. It can be referenced for risk-based decision-making, and thus can guide early efforts for mitigating those risks. Preliminary employment of bottom-up analysis usually misses the "big picture" because dependencies and multiple failures are not easily included.

FTA is not a substitute for FMEA, in that it is not as useful for ranking and managing risks. When FTA is used early to identify critical modules, they can be transferred into FMEA for more detailed analysis. FTA is advisable for critical processes which are heavily dependent upon user input. It can be used to identify critical data and improve the design of user entry screens.

References

1. *ISPE GAMP® 5: A Risk-Based Approach to Compliant GxP Computerized Systems*, International Society for Pharmaceutical Engineering (ISPE), Fifth Edition, February 2008, www.ispe.org.

2. Murray, K. and Reich, S., "Quality Risk Management (QRM) Tool Selection: Getting to Right First Time," *Pharmaceutical Engineering*, July/August 2011, Vol. 31, No. 4, pp. 8-16, www.ispe.org.
3. International Conference on Harmonisation of Technical Requirements for Registration of Pharmaceuticals for Human Use (ICH), ICH Q8 (R2) Pharmaceutical Development, December 2008.
4. International Conference on Harmonisation of Technical Requirements for Registration of Pharmaceuticals for Human Use (ICH), ICH Q9 Quality Risk Management, November 2005.
5. International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), IEC/ISO 31010 Risk Management – Risk Assessment Techniques, November 2009.
6. FDA, Warning Letters under subject MDR, http://www.fda.gov/ICECI/EnforcementActions/WarningLetters/
7. Use of Modeling as Applied to Business Process Management, http://en.wikipedia.org/wiki/Business_process_modeling
8. Kaplan, R. and Norton, D., Harvard Business Review on Measuring Corporate Performance," *Harvard Business Review*, Harvard Business School Publishing, September 1998.

Acknowledgements

I wish to acknowledge the supportive feedback from my colleagues in the IDS Scheer Consulting group of Software AG and the ISPE (anonymous) reviewers.

About the Author

Dr. Paul Thomas Noble is a Senior Consultant for IDS-Scheer Consulting GmbH, a European IT consulting firm. Noble received his PhD in chemical engineering at the University of Wisconsin, USA in 1981 and shortly thereafter entered the pharmaceutical industry at Bayer's U.S. blood plasma fractionation facilities. Starting in 1989, Noble began a consulting career in engineering, validation, and compliance in both the USA and Germany. He has published on topics ranging from SIP and biotechnology to quality assurance. He can be contacted by telephone: +49-172-6868591 or email: paulthomas.noble@softwareag.com.

IDS Scheer Consulting GmbH, Niederkasseler Lohweg 189 Duesseldorf 40547, Germany.