FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Goal Driven Software Project Risk Management: Designing A Domain Model Aligned To COSO-II-ERM-Framework

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Christoph Schiessl
Matrikelnummer 0826016

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: Univ.-Prof. Dr. Walter Schwaiger

Wien, 03.04.2016 ______________________ ______________________

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Erklärung zur Verfassung der Arbeit

„Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.“

Ort, Datum, Unterschrift

Acknowledgements

I would like to thank Prof. Schwaiger for the possibility of writing the master thesis at the institute for management science at the TU Vienna with his outright advise the entire time of creation. All time long Prof. Schwaiger was available for meetings and gave professional and adequate consult and support. As expert in financial control and risk management he made helpful suggestions an recommendations about thesis finding, definition of the research question, general advise about risk management, the COSO risk management framework understanding and many more. Therefore Prof. Schwaiger contributed an essential part for the successful completion of this master thesis.

Abstract

Risk management in software projects is crucial for success but also a very complex issue. Software project managers face multiple difficulties to manage software risks in an adequate manner. Different software risk management frameworks exist in literature to support software project managers in their business. Some of them also provide conceptual models which can be used to build a better understanding of the framework as well as provide a basis for further implementations in software development i.e. automated risk management tools.

This work takes up a goal-driven software project risk management framework (GSRM) with the aim to answer the research question about how to design and align a GSRM domain model concerning COSO-II-ERM Framework components and methods.

The methodological approach starts with an analysis of a state-of-the-art GSRM domain model about missing concepts in alignment to COSO-II-ERM-Framework components and enhances the model to fulfill those requirements. Further a database is implemented according to the designed domain model and deals as basis for a proof of concept approach.

The results of this work show that the designed GSRM domain model in alignment to COSO-II-ERM Framework is valid and applicable for further implementations and supports software managers in the task of manage software risks in a successful way.

Risikomanagement in Softwareprojekten ist essenziell für den Erfolg, aber auch eine sehr komplexe Angelegenheit. Softwareprojektmanager sehen sich mit einer Vielzahl von Schwierigkeiten in Bezug auf das adäquate Managen von Software Risiken konfrontiert. Es existieren verschiedenste Frameworks im Bereich des Software Risikomanagements, welche Softwareprojektmanager in deren Tätigkeit unterstützen. Einige dieser Frameworks stellen konzeptionelle Modelle zur Verfügung, welche zur Verständnisgenerierung der Frameworks aber auch für weitere Softwareentwicklungen, wie z.B. automatisierte Risikomanagement-Tools, verwendet werden können.

Diese Masterarbeit greift das Zielgetriebene Softwareprojekt Risikomanagement Framework (GSRM) auf und hat zum Ziel die Forschungsfrage, wie ein Domänenmodell für GSRM erstellt und in Bezug auf die Komponenten des COSO-II-ERM-Frameworks, angepasst werden kann.

Die methodische Ansatz beginnt mit einer Analyse eines State-of-the-Art GSRM Domänenmodell und Identifiziert fehlende Konzepte unter der Betrachtung des COSO-II-ERM-Frameworks. In weiterer Folge wird das Model um die fehlenden Konzepte erweitert um im Anschluss daran zur Datenbankmodellierung herangezogen werden. Abschließend wird die Datenbank für einen konzeptionellen Beweis verwendet.

Das Ergebnis dieser Masterarbeit beweist sowohl die Validität dieses, unter Betrachtung des COSO-II-ERM-Frameworks, erstellten GSRM Domänenmodells als auch die Verwendbarkeit des Modells für weiterführende Implementierungen und zur Unterstützung von Softwareprojektmanagern in der erfolgreichen Durchführung deren Aufgaben im Softwarerisikomanagement.

Table of contents

1. Introduction

1.1 Motivation and Problem Statement

[Isla09] investigated the problems current software development projects face. Therefore he mentioned that one out of five software development projects failed and almost half of those projects had problems in their budget, time and function achievement. He analyzed the main reasons for those problems which are of technical nature but much more a reason of poor management of people.

Risk Management in Software Projects became more and more important in the last decades. Failure in controlling uncertainties such as time-to market, budget, schedule estimations, technology and stakeholder’s expectations imposed potential risks [Isla11]. Because of the increasing complexity and demands of Software Projects Risk Management has become a vital part in managing Software Projects [Bro95]. Several risk management methods exist in the literature. [Isla11] draws attention on the importance of integrating software risk management in the early development phase and in requirements engineering. They detected lacks in the literature about this integration and therefore designed a promising approach of goal-driven software risk management (GSRM).

According to the wide-spread use of COSO ERM Framework and the advance for enterprise risk management the COSO ERM Framework can be also used to support Software Risk Management [COSO]. In [Schw12] a cybernetic management framework is designed which translates management into actions. Among others this framework is aligned to COSO ERM framework and can be applied to risk management process.

[DHMM10] focuses on Information System Security Risk Management (ISSRM) and addresses also the importance of risk management in early management phase. They mentioned the poor modeling support of recent and common risk management methods. Therefore they designed a domain model as conceptual model for ISSRM supporting risk managers in understanding and comparing several risk management methods.

Such ontologies can also be used for implementation of risk management tools. Zachman invented in [Zach87] the so called Zachman’s Framework which describes enterprise architecture and the related views. Described in this framework such ontology is assigned to the designer’s perspective which is the interface between the owner and the builder/manufacturing engineer.

Focusing on the design of a domain model the designer has to consider both views the user and programmer's view. Because GSRM can be seen as the state-of-the-art for software project risk management as well as COSO ERM Framework as standard for enterprise wide risk management literature lacks in an adequate domain model from GSRM considering both user's and programmer's view as well as COSO II ERM Framework's components. The actual GSRM domain/meta model shows problems for both views.

The research question therefore is framed as follows:
"How can a domain model for a GSRM process be designed in alignment to the operations view of COSO II Framework ?"

1.2 Aim of the work

The aim of this work is to design a conceptual model as domain model of state-of-the-art goal-driven software project risk management processes. The model should be aligned to COSO II ERM framework by highlighting the operations view. Therefore the activities in the cybernetic MGT framework are used. As proof of concept a MSSQL-DB, representing the designed domain model will be implemented with MSSQL Management Studio. To proof the domain model under usage of the data base test data is applied from the literature. To finalize the work the statement of validity and utilization of the model and database should be made.

1.3 Methodological Approach

The methodological approach consists of the following steps:

1. Literature Analysis with respect to goal-driven software project's risk management methods:

Current goal-driven software project risk management is analyzed starting from initially contributed risk management frameworks introduced in [BoRo89] to a state-of-the-art goal-driven software development risk management in [Isla11].

2. Analysis of current goal-driven software project risk management domain model with respect to COSO-II-ERM-Framework:

After a literature analysis a goal-driven software project risk management domain model, representing the starting point of analysis, is matched against COSO-II-ERM-Framework's components [COSOII04] with the result to identify the missing concepts and relationships of the model dealing as basis for further alignment.

3. Enhancement of the goal-driven software project risk management domain model under consideration of the analysis results from step2 with respect to the Generic Cybernetic Management Framework:

After an analysis of the GSRM domain model and identification of missing concepts and relations and further elements the model is aligned and enhanced due to the approach relating Generic Cybernetic MGT Framework in [ScAb13] to result in an accurate, complete, conflict-free and non-redundant goal-driven software project risk management domain model fulfilling all COSO-ERM requirements as well as including all COSO-ERM components.

4. Database modeling:

To acquire an adequate analysis basis and starting point for the further proof of concept and validity of the designed GSRM domain model from the previous steps in the methodological approach a database has to be modeled. For database model design MSSQL Management Studio is used. The database is modeled according to the designed domain model and enriched with test data from the literature which in further consequence is used for analysis and validation proof of the model.

5. Proof of concept:

The designed database model with test data is then applied for further use case evaluation defined in a previous sub step. Therefore the final statement should be made ensuring the validity of the designed domain model by proofing the utilization of the database model.

The applicability for further implementation i.e. in software development should also be made but is in detail not covered in this work.

The steps are shown in the following figures:

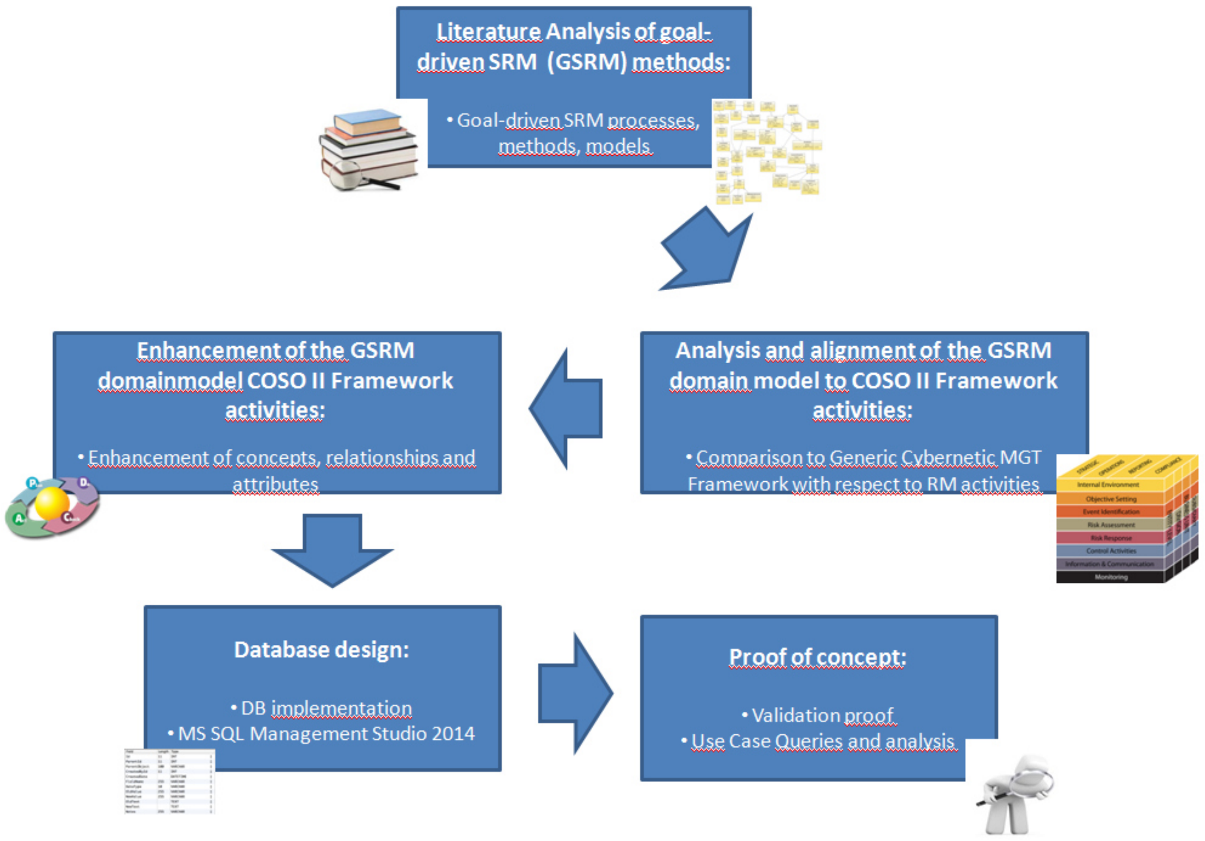

Literature Analysis of goal-driven SRM (GSRM) methods:
• Goal-driven SRM processes, methods, models
Analysis and alignment of the GSRM domain model to COSO II Framework activities:
• Comparison to Generic Cybernetic MGT Framework with respect to RM activities
Enhancement of the GSRM domainmodel COSO II Framework activities:
• Enhancement of concepts, relationships and attributes
Database design:
• DB implementation
• MS SQL Management Studio 2014
Proof of concept:
• Validation proof
• Use Case Queries and analysis

Figure 1 - Methodological Approach

1.4 Structure of the work

The structure of this work is, in a raw point of view, segmented in an introduction of the work, a state of the art analysis and analysis of existing approaches, the methodology introducing the used concepts, methods and models, the suggested solution representing the implementation, a critical reflection and a conclusion of the work in general as well as an prospect for future work finalizing in the bibliography.

Section 1 will give an introduction of the work including problem statement and motivation, aim of the work and a short overview of the methodological approach which is separated in five subsections.
Section 2 is about the state of the art analysis starting with a literature studies about risk management frameworks continuing with an analysis and comparison of existing approaches with respect to the aim and in general the superior topic of this work.
Section 3 covers the methodology which is divided in an definition of used concept with respect to software risks, events and likelihood, risk management in general and the concept of a domain model. The next subsection provides background information about methods and models which are essential for the methodological approach of this work. This

subsection distinguishes between generic risk management methods and models and specific risk management methods and models. Another subsection is addressing the major modeling language used in this work which is UML and cuts into UML object diagram and UML activity diagram. The section also includes a subsection listing the primary design methods which are separated in general domain model design methods, design methods for risk management domain models and database design methods. Finalizing this section the substantially analysis method for model validation of the designed domain model are presented.
Section 4 comprises the suggested implementation methods and solution of the previously mentioned problem statement. Starting with an analysis of current goal-driven software project risk management domain model with respect to COSO-II-ERM-Framework, the enhancement of the goal-driven software project risk management domain model under consideration of the analysis results from the previous subsection with respect to the Generic Cybernetic Management Framework, the database design for the succeeding proof of concept.
Section 5 gives a critical reflection including a summary of existing approaches as well as a discussion of open issues.
Section 6 summarizes the work and discusses the utilization and view for future approaches and works in the context of goal driven software project risk management, the designed domain model and the implemented database.
Section 7 lists the main figures in an appendix.
Section 8 consists of a bibliography referencing the sources of the literature on which the work is based.

2. State of the art / analysis of existing approaches

2.1 Literature Studies

The main focus of this literature studies are software risk management frameworks to be geared towards GSRM.
[Isla11] did a detailed literature analysis of risk management frameworks contributing to the design of the state of the art approach of a goal driven software project risk management framework. The main frameworks leading to this approach are mentioned as follows.
Boehm [Boe91] initially did a contribution to software risk management frameworks. who designed the first software risk management framework called Boehm's risk-driven Spiral Model in which he integrated risk management iteratively in the software development life cycle.

This approach is followed by Software Engineering Institute's (SEI) framework for risk evaluation [SJ94] which includes identification, analysis, communication and mitigation for software risk management and is central based on risk taxonomy and questionnaire from [CKM+93] consisting of the most important software development process elements as well as elements relating to management process and methods. In [ADH+96] the SEI Continuous Risk Management Guidebook, practical techniques for continuous risk management in all phases of the software development life cycle are provided.

In [Kar95] the Software Engineering Risk Model framework is proposed considering technology as well as business perspectives. Therefore software development is defined under the view of just-in-time approach. 81 risk factors are taken under account in combination with three main risk elements declared as technology, cost and schedule. These risk factors are associated with a risk measure and question utilizing a probability tree driven from calculating risk factor values.

Kontio [Kon01] also considers goals or expectation in his approach of a conceptual framework for risk management called Riskit methodology where risks represent threats over the course of reaching these goals. It is a model based approach in which risks are modeled and documented qualitatively under analysis of risk factors, risk events, risk effect sets and utility loss. This approach ranks risks by the Pareto ranking technique and supports managers from risk identification and analysis of risks through to monitoring and controlling them.

An integrated risk management process is proposed by Prokaldnicki in [EPYA06] especially for global software development (GSD). In [PNJE06] a reference model is developed
consisting of four phases including new project, project allocation, project development, evaluation and feedback with an individual process for each phase. Starting with project identification followed by risk assessment, which assists offshore development, further to communication of risks and decisions continuing to the operational level.

S. Islam [Isla11] designed a goal-driven risk management approach for software development. He focuses on goals in software risk management, main goals relating to schedule, cost and quality as well as goals like offshore, security, safety, compliance, business process and cultural diversity relating goals. Considering the fact that fixing errors in later phases of software development are much more cost intense and effortful as in earlier development phases he designed a goal-driven software risk modeling (GSRM) framework which aims in integrating risk management activities within the requirement engineering phase.

He considered and adopted existing modeling techniques and goal modeling languages like KAOS and i*/ Tropos to construct a GSRM framework contemplating technical as well as non-technical development components. KAOS and i*/Tropos [BPG+04] are commonly used methodologies for Goal-oriented Requirements Engineering (GORE) in the RE process where KAOS (Keep All Objective Satisfied) is the main foundation of his work and aims to model every aspect of requirements (what, how, why, who, and when) [DvLF93, vL09]. Goals in KAOS can be descriptive or prescriptive, soft or hard, functional or non functional and should be satisfied by the cooperation of the system's agents. The goal model, where each goal is represented graphically, consists of higher-level goals and lower-level goals their relation and conflicts among them.

2.2 Comparison and analysis of existing approaches

Except Islam's GSRM approach from [Isla11] most of the frameworks are not using particular processes to assess and manage risks. They often follow the same outdated process i.e. checklists or expert analysis. There is also an overall emphasis of integration of risk management as early as possible in the software project phase but unfortunately detailed instructions and activities for this integration are mostly missing. The GSRM approach takes up this omission and integrates risk management in requirements engineering in reference to a goal-risk model. Based on the circumstance that most software risk management framework focus on limited goals like in general related to schedule, cost and quality GSRM also mentions the importance of other goal instances as well like safety, compliance, maintenance, supporting business processes and coordination of projects under diverse circumstances in culture and location. Therefore GSRM is an effective approach for software risk management and its consideration in the early development phase. It also shows the importance of analyzing not only the technical perspective but also non-technical issues as well which are often overlooked in software development. But GSRM is limited i.e. to the scale of projects and the increasing amount of goals in large projects which are more difficult to manage in large-scaled projects. GSRM also depends on the context of projects because the estimation of specific goals and their importance can differ in stakeholder's opinions. A more conceptual approach of GSRM by elaborating a domain model aiming to integrate this in further development methods is also missing.
Therefore this work should help to decrease those limitations and design such a model.

3. Methodology

3.1 Used Concepts

This section covers the main concepts which will be used in this work. The concepts will refer to Software Risk Management as it is the main underlying topic but also to more generic ones.

3.1.1 Software Risk

According to the ISO-Guide 73:2009 risk is the "*effect of uncertainty on objectives"* [ISO-09, Ch.1.1], where an effect is defined as deviation from the expected and can be positive and/or negative. Risks are associated with other basic concepts like events, consequences and likelihood. Software risk is a special form of risk. In [Isla11] it is defined as, *"the possibilities of suffering a loss such as budget or schedule over-runs, customer dissatisfaction, poor quality and passive customer involvement due to an undesirable event and its consequences during the life cycle of the project."* [Isla11, P.12]

3.1.2 Risk Event and Likelihood

Events can sometimes be referred to an "incident" or "accident" and are defined as *"occurrence or change of a particular set of circumstances"* [ISO-09, Ch.3.5.1.3] whereas an event can have multiple occurrences and is not exclusively depended on something happening but can also consist of a missing eventuation.

Likelihood is defined as the *"chance of something happening"* [ISO-09, Ch.3.6.1.1]. It is related to the mathematically used term "probability" and in risk management used in combination with events. Therefore likelihood is defined, measured or determined in the objectively or subjectively way as well as qualitatively or quantitatively.

3.1.3 Risk Management

An abstract point of view of the definition of risk management is *"coordinated activities to direct and control an organization with regard to risk"* [ISO-09, Ch.2.1]. Risk Management is always related to a risk management process including activities like risk identification, risk analysis, risk evaluation and treatment as well as risk monitoring. Also important to consider are terms like risk management policy and risk management plan. Therefore risk management frameworks are used to continually improve risk management through the organization and provide certain *“foundations and organizational arrangements”* [ISO-09, Ch.2.1.1] for risk management including among others policy, objectives, plans, processes, etc. These frameworks are also embedded in the organization’s strategic and operational policies and practices.

3.1.4 Domain Model

A model in general abstracts the real world and can have different character. A domain model in particular is "*an object model of the domain that incorporates both behavior and data*" [Fow03, P.116]. A domain model combines both data and process and consists of object with multiple and various attributes. They are related to other objects and form complex associations. Objects represent both data in the business as well as business rules which describe the behavior of the model. Domain models also use inheritance. Their aim to solve certain problems (i.e. software design, implementation) or as they represent reality they can be used for communication in a non-technical manner. In order to implement or design such models UML [UML11] Unified Model Language is applied combining objects and activities in UML activity diagrams.
In this work a domain model is represented as data model for further implementation. According to the Zachman Framework [Zach87] this kind of model is associated with a Logical Data Model or System Model belonging to the Designer's perspective which inherits the role of an architect lying in between of the Owner's perspective and the Builder's perspective.

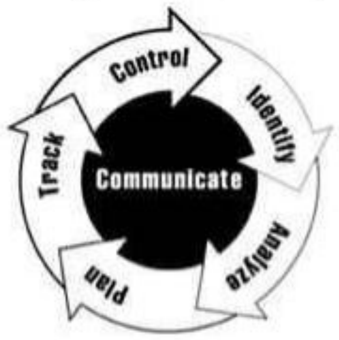
3.2 Methods and models

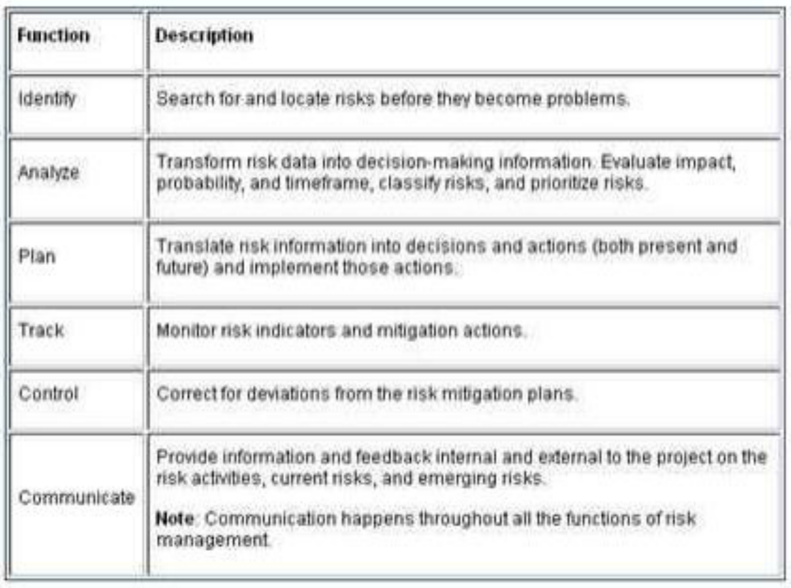
In this section important methods and models are investigated which are relevant for this work. As this work consists of different kinds of models in their structure and use like domain models or process models as well as different kinds of areas this section is subdivided. [JIL15] defined in their research about effective software risk management processes three different kinds of risk management process models which are General Software Development Risk Management Processes and Specific Process Oriented Risk Management Models. In this thesis these definitions will be adapted in a similar way.

3.2.1 Generic Risk Management Methods and models

According to [JIL15] this section covers the generic risk management methods relevant in this work under the premise that they “…can be applied to all types of projects related to a specific industry.” [JIL15, P.838]:

Generic risk management models and methods are researched in [SMJ14] where they identified four major approaches of risk management in software projects which encompass risk list, risk-action list, risk-strategy model and risk-strategy analysis. A risk list is a collection or list of prioritized risk items supporting the project manager in assessing risks. In this approach risk items are classified in system items, which occur in most software projects and specific items, which are related to project characteristics and organizational context. Although risk lists are easy to use they do not help identifying relevant actions which is obtained through risk-action lists where every risk item are prioritized and associated with resolution actions. The advantage is also the easy use, buildup and modification but as this approach focuses on isolated pairs of risk items it does not emphasize on a risk addressing policy.
Risk-strategy models relate collective risk items to aggregate resolution actions. They arrive a risk profile through extraction of risk categories, and afterwards through abstraction of action categories they arrive at a general risk strategy. They are easy to use and facilitate managers understanding the risk profile. Nevertheless such models are difficult to build up and modify because complete comprehension of influencing factors in risk profiles is needed.
Risk-strategy analysis is a stepwise process of linking risks to a complete risk management strategy. It is similar to risk-strategy model but with a looser coupled relationship of aggregate risk items and aggregate resolution actions. Because of this it's easier to modify than risk-strategy model.
As basic risk management process they mention also a model proposed by Software Engineering Institute (SEI) shown in the following figure:





Function	Description
Identify	Search for and locate risks before they become problems.
Analyze	Transform risk data into decision-making information. Evaluate impact, probability, and timeframe, classify risks, and prioritize risks.
Plan	Translate risk information into decisions and actions (both present and future) and implement those actions.
Track	Monitor risk indicators and mitigation actions.
Control	Correct for deviations from the risk mitigation plans.
Communicate	Provide information and feedback internal and external to the project on the risk activities, current risks, and emerging risks. **Note**: Communication happens throughout all the functions of risk management.

Figure 2 – Risk Management Process [SMJ14, P.848]

These functions represent continuous activities in a project life cycle.
The study results in the conclusion that each of these approaches is suitable for different kind of software projects. They also mention the fact that because of the complexity of software projects and the interlinkage of software project's stages risk management is important in every one of them.

The COSO-II-ERM Framework was introduced by the Committee of Sponsoring Organizations of the Treadway Commission in 2004 [COSOII04]. COSO embedded risk management in the enterprise management context. In the Enterprise risk management an enterprise therefore faces in all its entities uncertainties which influence the targeted value and value creation. Those uncertainties are called events which can have negative impact representing risks and positive impact representing opportunities.
Enterprise risk management is process oriented in detail it's defined as:
"Enterprise risk management is a process, effected by an entity's board of directors, management and other personnel, applied in strategy setting and across the enterprise, designed to identify potential events that may affect the entity, and manage risk to be within its risk appetite, to provide reasonable assurance regarding the achievement of entity objectives." [COSOII04, P.2].
Additional ERM has effects on every level of the organization including the human resources and is also applied in strategizing activities.
Figure 3 shows the COSO-Cube which includes the components of enterprise risk management, the achievement of objectives and the relationship of objectives and components representing the three dimensions of the matrix or cube:

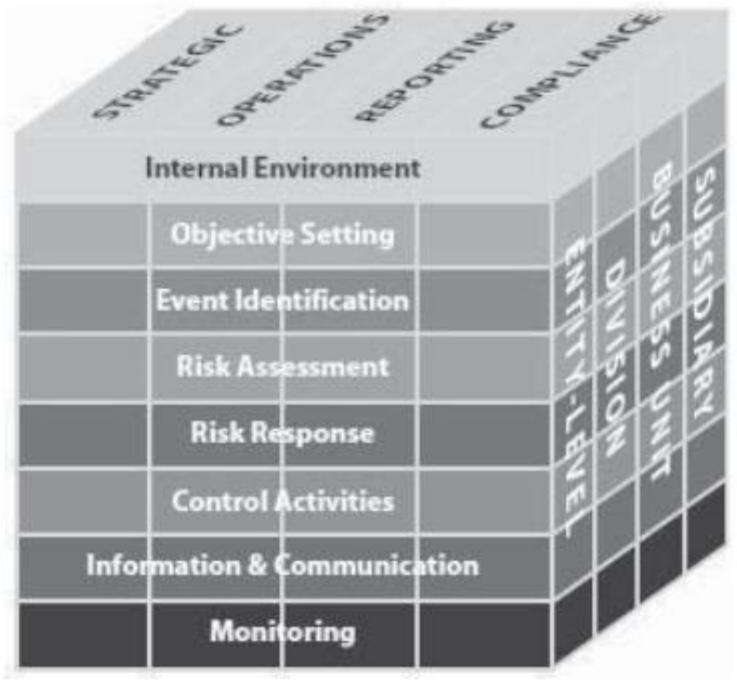


Figure 3 - COSO-Enterprise Risk Management Framework – COSO Cube [COSOII04, P.5]

COSO derives the components in eight interrelated elements derived from management activities:

- Internal Environment: This component incorporates the basic assumptions how risk is managed in an organization. Laying the foundation of ERM in a philosophical, ethical, environmental and cultural (risk and organization) manner.
- Objective Setting: To identify risks or events enterprises must set objectives in the first place. Objective setting includes defining an organization's risk appetite which has to be aligned with shareholders (i.e. stakeholder, key employees, suppliers, customers, etc.).
- Event Identification: The identification of events whether they are intern or extern, risks or opportunities is an important part in Enterprise Risk Management. Opportunities therefore effect also the enterprise's strategy reactively. Risk categories are helpful and can be used to react forgetting important risks.
- Risk Assessment: This component includes the probability and impact analysis of an event which requires an adequate precedent event identification.
- Risk Response: Obviously this is the most important component because of the impact the decision of an organization has whether to avoid, reduce, accept or to share a risk. An organization has to calculate with the consequences each risk management activity has (i.e. reducing a risk means to implement control activities and therefore consume resources).

- Control Activities: In COSO II ERM Framework control activities enlarge traditional ones which is not only about reducing probability or frequency of risks (preventive) but also about reducing cost impacts (detective).
- Information & Communication: Effective information systems and communication channels in an organization is a crucial part in ERM Frameworks. Timely information in order to make appropriate management decisions are essential.
- Monitoring: Monitoring is an important activity not only in the context of ERM. In ERM organization's use it to make decisions about expanding the framework (i.e. performing separate risk assessments, refining and response to them).

The objectives, representing another dimension in the cube, can be separated in different risk categories and corresponding types of risk [COSOII04, BH05]:

- Strategic: High-level goals and objectives which include risks in the context of governance, strategic objectives, business models, external forces, etc.
- Operations: Objectives for the efficient use of an organization's resources (i.e. risks of business processes, value chains, financials, etc.). This part of risks is also under main consideration of this work.
- Reporting: Objectives in reliability to reporting enforcing risks i.e. in the context of information technology, financials, internals, reputations, etc.
- Compliance: Means objectives treating with applicable laws and regulations, i.e. environmental, legal and contractual risks.

The third dimension of the cube represents the relationship between objectives and components as four different organizational levels of Enterprise Risk Management focusing *"..on the entirety of an entity's enterprise risk management, or by objectives category, component, entity unit, or any subset thereof"* [COSOII04, P.5].

3.2.2 Specific Risk Management Methods and models

This section mentions a few risk management process plans in the field of software development risk management but also in general. Their intention is to make current practicesof risk management more effective and therefore add or invent new practices in the field of (software) risk management:

[Schw12] in further consequence of the integration of risk management in enterprise management designed the cybernetic management framework in alignment to risk management called cybernetic risk management framework. Therefore the three cybernetic

principles are used: feedback principle and control and communication principle introduced by Wiener [Wiener48] and the double loop principle established by Foerster [Foer03]. The cybernetic management framework translates management into action by modeling management processes in an UML activity diagram. Giving a mental meta-model for any kind of managers, the framework can be used in different management domains e.g. strategic management processes in a way that risk management consideration are included.
For the integration and translation of risk management into action, [Schw12, p.428] used three different risk management frameworks:

- BCBS-risk management framework,
- ISO-risk management framework and
- COSO-enterprise risk management framework

While the measurement of risks (financial and operational risks) is adopted from the Basel Committee on Banking Supervision (BCBS) [Basel2-06], risk management process is implemented according to ISO-Risk Management Process [RMS09, p.14]. Also the Plan-Do-Check-Act (PDCA) cycle, the operating principle of ISO's management system standards is used as cybernetic feedback principle. [ISO-MSS11]

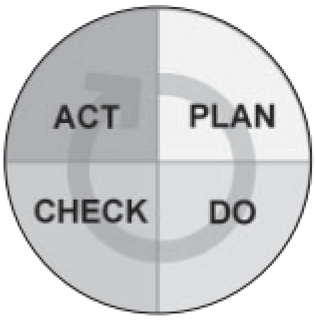


Figure 3 - ISO PDCA Cycle [ISO-MSS11]

The enterprise wide language for risk management is used from COSO-enterprise risk management framework [COSOII04, P.2] which:

- is process oriented,
- relates to strategic, operations, reporting and compliance objectives and
- is relevant for all organizational units of the enterprise

Using these frameworks and after translation of management into action first a cybernetic management framework (single and double loop) followed by a generic cybernetic framework have been designed. And finally a cybernetic risk management process as a supervised closed double loop is developed shown in Figure 5:

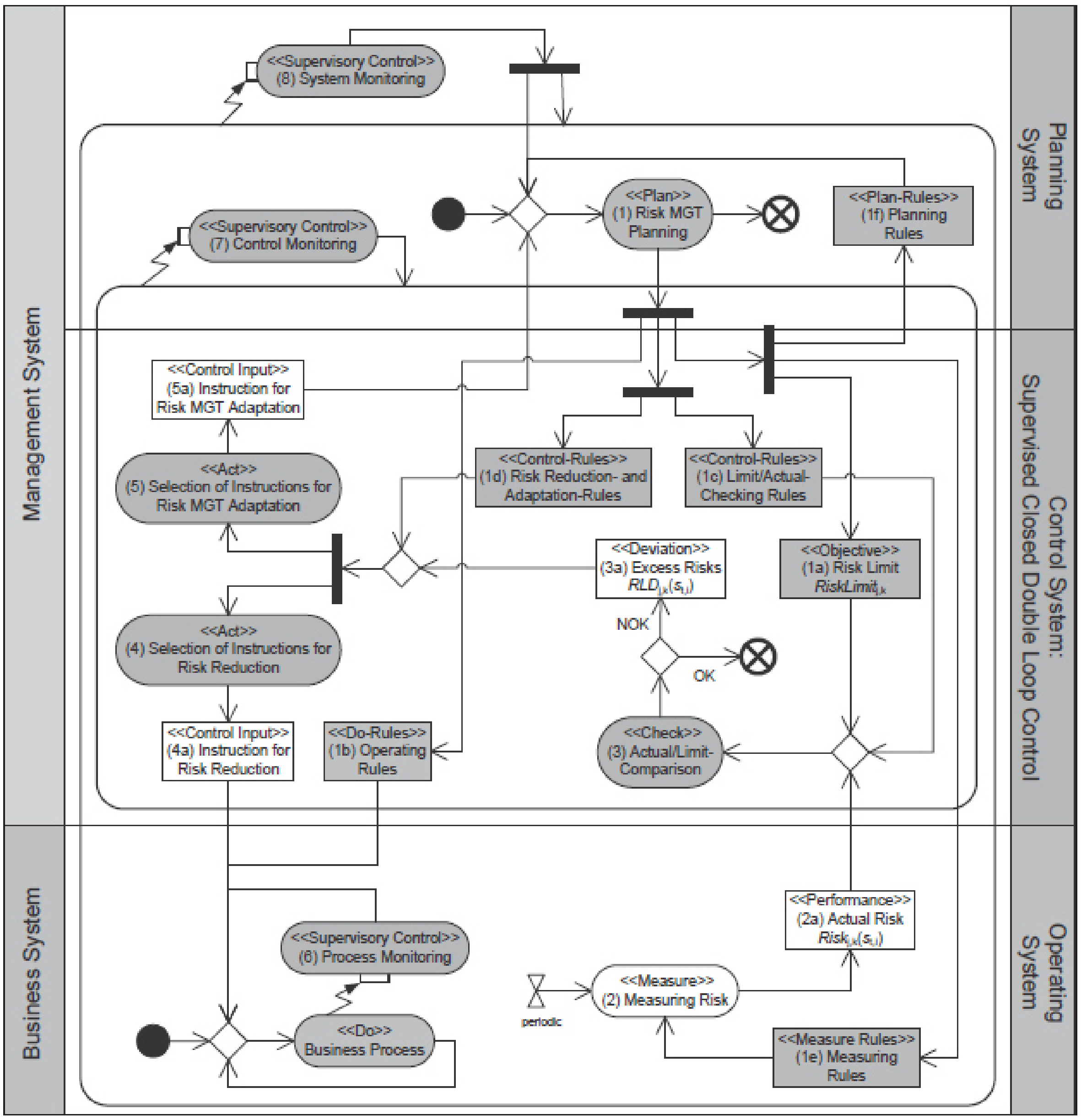

Management System
Business System
Planning System
Control System: Supervised Closed Double Loop Control
Operating System
<<Supervisory Control>> (8) System Monitoring
<<Supervisory Control>> (7) Control Monitoring
<<Plan>> (1) Risk MGT Planning
<<Plan-Rules>> (1f) Planning Rules
<<Control Input>> (5a) Instruction for Risk MGT Adaptation
<<Act>> (5) Selection of Instructions for Risk MGT Adaptation
<<Control-Rules>> (1d) Risk Reduction- and Adaptation-Rules
<<Control-Rules>> (1c) Limit/Actual-Checking Rules
<<Deviation>> (3a) Excess Risks $RLD_{i,k}(s_{i,j})$
<<Objective>> (1a) Risk Limit $RiskLimit_{i,k}$
NOK
OK
<<Act>> (4) Selection of Instructions for Risk Reduction
<<Control Input>> (4a) Instruction for Risk Reduction
<<Do-Rules>> (1b) Operating Rules
<<Check>> (3) Actual/Limit-Comparison
<<Performance>> (2a) Actual Risk $Risk_{i,k}(s_{i,j})$
<<Supervisory Control>> (6) Process Monitoring
<<Measure>> (2) Measuring Risk
periodic
<<Do>> Business Process
<<Measure Rules>> (1e) Measuring Rules

Figure 5 - Cybernetic Risk MGT Process – Supervised Closed Double Loop MGT [Schw12, P.438]

In a more generic context the model consists of eight managerial activities, which are:

- Plan-activity
- Measure-activity
- Check-activity
- Corrective Act-activity
- Adapting Act-activity
- Process related Supervision-activity
- Control related Supervision-activity
- System related Supervision-activity

The Plan-activity related to risk management consists of setting objectives in form of risk limits and defining rules for multiple elements like planning, measuring, controlling (adaption as well as check) and do-rules for operating processes.
The Measure-activity is the central part of the risk assessment which outputs in the actual risk which is then compared in the Check-activity to the risk limit defined in the Plan-activity. The deviation represents the output of the Check-activity in detail the difference between the actual risk and the risk limit.
According to the double loop character the Act-activity is spitted in Corrective Act-activity and the Adapting Act-activity.
While the Corrective Act-activity outputs in the instructions to reduce the risk in the business process, the Adaptive Act-activity defines instructions to take adaptive actions in the Plan-activity.
The Supervision-activities are divided in process related supervision, control related supervision and overall system related supervision. Those activities represent the information processes and exception handling actions corresponding to the relating system areas.

A more detailed correlation between the cybernetic risk management framework and the COSO II ERM framework [COSOII04] can be done with the usage of the COSO II cube shown in figure 3. Relating to the eight components of the COSO II cube described in section 3.2.1 activities of the cybernetic risk management framework can be matched to as follows:

Risk management planning defined in [Schw12] can be associated to several components from the COSO II cube. Internal environment, objective setting and risk identification are all covered in this activity. By definition of plan-, do-, measure- and control-rules the elements of internal environment are satisfied. Objective setting is related to the similar called information entity objective which is one of the outputs of the plan activity. Risk identification is indirectly included in the planning process and aims in the definition of control rules especially in the checking rules for which risks have to be identified in the first place.

The risk assessment component is based in the checking activity as well as the measurement activity where the deviation is determined which is a crucial part of risk assessment.

In further consequence risk response and preventive control activities can both be found in the corrective and adaptive act activities. The most important part of risk management is the choice of decision how to act on different outcomes of risk assessment which then serves as corrective or adaptive instructions closing the double loop framework.

The information and communication component consists of the different kinds of information entities of the cybernetic risk management framework i.e. all kind of rules, deviation, objective, performance and control inputs which are processed in the context of the closed double loop management.

Finally the monitoring component can be related to the supervisory activities of the framework in which also the detective part of the control activity component is based.

An integration approach of this framework is done in [ScAb13] where the cybernetic management framework is integrated into the REA business ontology which allows the semantic design and implementation of accounting-based management information systems. The REA accounting framework introduced by McCarthy [McC82]where R stands for economic resources, E for economic events and A for economic agents was then extended by Geerts and McCarthy [GeMc02] to the REA business ontology including also future related information and business policy considerations. The relationship between the three elements and the policy infrastructure which builds the REA business ontology is shown in the following figure:

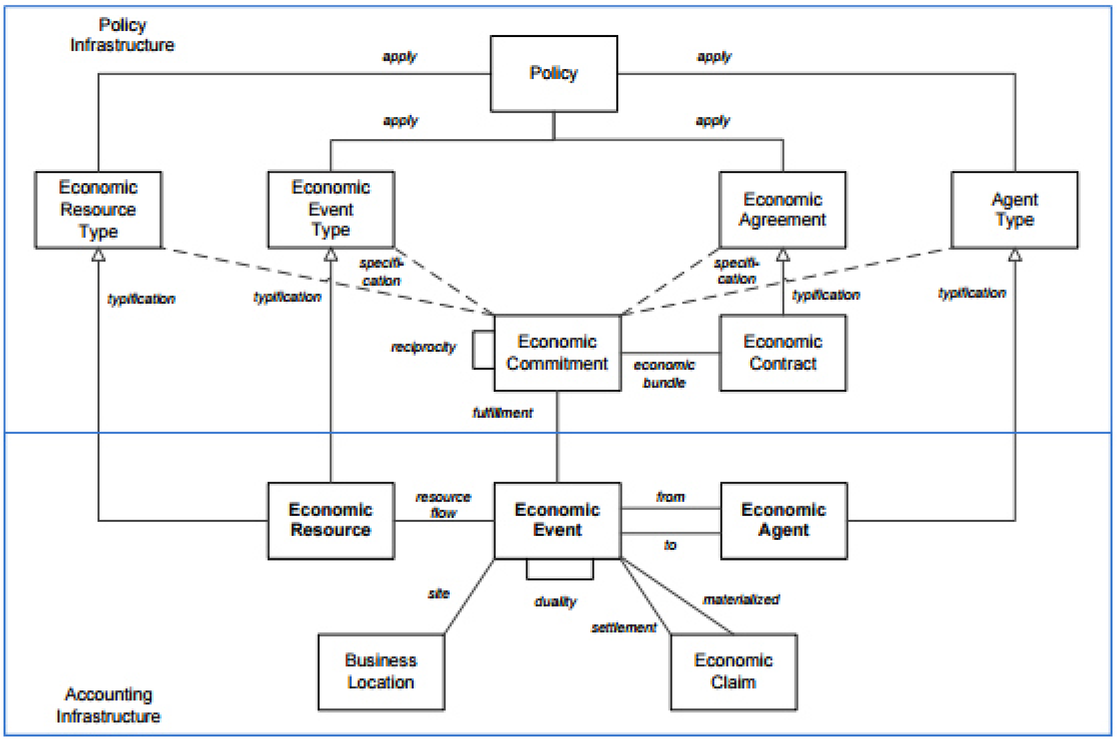


Figure 6 - REA Business Ontology [ScAb13, P.347]

After integration of the cybernetic management framework introduced by Schwaiger [ScAb13] and mentioned before in figure 6 they call it probabilistic REA management ontology extended by business and management policy and probabilistic events like plan and risk events resulting in the probabilistic policy infrastructure:

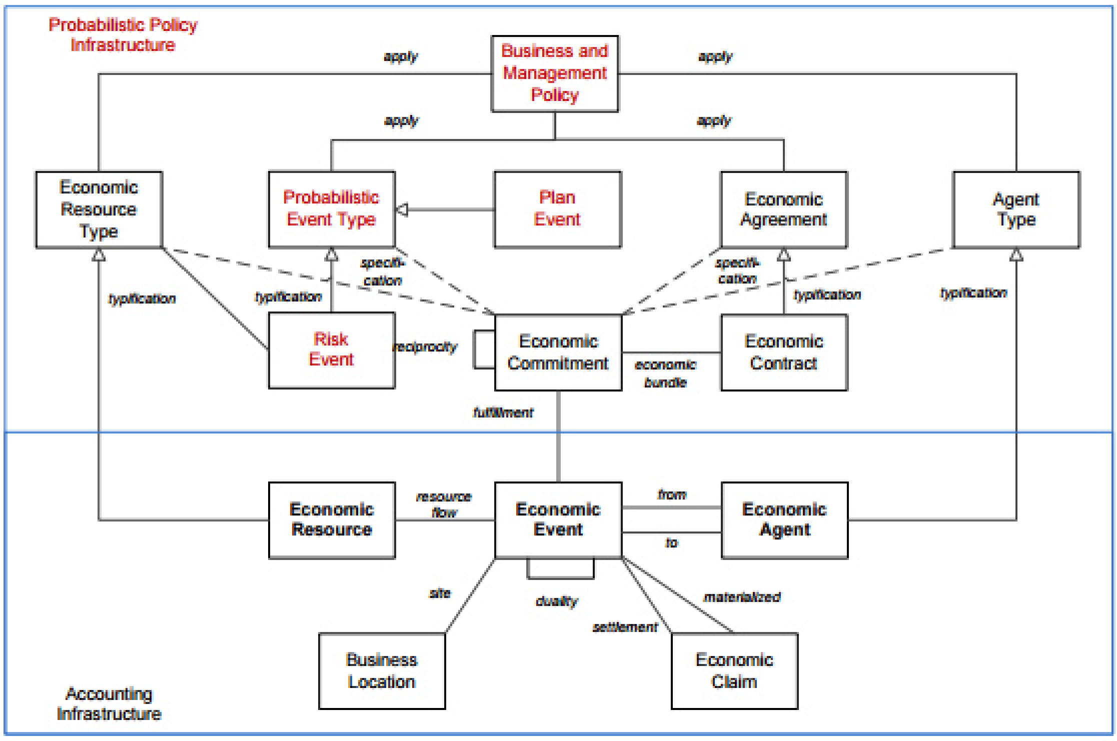


Figure 7 - Probabilistic REA Management Ontology [ScAb13, P.350]

"The probabilistic event type is represented in the simplest case as probabilistic binary tree and it is used to specify future events." [ScAb13, P.350] Plan and risk events represent future events which both inherit the probabilistic nature possessing values for the likelihood of their occurrence. Important is that the plan events are explicitly distinguished from the economic events in the REA business ontology. Plan events contain objectives in the planning process while risk events are used in the risk management systems. The probabilistic REA management ontology "*…allows the consistent design and implementation of accounting-based management information systems which include advanced management concepts in form of rational planning and stochastic optimal control systems"* [ScAb13, P.350-351].

A goal-driven Software Development Risk Management Model was designed in [Isla11]. They defined a layer based concept of four layers with the advantage of separation and therefore the performing tasks without affecting the other. These layers are:
Goal, risk-obstacle, assessment and treatment layer. The model is shown in the following figure:

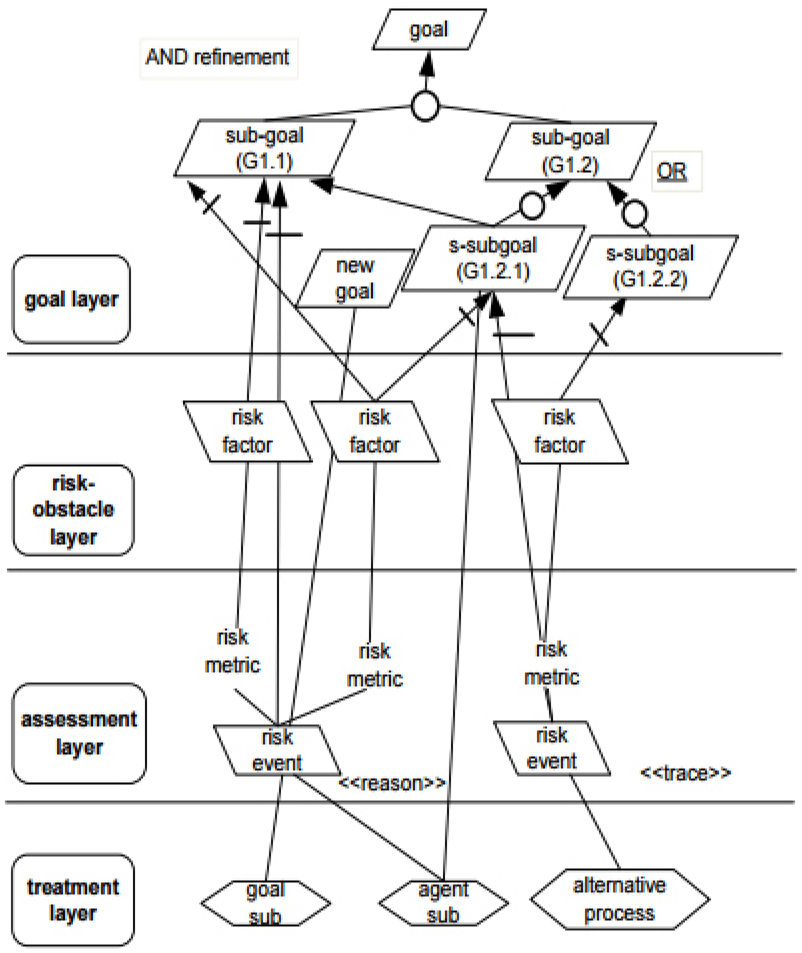


Figure 8 - Goal-driven software development risk management model [Isla11, P.65]

The goal layer contains the identification, elaboration and modeling of goals in the perspective of project success including meeting business objectives, completion in budget, time and function as well as customer satisfaction. The goal management is also contained in this layer which means achievement, maintainment, ensurance and improvement of these goals. Goals are therefore classified in different levels of abstraction defined as sub-goals which satisfaction attains the main goal.
Obstacles defined in the so called risk-obstacle layer are potential software development risk factors which negatively influence the goals. They can influence multiple goals and are identified through e.g. checklists, questionnaires or brainstorming.
Analyzing risk events caused by the identified risk factors from the risk-obstacle layer. These risk events are characterized with two properties: likelihood and severity. As in the previous layer where one goal can be influenced by more than one risk factor, the same risk factor can pose multiple risk events. Bayesian subjective probability is used to determine the likelihood of the individual risk event. Considering only risk events with negative impact to goals implies that improbable risk events relate to a high goal fulfillment.

The treatment layer contains the risk control action and the performance of them. The aim is the properly attainment of the goals. Treatment actions are connected to risk obstacles as well as to goals as goal contribution respectively obstacle obstruction.

3.3 Languages

The design of the domain model in form of a data model is done by Unified Modeling Language [UML11] which is well known in the context of modeling and therefore widely used. UML defines a set of various modeling concepts as well as their semantics, the interpretation of a computer and how it is readable by humans. The usage of UML is very broad therefore only a subset of modeling techniques and concepts will be applicable in this work.
Using UML as a general purpose modeling language has significant advantages because of the existence of an adequate support infrastructure and therefore the high practical value also due to the economic incentive of implementing tools for support in contrast to highly specialized modeling languages. [Sel07] Another reason for general purpose modeling languages is the availability of well trained programmers and experts which is much more difficult the more complex and specialized to a certain domain the language is. Also the availability of pre-packaged program libraries for general purpose languages is a considerable benefit of the same.
[Sel07, P.2] provides a systematic approach of defining UML profiles in the context of domain specific modeling languages. Therefore they mention the key challenges for designing modeling languages:

- *The need to simultaneously support different levels of precision:*
  Here, UML can be of advance due to its usage in different varieties and its simple syntax and loose semantics.
- *The need to represent multiple different but mutually consistent views of certain elements of the model:*
  According to the complexity of most systems and the variety of different perspectives modeling languages should complementary and mutually be consistent in their problem and implementation oriented approach.
- *The graph-like nature of most modeling languages:*
  Graphical representations of complex systems are better understandable and implementable than linear text-based representation.

The usage of UML is able to face these challenges and in further consideration can be used to model a domain specific language (DSML) which is more prudent than design a brand-new one. In the three methods of defining a DSML identified in [Sel07] UML can be applied in two of them which are: "*Refinement of an existing more general modeling language by specializing some of its general constructs to represent domain-specific concepts.*"[Sel07, P.2] and "*Extension of an existing modeling language by supplementing it with fresh domain-specific constructs.*" in opposition to "*Definition of a new modeling language from scratch."* [Sel07, P.2]

In this work UML is used as object or class diagram as well as activity diagram. The main characteristics of each and the major differences between those two areas of application are reviewed in the following two subsections.

3.3.1 UML object diagram

UML object diagrams or class diagrams are common used for a quantity of applications in data modeling and software engineering. UML object diagrams find their basis in mathematical concept of relations supported by Entity Relationship diagrams. In further development, UML finds usage in Object-Oriented data modeling and supports data querying. [AkBo01]

According to [Bell04] the basics of these kinds of diagrams are to show the types being modeled within the system including types like classes, interfaces, data types and components. Those elements can have attributes and are connected via associations which can in particular also model generalizations/aggregations and inheritances.

Unfortunately according to [AkBo01], a UML class diagram does not sufficiently support all aspects of specification which therefore Object Constraint Languages can be used. They did a further approach in their work.

In this work the concept of UML object diagrams is used to design the conceptual domain model and in a further approach, use this model to implement the database prototype for the model evaluation.

3.3.2 UML activity diagram

An introduction of UML activity diagrams is given in [Bell03] in which the purpose of those diagrams is defined as *"..to model the procedural flow of actions that are part of a larger activity"* [Bell03, P.1] In contrast to UML class diagrams, activity diagrams contains action or activity elements and states as well as flows. They focus on the sequential execution and triggering of actions by conditions. An important element of those diagrams is the communication of information which in further perspective allows i.e. to model business processes and helps managers to get a better understanding of the system and how it works.

UML activity diagrams show parallels to workflow specifications. An usage of UML activity diagrams and adequacy for certain kinds of workflow patterns is examined in [DuHo01].

This work will use UML activity diagrams in a more process oriented way for to consider all the important elements for the design of the conceptual domain model.

3.4 Design methods

This section covers general information about domain modeling as well as design approaches for risk management domain model and database modeling as proof of concept and describes the methodologies used for such an implementation. It shows the relevance and practicability for development methods. The section is therefore separated in design methods for conception of a domain model for risk management and the design methods for the implementation of a database as prototype of this domain model.

3.4.1 Design Methods for Risk Management Domain Model

An insight into the design of a risk management domain model or ontology is provided by [DHMM10]. They proposed and applied a rigorous approach to build an ontology for information system security risk management which can be used furthermore to "..*compare, select or otherwise improve security risk management methods."* [DHMM10, P.289]
They mention the increasing demand of risk management in information systems and the benefits of risk management ontology for aligning a company's business strategy with its information technology strategy. They split their overall research method into four steps:

- Step1-Concept alignment: Which starts by investigating the state of the art in information system security risk management (ISSRM) identifying the core concepts and harmonizing the terminology. As a result of step 1 they obtain a *concept alignment table*, which highlights the core concepts and indicates synonymy and other semantic relationships and a *glossary* of the found terms.
- Step 2-Construction of the ISSRM domain model: Based on the results from step 1 they defined a conceptual model of the ISSRM domain using UML notation (UML class diagram).
- Step 3-Comparison between ISSRM domain model and security-oriented languages: This step consists of the comparison or confrontation of the ISSRM domain model

and prominent security-oriented RE languages including e.g. extended KAOS [vLam04], Abuse Cases [LNIJ04] and Secure-Tropos [MGMP02]. The investigation was about whether and which concept is fully supported, partially supported or missing in the ISSRM domain model.

- Step 4-Definition of ISSRM language support: The last step and the final goal of this work was "*to provide ISSRM-compliant versions of common RE languages.”* [DHMM10, P.292]. They also addressed the formal definition of syntax and semantics including "softer" properties like graphical symbols and structuring mechanisms.

As a result of Step 2-Construction of the ISSRM domain model they designed the ISSRM domain model shown in Figure 9 which is of great importance for this work and explained in more detail later.

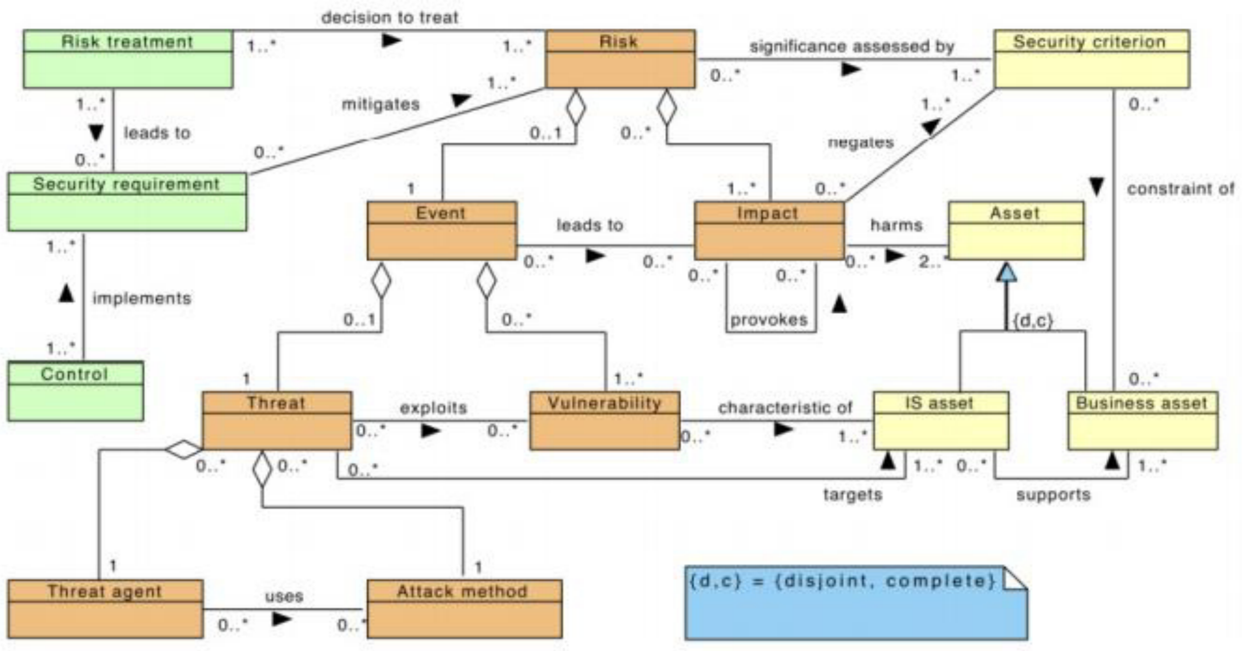


Figure 4 - ISSRM domain model [DHMM10, P.300]

3.4.2 Database Design Methods

Data modeling or database design is a formal process or technique used in software engineering aiming in the analysis of data requirements to the creation of a data model and finally in the implementation of a database [SGWG05].

In [CoBe02] a database system development lifecycle, shown in Figure 10, is described representing the different steps of database design as well as planning, implementation and maintenance. As database design is not the main aim of this work and therefore not every element of the lifecycle is covered, only the important ones are highlighted here.

The preliminary steps of database design are database planning, system definition and requirements collection and analysis. Database planning is basically an important a management activity to realize the database in the most effective way where defining mission statement and mission objectives is taking part. In this work database planning means to consider about the usage of the database and the main aim which is the validation of the conceptual domain model. System definition includes the description of scope and boundaries as well as the main user views. Therefore some different kinds of user views are applied in context of the designed database of this work like described in [Zach87] the Owner's who wants to proof if its requirements are fulfilled, the Designer's view who's main aim is the validation proof of the designed model and the Builder's view who uses the database for further implementations. In [CoBe02] requirements collection and analysis is about information gathering and analyzing regarding requirements which means in this thesis information from GSRM [Isla11] and management activity diagram [Schwa12]. Also identifying requirements for future use of the database is included in this element of database system development lifecycle [CoBe02] which is not considered in this work.

The design process of a database supporting organization's mission statement and objectives is covered in the database design section in [CoBe02].

According to [SGWG05] this process therefore distinguishes three different types of data models are produced. Initially a conceptual data model is created meeting the business requirements and stakeholder's needs. This model is technologically independent and can be used to discuss and adapt according to stakeholder's requirements. By translation of the conceptual data model a logical data model is designed which is a more accurate model representing organization's data and documenting the data structure including all necessary elements and relations. For the organization of the database and the data into tables a further translation of the logical data model into a physical data model is done. In this approach as well as for continuing implementation typically top-down principle is used. In contrast to bottom-up where the model is a result of reengineering and starting with existing data structures, top-down means to implement the database due to data models which therefore are results from analyzing and documenting different business requirements.

DBMS selection is an optional element of the lifecycle which is about to choose the most adequate DMBS for database design. In this work Microsoft SQL Server Management Studio 2014 [MSMS14] is used according to the common usage and open source availability.

Application design [CoBe02] therefore is about to design the user interface and application programs which is not covered in this work.

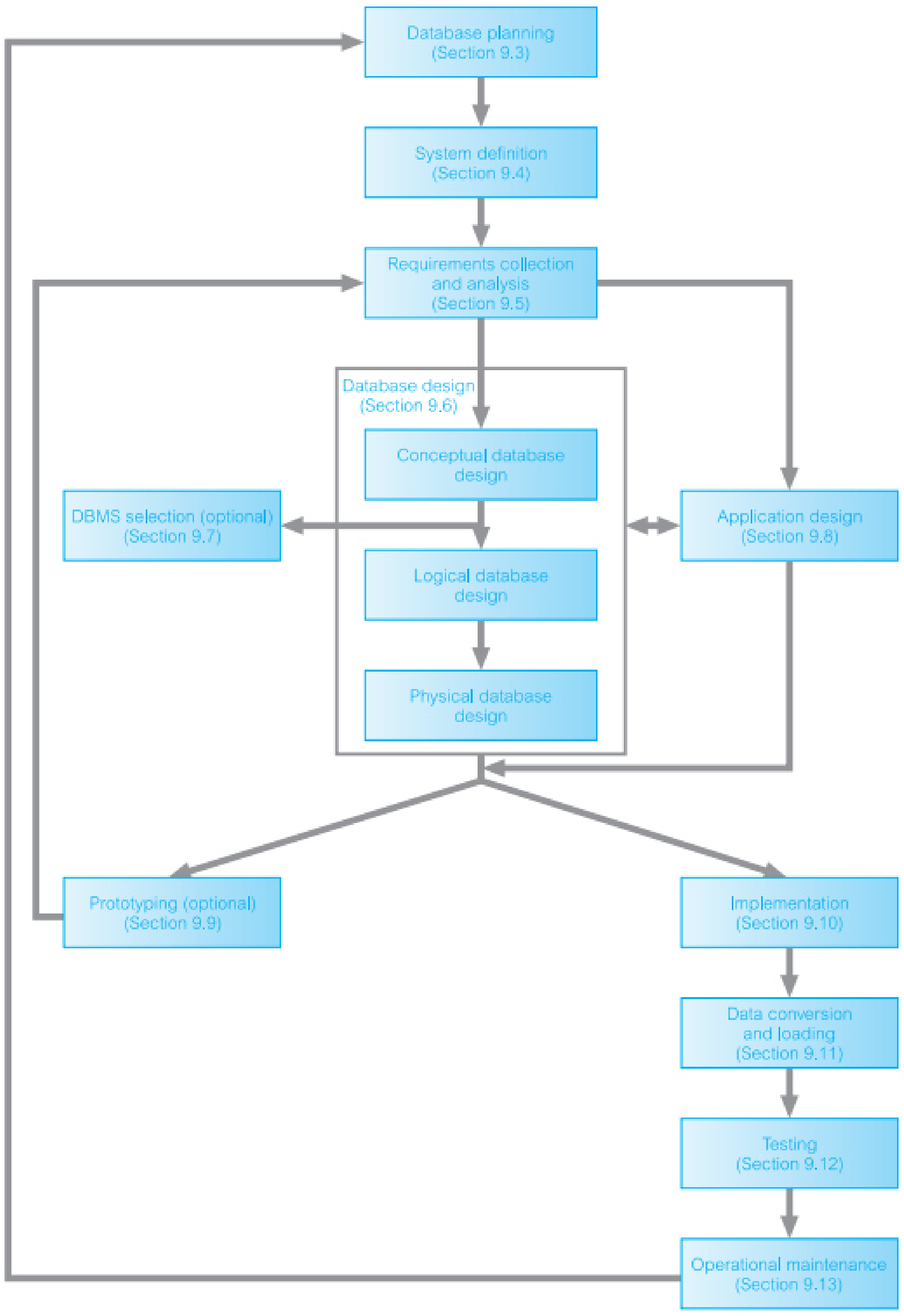

Database planning (Section 9.3)
System definition (Section 9.4)
Requirements collection and analysis (Section 9.5)
Database design (Section 9.6)
Conceptual database design
DBMS selection (optional) (Section 9.7)
Application design (Section 9.8)
Logical database design
Physical database design
Prototyping (optional) (Section 9.9)
Implementation (Section 9.10)
Data conversion and loading (Section 9.11)
Testing (Section 9.12)
Operational maintenance (Section 9.13)

Figure 10- Database system development lifecycle [CoBe02, P. 284]

Database prototyping is important in software development and aims in building a working model and in [CoBe02] is segmented in requirements prototyping in which the prototype is discarded after completing the requirements and evolutionary prototyping which is used for further implementations. Prototyping in general helps to understand system requirements in a more accurate way, leads to more usable software and supports the development and maintenance of such applications [BiGr02]. In [BiGr02] different database prototyping approaches are discussed as well as their further usability in information system development processes. Database prototype is defined as *"any database used to model*

(part of) the data and/or the semantics of another database" [BiGr02, P.448]. The results are two significant database prototypes which are sample databases and test databases. Test databases use synthetic values in the context of prototyping. Those databases often do not support information system development processes as good as sample databases but they are applied when there is no operational database to sample data from. Test databases find utilization in requirements analysis, when it is important to gain better understanding of user requirements, as well as database design, for experimenting different design alternatives and improve understanding in completeness and correctness of data. Another area of usage of test databases is software testing, in detail testing functionality, performance and back-to-back testing. For sample databases prototypes a sample of an original database or consistent sample database is taken which is therefore more valuable for further analysis and can be used for user training, legacy migration (testing, target system development), approximate query evaluation and data mining as well.
The finalizing elements of the database development lifecycle [CoBe02] cover the implementation, data conversion and loading, testing and operational maintenance. These steps represent the parts of the lifecycle which are about the physical realization of the database, the transferring of data from the existing database to the new developed one as well as testing to find errors and monitoring and maintenance. As in this work database prototyping to validate the designed conceptual model is the final aim a sustainable implementation of a database is not covered.

3.5 Analysis methods

For the analysis methods a proof of concept or proof of principle is used showing that the designed and aligned domain model is feasible. The main goal is to prove whether the conceptual model has the potential of being used in the implementation context or not.

As appropriate analysis of the designed conceptual/domain model the implemented prototype in form of a test database sample is used. The validation of a conceptual model means that it is designed in accordance to the domain and represents all the important elements. [STW03] mentions the use of ontologism to proof whether a certain conceptual model is faithful in its representation of the focal domain which should be its main goal. Therefore they list the most relevant attributes a conceptual model has to fulfill to justify to be called valid:

- Accuracy: The accuracy of a domain model is one of the major attributes which incorporates how exactly the semantics of a domain model represents the real world. There are certain approaches to measure the accuracy of a model. [STW03] mentions the incorporation of stakeholder's to proof if the model fits their requirements which is a similar approach to the proof by experts. A more technical approach in the context of UML models can be done with Object Constraint Language (OCL) which is introduced in [WK98]. OCL can support the conceptual modeler in proofing model validation by enriching the model with constraints to query the accuracy of the model.

- Completeness: Completeness is the attribute which predicates that a model includes all the necessary concepts. [STW03] proofs this attribute in association with stakeholder's requirements once more. Another and more computational approach are Natural Language Approaches [Kop12]. In their opinion *"completion is a communication process between end users and designers"* [Kop12, P.33] and furthermore a domain model has to be checked about completeness on an ongoing basis. They list some items which have to be fulfilled to call a domain model complete: Appearance of every relevant classes, attributes and relationships in the model, necessary data types for each attribute, specified multiplicities for each association as well as multi-valued attribute and if necessary definitions for default values of attributes, mandatory or optionally of an attribute as well as unique values and values for certain formats of attributes are defined. Finally there must not be any open task or question related to the classes and attributes in the model. For natural language query analysis therefore linguistic instruments called tagging and chunking are used. *"A tagger is a tool, which takes as input a text and returns a list of sentences with tagged words"* [Kop12, P.38]. Chunking on the other hand is to group words together that can be seen as a phrase.
- Conflict-free: Semantics of a domain model should not contradict one another under the assumption those semantics can be separated in different parts. Concurrency in domain models should be avoided from the very first beginning.

- No redundancy: Similar to conflict-free semantics, semantics of domain models shouldn’t include redundancy as well. Redundancy can therefore arise from conflicts if semantics are subsequently updated. In domain models redundancy refers to situations where different entities own similar or same information and therefore the domain or the context cannot be clearly separated. Automatically this problem is passed on to the database model in the context of implementation and should therefore be eliminated in the first place.

For the validation of conceptual models according to [STW03] *“unfortunately, little is known about how to validate conceptual models effectively and efficiently”* [STW03, P.89].

Nevertheless they introduced an approach for model validation under involvement of stakeholder's at the outset. Unfortunately in this work there is no stakeholder to involve. Therefore the requirements are driven from expertise found in [Isla11] where information is gathered about goal-driven software risk management and the resulting model as well as from [Schw12] where certain information about requirements for the management activity diagram is used. Both are combined to simulate stakeholder's requirement and further validate the model.

[STW03] also mentions three important issues in context of model validation which have to be considered. They are scope, involvement of peoples in the process and methodology. If the conceptual model has to represent a large scope domain there is often the approach to just validate a certain part of the model also because of cost-effective reasons. A second issue is the involvement of different people in the process of model validation i.e. stakeholder's professionals or independent individuals. The third issue to consider is methodology which can have several approaches like reviewing and evaluate the model by participants, questioning participants about the domain being modeled, problem solving using the conceptual model (reflecting use cases or scenarios) and transaction testing where events are used to determine if they are faithfully represented in the conceptual model.

Nevertheless they mention the support of ontologies which are *"theories about the structure and behavior of the real world in general"* [STW03, P.86] for model validation and highlight the importance of choosing the best fitting conceptual modeling grammar for model designing representing specific domain phenomena. As important is that model designers are making sense about of ambiguous semantics of the conceptual model. In the context of ontologies those theories are also derived from the main sources about GSRM [Isla11] and management activity diagram [Schwa12] in this work.

4. Suggested solution/implementation

4.1 Analysis of current goal-driven software project risk management domain model with respect to COSO-II-ERM-Framework

This section covers the analysis of a current GSRM domain model's concepts, relationships and attributes considering the components of the COSO-II-ERM-Framework [COSOII04].

In [Isla11, P.65] a meta-model is designed according to the GSRM framework described in Section 3.2.2 Figure 8. The GSRM meta-model will serve as basic foundation of the suggested solution of this work. Figure 11 shows the meta-model for goal-driven risk management:

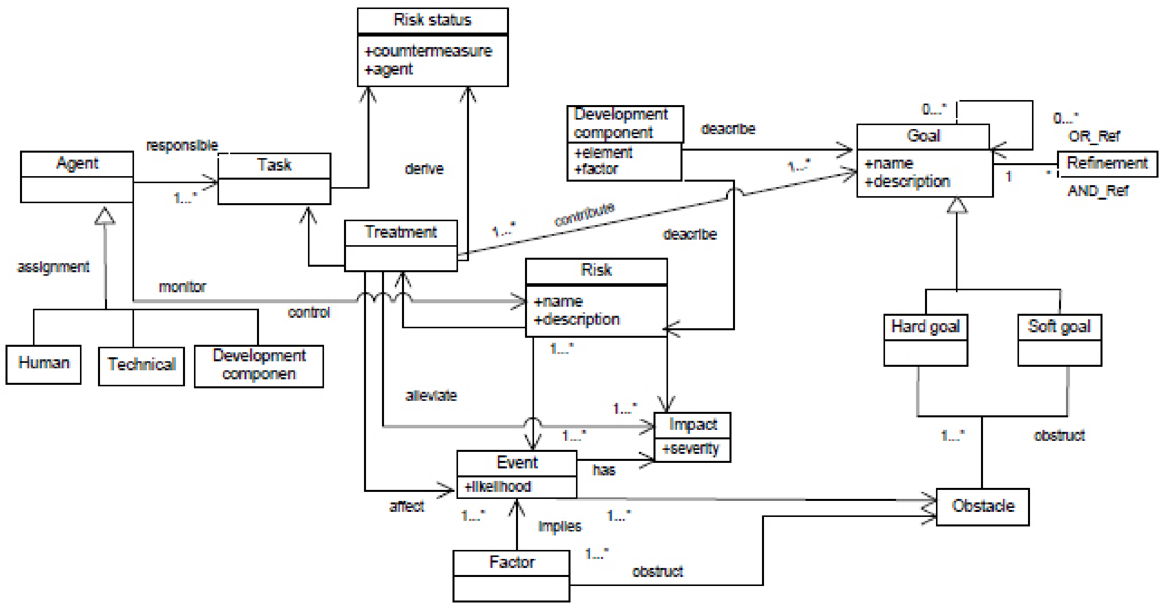


Figure 11- Meta-model for a goal-driven risk management [Isla11, P.66]

The meta-model is designed as UML class diagram and deals as conceptual abstraction including all concepts (i.e. goals, obstacles, risks, agents, etc) and relationships between them (derive, affect, monitor, obstruct, etc). Meta attributes like goal description and event likelihood are also covered. According to [Isla11, P.65-66] goals are declared as the main part similar to the process model. Goals have certain attributes and can be refined by sub-goals and obstructed by obstacles. Development components involve the description of goals as well as obstacles in indirect way as risk description where in the one hand the perspective of desirable properties of the development process is described and on the other hand undesirable circumstances. Risks are therefore defined as software development risks with corresponding attributes. Risk factors in this model represent causes where risk events are similar to consequences which have a negative impact or association to the goal whether it's a single goal or is separated into multiple sub-goals. They define treatment as support of goal satisfaction and risk obstruction and to a certain part controlling and monitoring of the risks which is done by different types of agents.

As meta-models in a conceptual view or class diagrams don't distinguish between information flow and activity flow we can see that in this model actor as well as information and activity are all defined as classes in the same way and with similar character. Actors in this model are represented by different types of agents which can be humans, technical agents or development components. They are responsible for tasks and execute activities like controlling and monitoring risks. Activities on the contrary are figured as concepts or classes and relationships between them as well. An example is the treatment concept which is not clearly separated and defined while on the other hand control and monitor are both depicted as relationships. Classical information elements like risks, goals, obstacles, tasks etc. are represented as concepts as well. Where a mature activity diagram distinguishes between these elements a class diagram in the conventional way does not.

Under consideration of COSO II ERM [COSOII04] and more particular in relation to the COSO ERM Cube some further coherencies can be derived:

According to COSO ERM Framework the first component "*internal environment*" incorporates the basis for all enterprise risk management frameworks and therefore influences the whole process and in this case the conceptual model in a more indirect way than other components. It is about risk philosophy from enterprises broken down into software development and provides structural directives for actions according to this process. A clear definition in form of a concept in the meta-model is missing. A rule or guideline as an output of the planning phase which deals as inputs for further components would be an appropriate integration of this ERM component.

"Objective setting" is about defining objectives in different aspects like strategy, operations, compliance and reporting where in this work the focus lies on operational objectives which are associated with the business process and the related success. It serves as precondition for the event identification component which is described further. In the GSRM meta-model objective setting is embodied in the goal concept as output and in the description of goals in the development component. Risk appetite, in qualitative as well as quantitative aspect, is aligned to objectives and is a major driver for risk tolerance [CSAT04]. Hence there is a need to define the frame of risk tolerance/risk limit in different levels which can be done by integration as input information for further components like risk measurement and risk response activities in the conceptual model.

Using *"Objective setting"* as precondition for the next ERM component *"Event Identification"* is to a certain part well covered in the GSRM process and the corresponding meta-model. Nevertheless COSO defines events not only with negative impact on an objective but also with positive impact in form of opportunities. While the GSRM process also uses the concept of events, concepts namely factors and obstacles are embodied in this ERM component vice versa. What is missing in the meta-model is another kind of event which has positive impact for goal achieving and triggering different actions in the controlling of risks should be used, i.e. a concept called opportunity.

The following ERM component *“risk assessment”* is also pictured in the meta-model in form of the concept of an impact and the event attribute likelihood. According to COSO in this component likelihood and impact are assigned to events and in further activity risk are assessed which is also covered in the GSRM meta-model. Nevertheless it should be stated that an risk respective an opportunity can be a composition of multiple events and impacts as well.

The next component *"risk response"* is according to COSO ERM Framework associated with management activities like risk avoidance, reduction, sharing and acceptance. Those response actions are chosen under consideration of bringing risks within desired risk tolerances. Decisions about risk response actions are made as results of the assessment of the effects on risk likelihood and impact as well as costs and benefits. In the GSRM meta-model this component and the corresponding activities are settled in the treatment concept which is in this case a very extensive concept due to the additional inclusion of controlling and monitoring activities. Therefore a clear distinction should be made about risk response, risk controlling and risk monitoring concepts as well as a guideline for risk measuring should be made.

"Control activities" by definition of COSO incorporates policies and procedures in relation to risk response activities and their execution. But they are also associated with control activities throughout all components and therefore include different kind of activities as well. According to [CSAT04] the control activities component can also serve as risk response in case of reporting objectives. The meta-model incorporates this component at most in the treatment concept but an overall control mechanism is missing. A comprehensive controlling information or policy in form of a rule influencing different concepts should be integrated in the model.

Commonly known, *"information and communication",* which is another COSO component, is needed in all levels of risk management. Information also needs to be identified, captured and communicated. To effectively execute risk management's actions information and the communication thereof is essential. Information can have different shapes in risk management and particular in goal driven software risk management. In the first place attributes of concepts like name, description of goals and respectively of risks but also objectives like goals, their related obstacles and identified events embodies information. In general inputs or outputs which are generated by certain activities represent information. The

meta-model contains information in form of concepts and attributes and the communication in form of relationships as well but to complete the model in a more appropriate way of COSO ERM Framework in context of information, policies for certain kind of concepts should be used i.e. rules for risk measuring, risk response, etc.
"Monitoring", in COSO means an ongoing management procedure, a frequented evaluation of risk management by assessing the presence and functioning of risk management activities. It is important for further analysis and aims in generating reports for top management. The GSRM process, as well as the meta-model, lacks in monitoring concepts which could be integrated as supervisory elements across the entire process and model.

Concluding, the GSRM meta-model does integrate the main components of the COSO ERM Framework Cube although there are some particular elements missing ,mainly the concept of events with positive impact on goal satisfaction in form of opportunities or a planning activity and the corresponding rules for certain parts of the risk management process. Basically it has to be said that it is not the aim of this work to change, adapt or optimize the GSRM process but in more particular to construct a conceptual domain model under adoption of the GSRM meta-model for further implementation of a database supporting managers to perform software risk management in consideration of a goal driven approach.

The conclusion of this analysis is shown in figure 12 in which the relations to the COSO-II-ERM Framework components are highlighted in red. The figure is redesigned on basis of figure 11 which is the GSRM meta model from [Isla11,P.66]. In further sections the model from figure 12 is used to redesign and align the GSRM meta model with respect to the results from the analysis of this section.

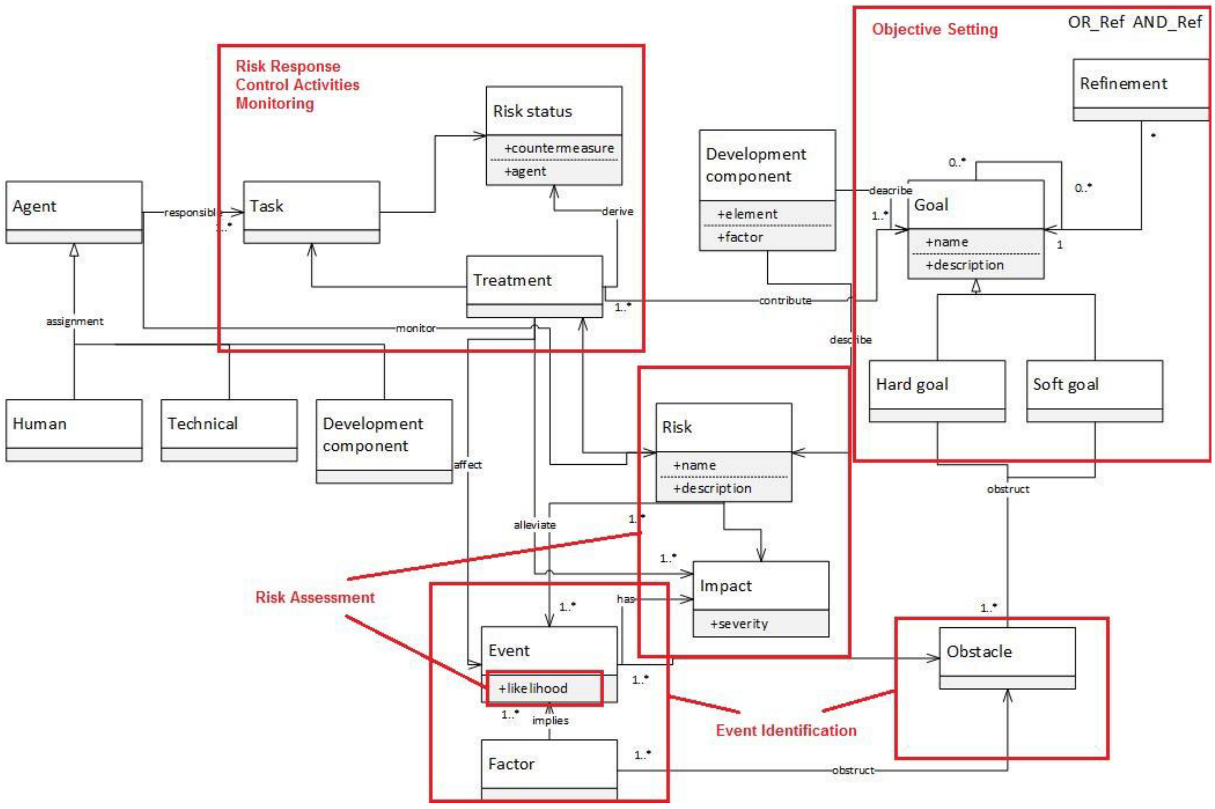


Figure 12- Relations to COSO-II-ERM-Framework and GSRM meta model from [Isla11,P.66]

4.2 Enhancement of the goal-driven software project risk management domain model considering the Generic Cybernetic Management Framework

This subsection addresses the enhancement of the GSRM meta model [Isla11] analyzed in section 4.1. The conclusion of this analysis is incorporated in Figure 12. This section is separated in different topics according to the COSO-II-ERM components [COSOII04] in which every component is dealt as an individual subsection. The enhancement is made according to the results of the previous section as well as to components of the Generic Cybernetic Management Framework [Schwa12] which is shown in Figure 5 as well.

4.2.1 Enhancement related to Internal Environment

Internal Environment must be covered in the planning phase preceding the objective setting. As is it not applicable to include a planning activity in the domain model the outputs of this activity is integrated. Therefore three different plan outputs are defined as rules for certain concepts: plan rules, measure rules and act rules.
Plan rules influence the goal concept considering the comprehensive strategy among other things by prioritization of different goals, define dependencies to other goals, etc.

Measure rules regulate the risk measuring according to the planning phase which includes the rules for measurement of the deviation between the actual risk and the risk limit definition.
Internal Environment incorporates also rules for risk response activities in form of act rules which therefore influence the treatment actions by specifying guidelines if risks exceed a certain tolerance level/risk limit.

4.2.2 Enhancement related to Objective Setting

To keep it simple the distinction between soft or hard goal as well as the corresponding refinement concept is not used any more in the enhanced domain model. The description of a goal by a certain software development component has also been removed aiming in complexity reduction. The essential information therefore is included in the goal concept by adding attributes instead. The risk appetite specification which is a result of objective setting activity is covered in the act rule concept as well as the risk tolerance/risk limit which is realized by a individual concept relating to a particular risk. Additionally the concept of obstacles is removed because of the neutral characteristics of the risk factor concept and incorporation in those as well which will be explained in succeeding sections.

4.2.3 Enhancement related to Event Identification

Risk events according to [Isla11] are implied by risk factors and lead to impacts which are solely of negative nature meaning they obstruct goals by constituting an obstacle. Based on the character of COSO [COSOII04] events can also have positive impact which generates opportunities hence there is the need to integrate events with positive impact as well. This integration is done by specifying the concept of an impact in negative impact and positive impact whereby negative impact is related to risks and positive impact is connected to opportunities. Although the risk factor is still the cause of an event it is not anymore related with exclusively negative aspects.
In contrast to exclusively negative defined risk factors and risk events in [Isla11] this model will use the concept of uncertainty events therefore the option is given to derive positive impacts which are incorporated in opportunities.

4.2.4 Enhancement related to Risk Assessment

In this component risks are assessed by definition of the likelihood of uncertainty events and their impact on the goal reaching process which can be positive and negative and have different kinds of severity. Therefore risks and opportunities can be derived from these concepts. Both are characterized as combinations of uncertainty events and impacts. They are modeled as aggregations of those two concepts where impacts however are realized as interfaces inherited by negative and positive impacts. There exists uncertainty events which are not assessable in detail therefore it should be possible to define the likelihood in a more general way i.e. low, medium, high and specifically in a value of percentage.

4.2.5 Enhancement related to Risk Response

Risk Response is nearly completely covered in the concept of treatment and in order to not differ to much from the underlying GSRM process [Isla11] the concept is not changed in its sense. But to bring some clearance in the concept an enumeration of risk response categories or treatment decisions is integrated listing the itemsavoiding and reducing for risk treatment decisions, exploiting and improving for opportunity treatment as well as transferring and accepting for both concepts.
The model also contains a risk limit concept which deals as output of the objective setting component and characterizes the tolerance level embodying the risk appetite specification for each risk.
The result of this component in form of relations and outputs of the treatment concept are contributing the goal concept exclusively in positive meaning by mitigating risks or seizing opportunities.
Considering Schwaiger's Generic Cybernetic Management Activity Framework [Schwa12] the treatment activities can be of different nature like corrective and adaptive activities which therefore influence different kinds of software project management activities i.e. corrective actions affect the development process itself and adaptive actions have impact on the preliminary goal definition. They are also regulated from a preceding output of the planning phase called act rules.
The former concept of risk status with an attribute for countermeasures is not needed anymore because all the relevant information is covered in the control action concept now.

4.2.6 Enhancement related to Control Activities

Control activities according to [COSOII04] ensure risk response decisions to be carried out in an appropriate way. The activities therefore are chosen depending on the different treatment decision based on the respective risk response decision.
The control actions are also assigned to an agent concept which is responsible for execution of those tasks. The agent concept is therefore carried over from the GSRM meta model [Isla11] because of the meaningfulness of mapping a task to an individual agent.

4.2.7 Enhancement related to Information and Communication

Information and communication is underlying and incorporated in the whole domain model mostly as attributes of different concepts but also as concepts itself and relationships between them. This model also includes rules i.e. goal definition rules, measure rules, and control rules which deal as information concepts.
Information flows according to [CSATII04] are included in every stage or component of the ERM Framework i.e. risk management philosophy and risk appetite (internal environment), objectives/goals, risk tolerance (objective setting), inventory of risks (event identification), assessed risks and opportunities (risk assessment), risk responses (risk response) and control outputs, reports (control activities and monitoring).
The different attributes of each concept are addressed in another subsection.

4.2.8 Enhancement related to Monitoring

Monitoring covers the reporting activities of the comprehensive risk management process and is realized as separate monitoring concept in the domain model. Agents therefore are responsible for those actions like in the control action concept and monitor risks as well as opportunities. The monitoring concept does not only cover reports but documentation as well and evaluates the software development risk management process in its entirety.

4.2.9 Enhanced GSRM Domain Model

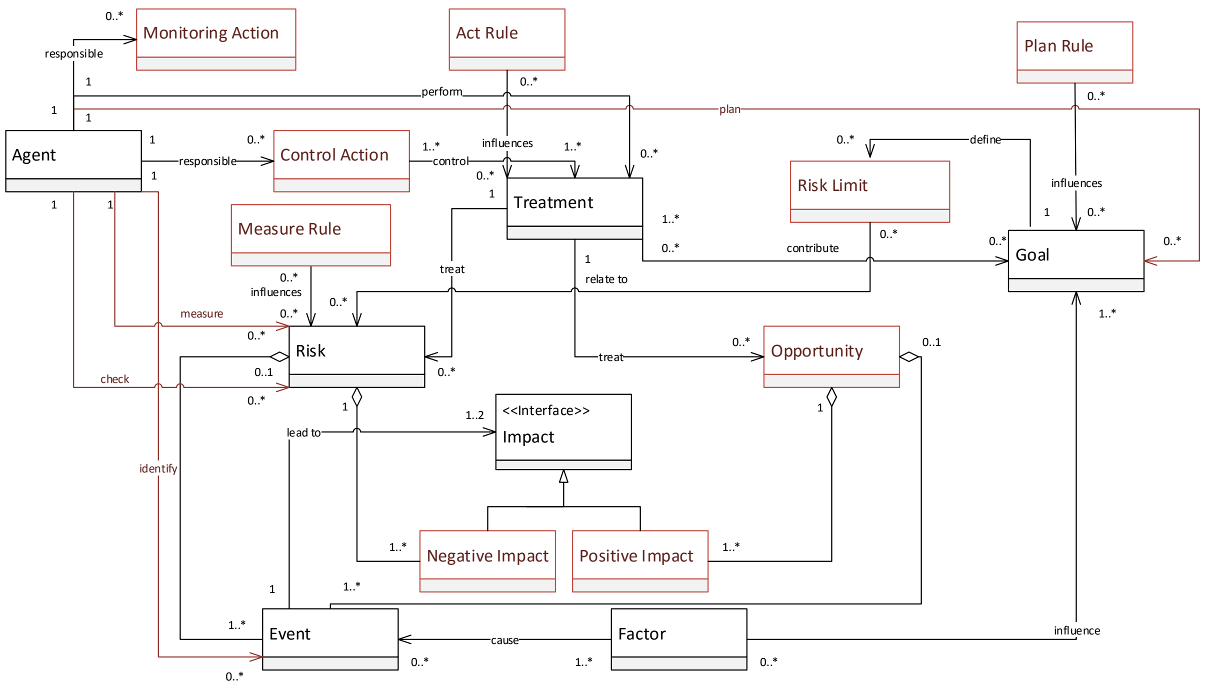


Figure 13- Enhanced GSRM domain model

The model from Figure 13 incorporates all the necessary concepts mentioned in the previous subsections of the enhancement chapter. To get a better overview the attributes of the different concepts are not included here and are discussed in the following. In contrast to the previous existing GSRM meta model from Figure 11 [Isla11] this model is applicable serving as a foundation for database design considering the essential requirements from section 3.5. The GSRM meta model lacked in a clear distinction of associations between the concepts as well as appropriate multiplicities. The focus was set on the differentiation of several kinds of goals and the obstruction by obstacles as on feasibility for database design and further implementations. The GSRM meta model incorporated also a unclear boundary of treatment, control and monitoring actions.

Considering the enhanced domain model some definitions have to be made. The factor concept in this model does now represent the different risk factors according to [Isla11,P.81] including the software development component. The risk factor in combination with the uncertainty event influences the goal in an inherent way. Former risk events are now called uncertainty events which derive from risk factors and are an aggregation of multiple risk factors. An essential change in definitions is the risk-opportunity concept which are both an aggregation of multiple uncertainty events and impacts where risk consist of negative impacts and opportunities of positive ones. The measurement of risks is also a new definition which is incorporated in the risk concept as actual measured risk and is measured in consideration of the measurement rule with an risk limit defined in the planning phase. On this basis the treatment action should be chosen considering an act rule which is also an output of the planning phase and further contributes to the goal as residual risk. Opportunities as they are not meant to derive in the first place of risk factor identification are not measured to a certain limit. Nevertheless the treatment concept does also include activities to handle them too. The agent is now a concept representing an actor who is responsible for several activities which are modeled as relations to other concepts like planning goals, measuring risks, perform treatment activities as well as control and monitor activities.

4.2.10 Domain Model Details

In this subsection the details of the domain model are discussed considering the details of each concept with respect to the related attributes. The concepts are depicted as class of UML [UML11] notation and a short description of each attribute is given in addition.

<u>Goal concept:</u>

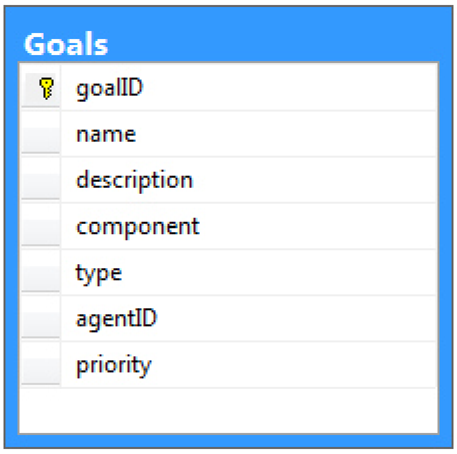


Figure 14- Goal concept

GoalID: A unique ID for each goal.

Name: A string which represents the name of the goal.

Description: An optional additional description of the goal.

Component: A component which relates to the components in [Isla11, p.136] which are Project execution, Process, Product, Human and Environment.

Type: The several goal types related to [Isla11, p.73] which are goals of the type of information, satisfaction, maintain, improve, reduce and product quality factors.

AgentID: The agentID relates to an agent who planned each particular goal in the planning phase.

Priority: Represents the goal priority from 1-10 where 1 is the highest priority and 10 the lowest priority.

<u>RiskFactor concept:</u>

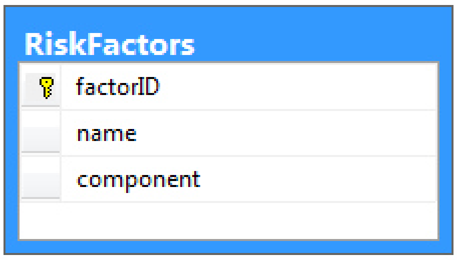


Figure 15- RiskFactor concept

FactorID: A unique ID for each RiskFactor.

Name: A string which represents the name of the RiskFactor.

Component: A component which relates to the components in [Isla11, p.136] which are Project execution, Process, Product, Human and Environment.

Uncertainty Event concept:

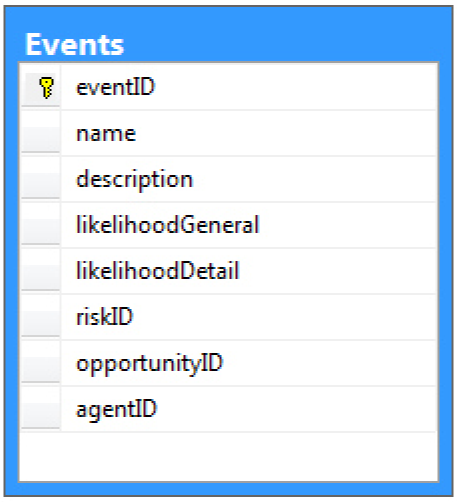


Figure 16- Event concept

EventID: An unique ID for each Event.

Name: A string which represents the name of the Event.

Description: An optional additional description of an event.

LikelihoodGeneral: The general likelihood of an event which is defined with low, medium or high. This likelihood is also used for uncertainty events which likelihood attribute cannot defined in percentage.

LikelihoodDetail: The detailed likelihood is optional and ranges between 0-100% where 0% means the event is not likely to occur and 100% means the event is certain to occur.

RiskID: The riskID relates to a risk from which the event is part of.

OpportunityID: the opportunityID relates to an opportunity from which the event is part of.

AgentID: The agentID relates to the agent who identified the particular uncertainty event including the likelihood and impact assessment.

Impact concept:

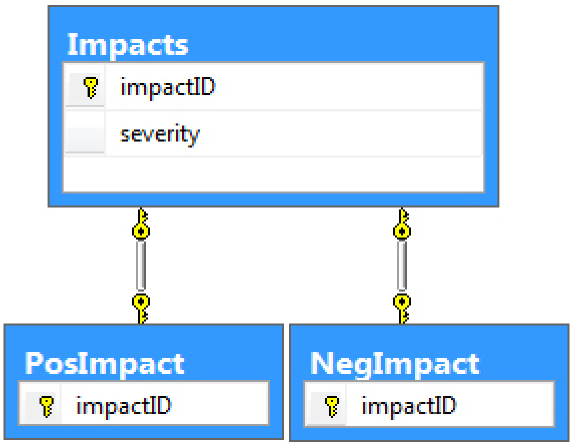


Figure 17- Impact concept

ImpactID: An unique ID for each impact which relates in a 1:1 notation to the causing event.

Severity: The severity of an impact which is related to [Isla11, p.78] which can be low, medium or high.

Each kind of impact (either negative or positive) is related to the corresponding uncertainty event by the impactID.

Risk concept:

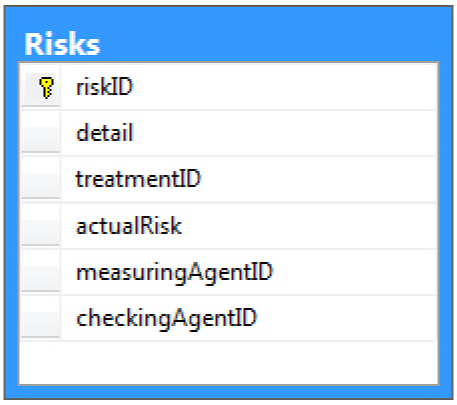


Figure 18- Risk concept

RiskID: An unique ID for each risk.

Detail: A non-optional detail description for each risk.

TreatmentID: The treatmentID relates to the corresponding treatment for each particular risk.

ActualRisk: Represents the actual risk as output of the measurement activity. It can either be a value of percentages or amounts or be of informational manner like grades.

MeasuringAgentID: The agentID relates to the agent who is responsible for measuring a particular risks.

CheckingAgentID: The agentID relates to the agent who is responsible for checking a particular risk.

RiskLimit concept:

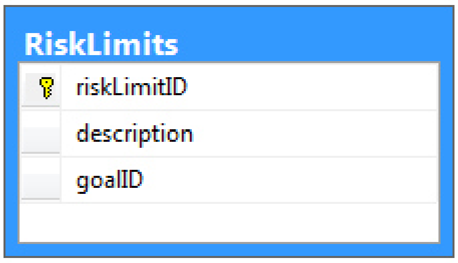


Figure 19- RiskLimit concept

RiskLimitID: An unique ID for each risk limit.

Description: A non-optional description for each risk limit.

GoalID: The goalID relates to the goal from which the risk limit has been driven.

Opportunity concept:

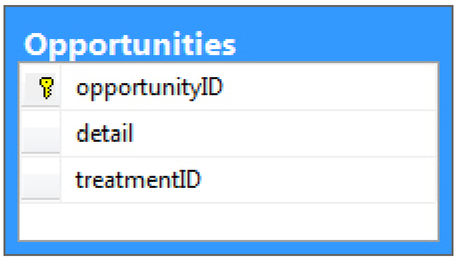


Figure 20- Opportunity concept

OpportunityID: An unique ID for each opportunity.

Detail: A non-optional detail description for each opportunity.

TreatmentID: The treatmentID relates to the corresponding treatment for each particular opportunity.

Treatment concept:

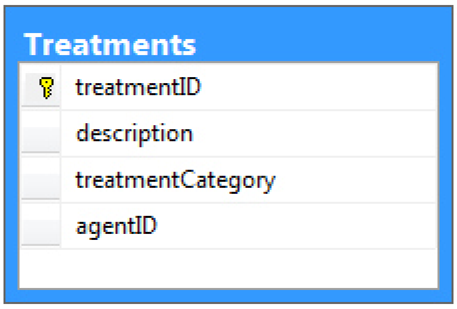


Figure 21- Treatment concept

TreatmentID: An unique ID for each treatment.

Description: A non-optional description for each treatment.

TreatmentCategory: The treatment category describes the individual type of each treatment whether is it accepting, transferring, risk avoiding, risk reducing, opportunity exploiting or opportunity improving.

AgentID: The agentID relates to the agent who is responsible for the risk response action.

ControlAction concept:

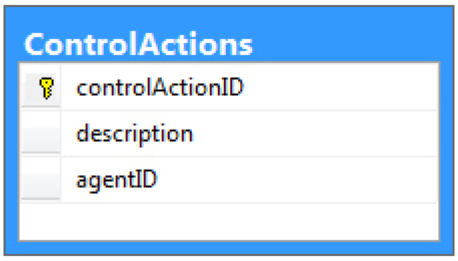


Figure 22- ControlAction concept

ControlActionID: An unique ID for each control action.

Description: A non-optional description for each control action.

AgentID: The agentID relates to the agent who is responsible for the control action.

MonitoringAction concept:

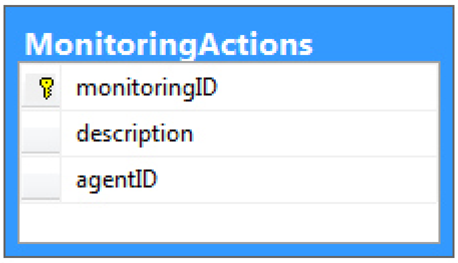


Figure 23- MonitoringAction concept

MonitoringActionID: An unique ID for each control action.

Description: A non-optional description for each monitoring action.

AgentID: The agentID relates to the agent who is responsible for the monitoring action.

Agent concept:

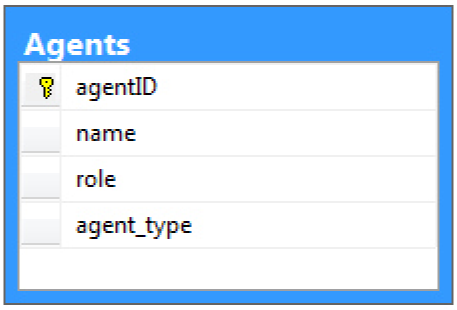


Figure 24- Agent concept

AgentID: An unique ID for each agent.

Name: A string representing the name of an agent.

Role: The role of a particular agent.

AgentType: Representing the agent type relating to [Isla11, p.88] which distinguishes between human agent and technical agent. The component agent is included in the technical agent as well.

ActRule concept:

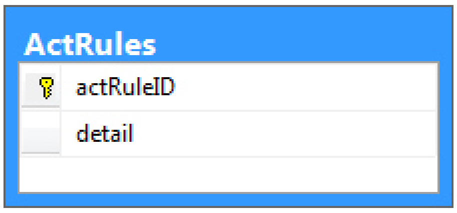


Figure 25- ActRule concept

ActRuleID: An unique ID for each act rule.

Detail: A non-optional detailed description for each act rule.

MeasuringRule concept:

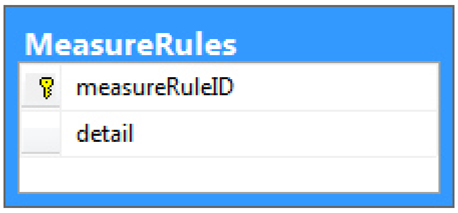


Figure 26- MeasruingRule concept

MeasuringRuleID: An unique ID for each measuring rule.

Detail: A non-optional detailed description for each measuring rule.

PlanRule concept:

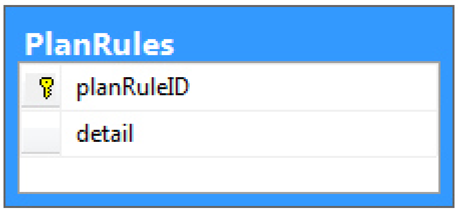


Figure 27- PlanRule concept

PlanRuleID: An unique ID for each plan rule.

Detail: A non-optional detailed description for each plan rule.

4.3 Database Modeling

For database design Microsoft SQL Server 2014 Management Studio [SSMS14] has been used. The database has been designed considering UML class diagram notation [UML11].
The aim of this database is to proof the validity of the designed domain model from the previous section. Therefore a test sample of data is used to make applicable analysis. This section is separated in different subsection representing important subparts of the database to gain a better overview of the database as a whole.

4.3.1 Goal-PlanRule-RiskFactor concept

The goal-planRule-riskFactor concept is separated in two subconcepts.
This goal-riskFactor concept shows the relation between goals and risk factors incorporated in a junction table where goalID and factorID are used as keys which means there can be multiple goals related to multiple risk factors representing a classical many to many relationship. The testsample is used considering the top ten risk factors table from the results of case study 1 [Isla11, p.110]. Where the goals are mainly used from this table representing sub goals as well as main goals, the risk factors are altered in a more neutral way to be able to derive events with a positive impact as well. Figure25 shows the concept in database diagram notation of SQL Management Studio 2014 [MSMS14].
For the IDs of each table unique identifier are taken which are generated automatically.
There are also some restrictions included in the concept, i.e. a check-constraint which ensures that only priority values between 1 and 10 can be chosen.

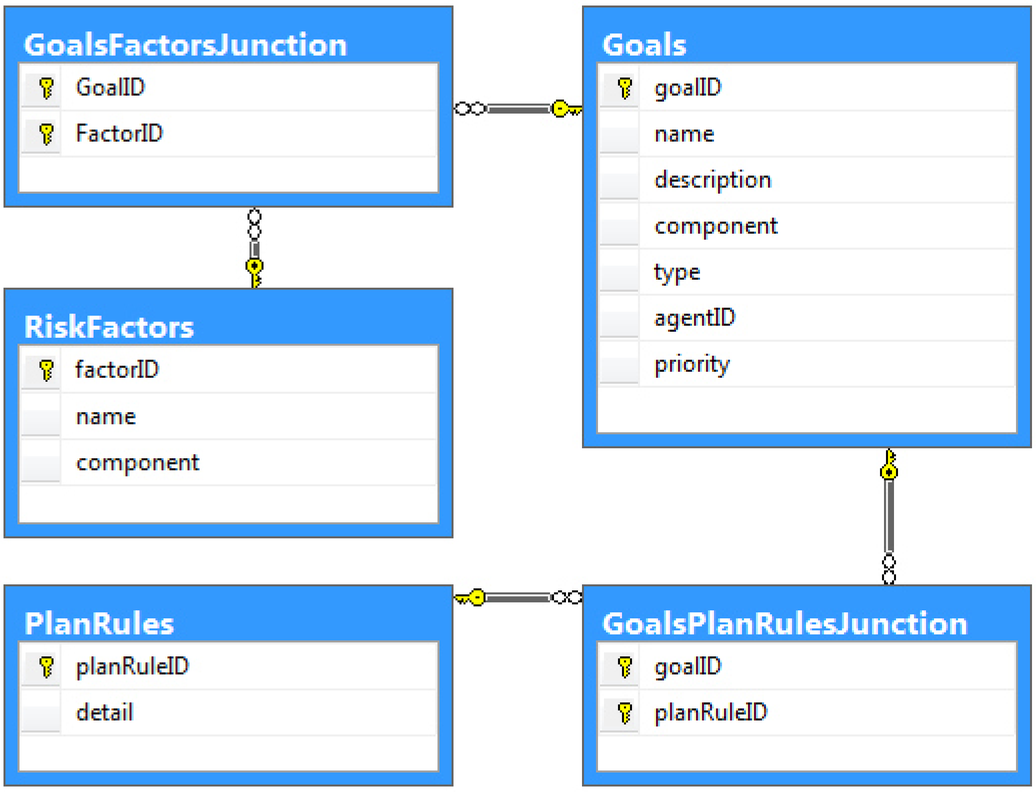
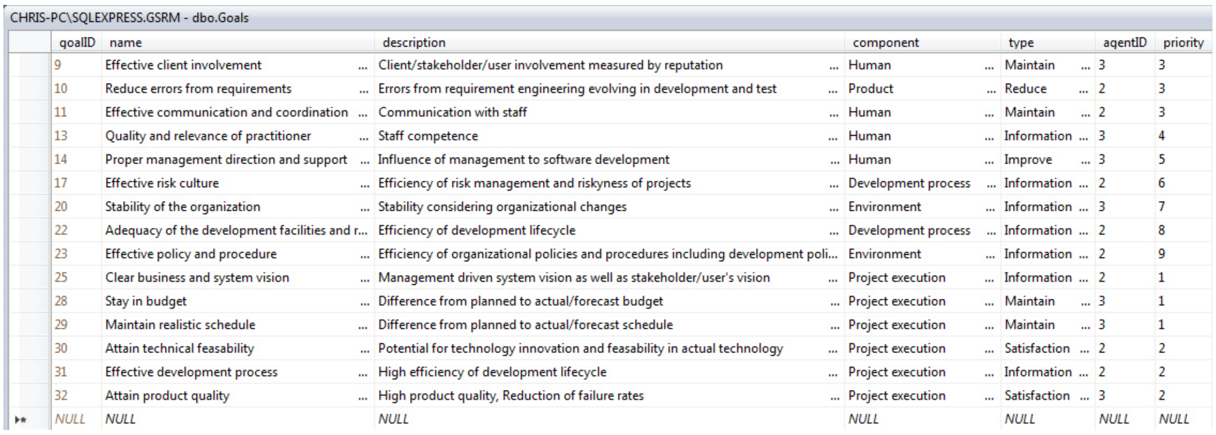


Figure 28- Goal-PlanRules-RiskFactor concept

Figure29 shows the goal table filled with the described test sample.



CHRIS-PC\SQLEXPRESS.GSRM - dbo.Goals

	goalID	name	description	component	type	agentID	priority
	9	Effective client involvement ...	Client/stakeholder/user involvement measured by reputation ...	Human ...	Maintain ...	3	3
	10	Reduce errors from requirements ...	Errors from requirement engineering evolving in development and test ...	Product ...	Reduce ...	2	3
	11	Effective communication and coordination ...	Communication with staff ...	Human ...	Maintain ...	2	3
	13	Quality and relevance of practitioner ...	Staff competence ...	Human ...	Information ...	3	4
	14	Proper management direction and support ...	Influence of management to software development ...	Human ...	Improve ...	3	5
	17	Effective risk culture ...	Efficiency of risk management and riskyness of projects ...	Development process ...	Information ...	2	6
	20	Stability of the organization ...	Stability considering organizational changes ...	Environment ...	Information ...	3	7
	22	Adequacy of the development facilities and r...	Efficiency of development lifecycle ...	Development process ...	Information ...	2	8
	23	Effective policy and procedure ...	Efficiency of organizational policies and procedures including development poli...	Environment ...	Information ...	2	9
	25	Clear business and system vision ...	Management driven system vision as well as stakeholder/user's vision ...	Project execution ...	Information ...	2	1
	28	Stay in budget ...	Difference from planned to actual/forecast budget ...	Project execution ...	Maintain ...	3	1
	29	Maintain realistic schedule ...	Difference from planned to actual/forecast schedule ...	Project execution ...	Maintain ...	3	1
	30	Attain technical feasability ...	Potential for technology innovation and feasability in actual technology ...	Project execution ...	Satisfaction ...	2	2
	31	Effective development process ...	High efficiency of development lifecycle ...	Project execution ...	Information ...	2	2
	32	Attain product quality ...	High product quality, Reduction of failure rates ...	Project execution ...	Satisfaction ...	3	2
▸*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 29- Goal Table

The goal-actRule concept is based on the same principle. A junction table is used incorporating the goalID and planRuleID. Similar to goals and riskFactors, goals and planRules are related in a many-to-many connection as well.

4.3.2 Event-Impact-Risk-Opportunity concept

The following Figure30 shows the event -impact-risk-opportunity concept. The event, which was already mentioned, is designed as an uncertainty event not exclusively a risk event. Deriving from a risk factor which is connected through an EventRiskFactorJunction, similar to the previous described GoalRiskFactorJunction, events are associated in a many to many relation as well. An event is related to an impact in a one to one-two relation which is a special case and means that an event can be associated to at least one and at most two impacts therefore one negative and one positive impact, i.e. high project complexity which can have negative impact in case of risks to overrun budget or positive impact to increase staffs availability for complexity handling. The test sample for events are to a certain part, which represents the negative events, taken from Table 6.4 "Identified Risk Factors and Events" in [Isla11, p.117] but similar to them complemented with potential positive events.

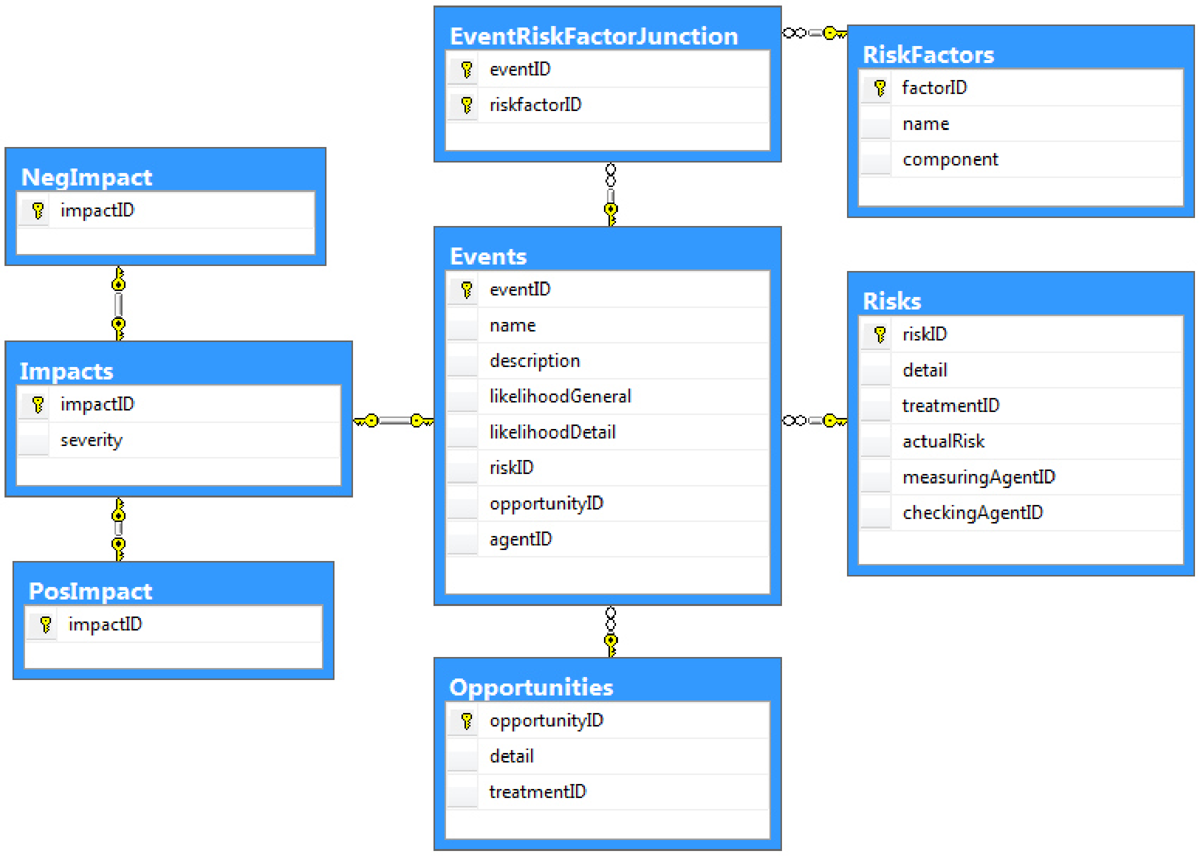


Figure 30- Event-Impact-Risk-Opportunity concept

Positive and negative impacts are related one-to-one to impacts therefore they are identified with the same ID. The concepts also include some check constraints like severity (low, medium, high), likelihoodGeneral (low, medium, high) and likelihoodDetail (only values between 0 and 100 are allowed representing procentual values). Risks and opportunities are connected via one-to-many relations to events representing aggregations therefore each event incorporates the riskID respectively the opportunityID.

4.3.3 Risk-MeasureRule-RiskLimit-ActRule-Opportunity-Treatment concept

The following figure represents the Risk-MeasureRule-RiskLimit-ActRule-Opportunity-Treatment concept. Risks and opportunities are connected via a one-to-many relationship where the ID of each treatment is included in the related risks respectively opportunity.Each treatment contributes to a goal in a many-to-many relationship therefore another junction table is needed with goalID and treatmentID.

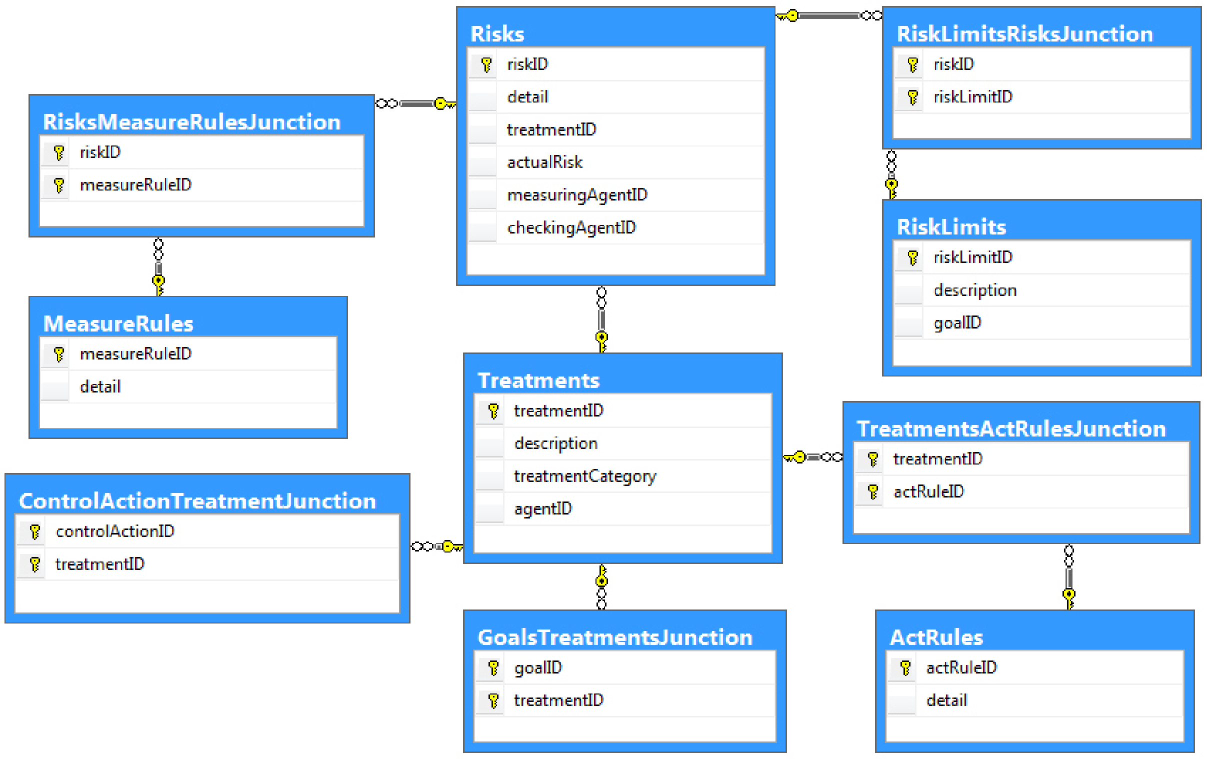


Figure 31- Risk-RiskLimit-Opportunity-Treatment concept

Treatment categories are restricted via a check constraint allowing only the defined values from section 4.2.10. Each treatment is also controlled by a control action also via a many-to-many relation. An agent is responsible for a certain treatment therefore a one-to-many association is used similar to risks and agents relationships which embodies the measurement and checking activity. Risks are connected via a RiskMeasureRulesJunction to measure Rules as well as treatments and actRules. The measurement is dependent on the risk limit driven from a particular goal implemented via a one-to-many relationship. Because there exist multiple risk limits for a particular risk and vice versa these connection is realized via a many-to-many association incorporated in a RiskLimitRisksJunction table.

4.3.4 Agent responsibility concept

Figure32 shows the concept of agents and their responsibility. As already mentioned in the domain model design particular agents are responsible for certain tasks. An agent can be responsible for defining rules which are incorporated in particular concepts like measurement rule, plan rule or act rule.

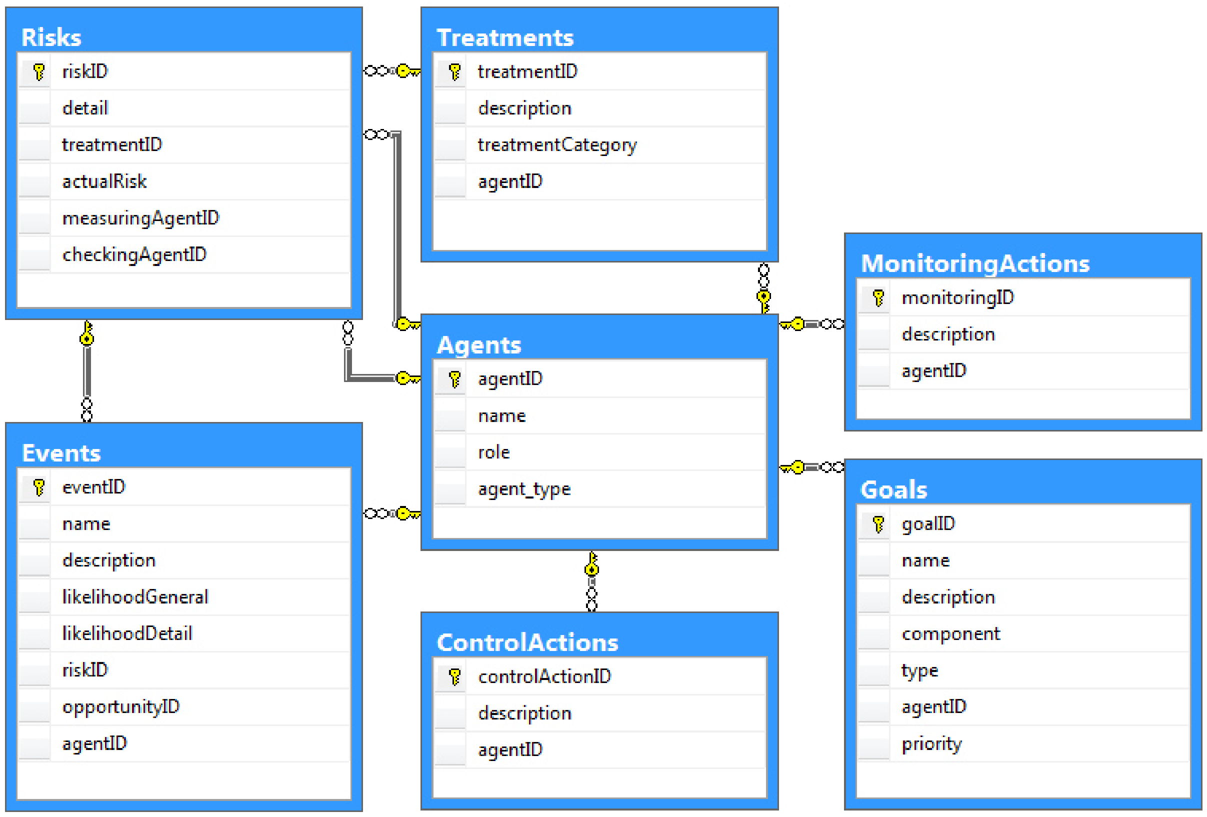


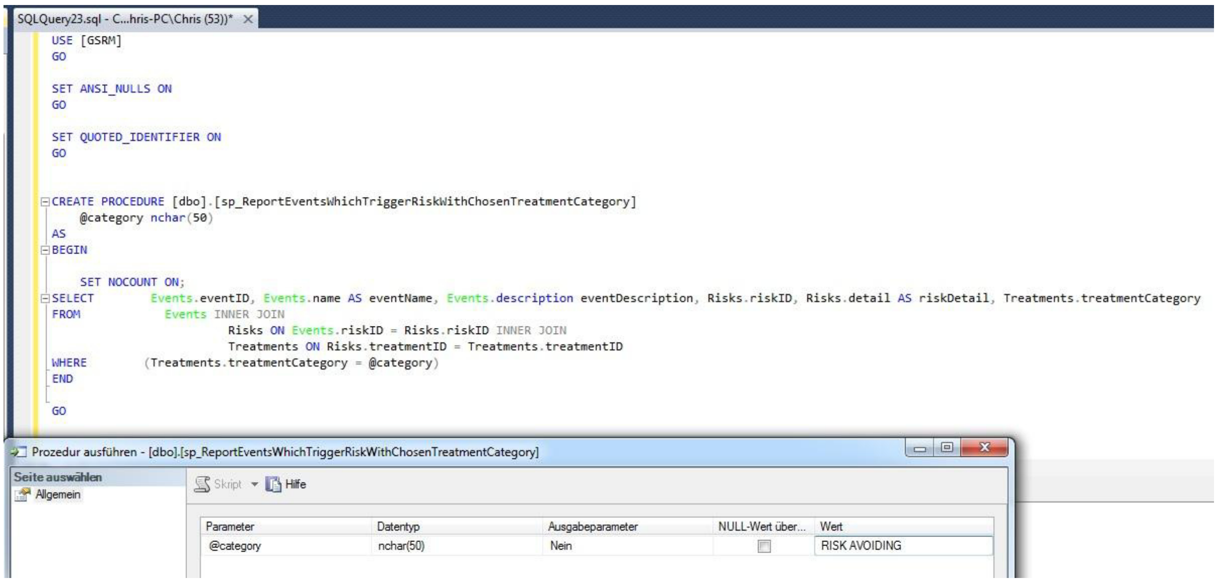
Figure 32- Agent responsibility concept

Agents are responsible for identifying goals and events, measuring and checking risks, performing treatments as well as control and monitoring actions. Each of these relationships is implemented via a one-to-many relationship embodied as unique agentID in each of these concepts. Nevertheless there is no restriction that an agent which is responsible for a treatment also has to be responsible for the control action controlling this treatment analog to risk checking and measurement. It is most likely that control and monitoring actions are performed via technical agents i.e. automated tools.

4.4 Validation proof

The validation proof should show that the domain model satisfies all the validation requirements from section 3.5. More precise, the designed database is used for certain analysis representing reports for risk managers, simple and more complex queries as well as insert queries. They should contribute not only to a better understanding of the domain model but much more to serve risk managers in case of applications and proof the domain model of its validity. Therefore Microsoft SQL Server Management Studio [MSMS14] is used and in more detail stored procedures are created to ensure a repeatedly execution applicable by different programming languages.

The first example shows a simple report of all events which are part of risks which again are treated by a treatment of a chosen category. Figure33 illustrates this stored procedure in programming code as simple SQL query consisting of select, from and where clause.


SQLQuery23.sql - C...hris-PC\Chris (53))*

```sql
USE [GSRM]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


CREATE PROCEDURE [dbo].[sp_ReportEventsWhichTriggerRiskWithChosenTreatmentCategory]
    @category nchar(50)
AS
BEGIN

    SET NOCOUNT ON;
SELECT        Events.eventID, Events.name AS eventName, Events.description eventDescription, Risks.riskID, Risks.detail AS riskDetail, Treatments.treatmentCategory
FROM            Events INNER JOIN
                         Risks ON Events.riskID = Risks.riskID INNER JOIN
                         Treatments ON Risks.treatmentID = Treatments.treatmentID
WHERE        (Treatments.treatmentCategory = @category)
END

GO
```

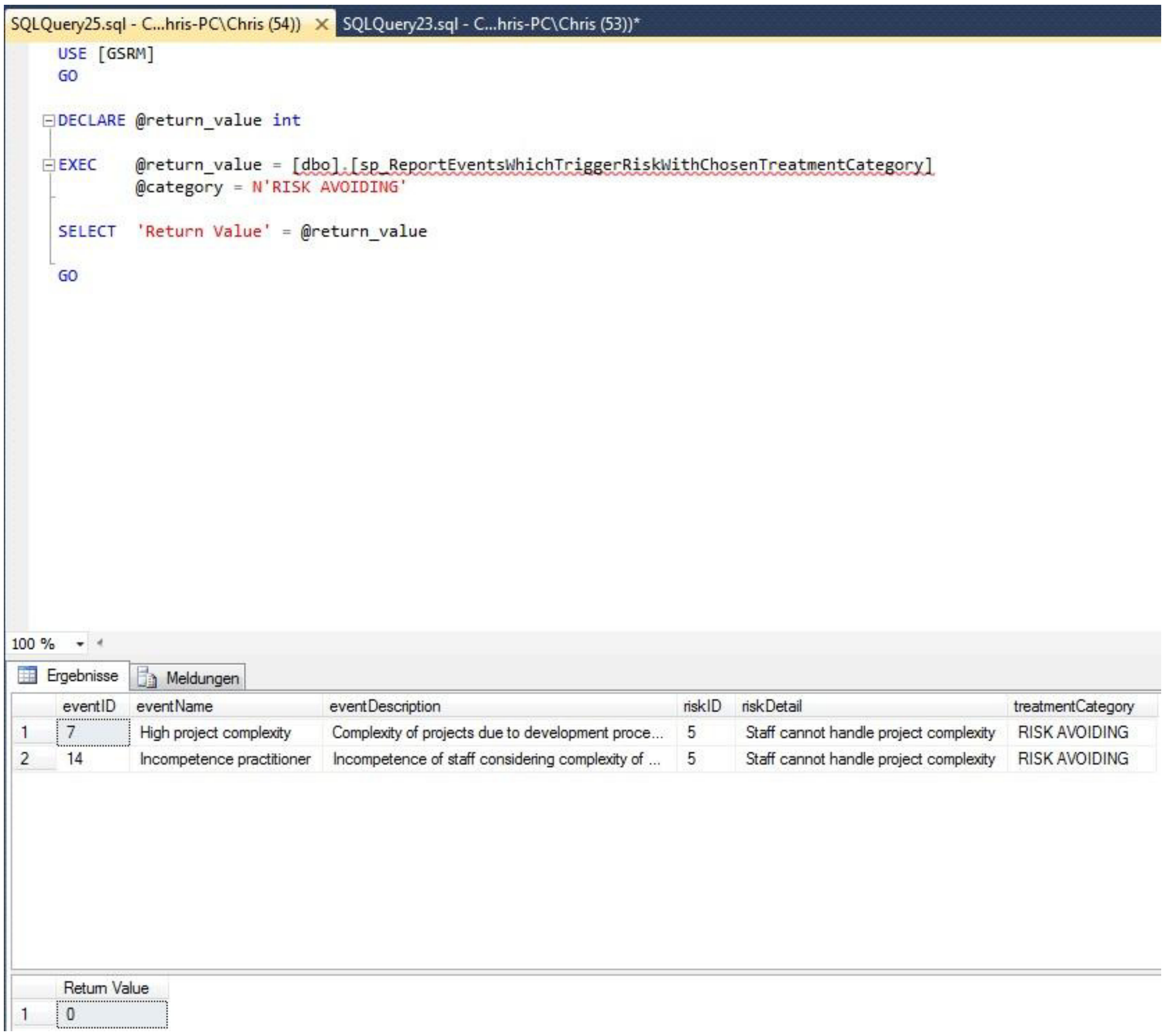
Prozedur ausführen - [dbo].[sp_ReportEventsWhichTriggerRiskWithChosenTreatmentCategory]

Parameter	Datentyp	Ausgabeparameter	NULL-Wert über...	Wert
@category	nchar(50)	Nein	☐	RISK AVOIDING

Figure 33- Stored Procedure ReportEvent Coding

The stored procedure as well as the query can be executed on its own and results in an output of the database sample. The “select” clause defines which attributes of which concept or table should be considered in the output or report. Important attributes therefore are eventID, name of each event, description of each event, ID and detail of each risk and the chosen treatmentCategory. The concept tables which are relevant for the report are listed and connected in the “from” clause which are events, risks and treatments. The connection between them are coded via join commands. The “where” clause embodies the filter which in this case only samples are chosen and included in the output which treatment category equals the input of the user.

The following Figure34 shows the output of the complete database sample which fits these requirements. In this case the output consists of two different events of which each has a unique identifier. The return value is zero, stating for an error free query respectively stored procedure. Therefore the stored procedure proofs the validation of the domain model considering the addressed conceptions and tables of the database.



SQLQuery25.sql - C...hris-PC\Chris (54)) SQLQuery23.sql - C...hris-PC\Chris (53))*

```
USE [GSRM]
GO

DECLARE @return_value int

EXEC    @return_value = [dbo].[sp_ReportEventsWhichTriggerRiskWithChosenTreatmentCategory]
        @category = N'RISK AVOIDING'

SELECT  'Return Value' = @return_value

GO
```

100 %

Ergebnisse | Meldungen

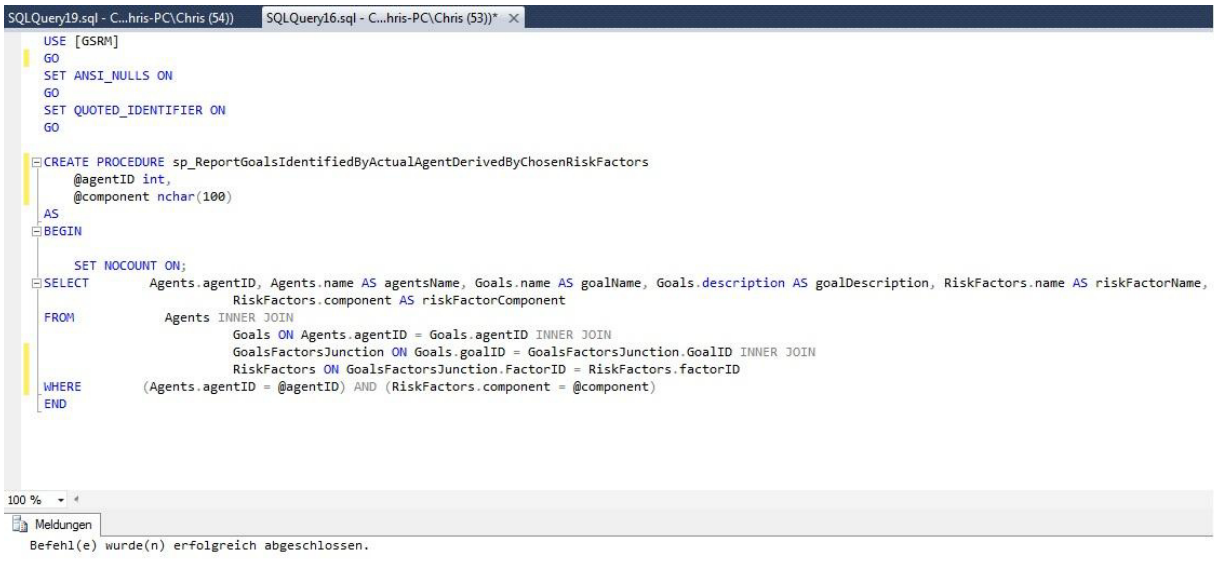
	eventID	eventName	eventDescription	riskID	riskDetail	treatmentCategory
1	7	High project complexity	Complexity of projects due to development proce...	5	Staff cannot handle project complexity	RISK AVOIDING
2	14	Incompetence practitioner	Incompetence of staff considering complexity of ...	5	Staff cannot handle project complexity	RISK AVOIDING

	Return Value
1	0

Figure 34- Stored Procedure ReportEvent Execution

Another procedure proofs the validation of agents, goals and risk factors concepts. Figure35 shows the coding of the ReportGoalsIdentifiedByActualAgentDerivedByChosenRiskFactors procedure.

The procedure outputs the goals influenced by a chosen risk factor type/component and identified by a particular agent which is in this case an agent identified by the a chosen agentID. This would be automatically be chosen in software implementations when a user who is logged in has a specific ID which is transported in the procedure in the background of the application.

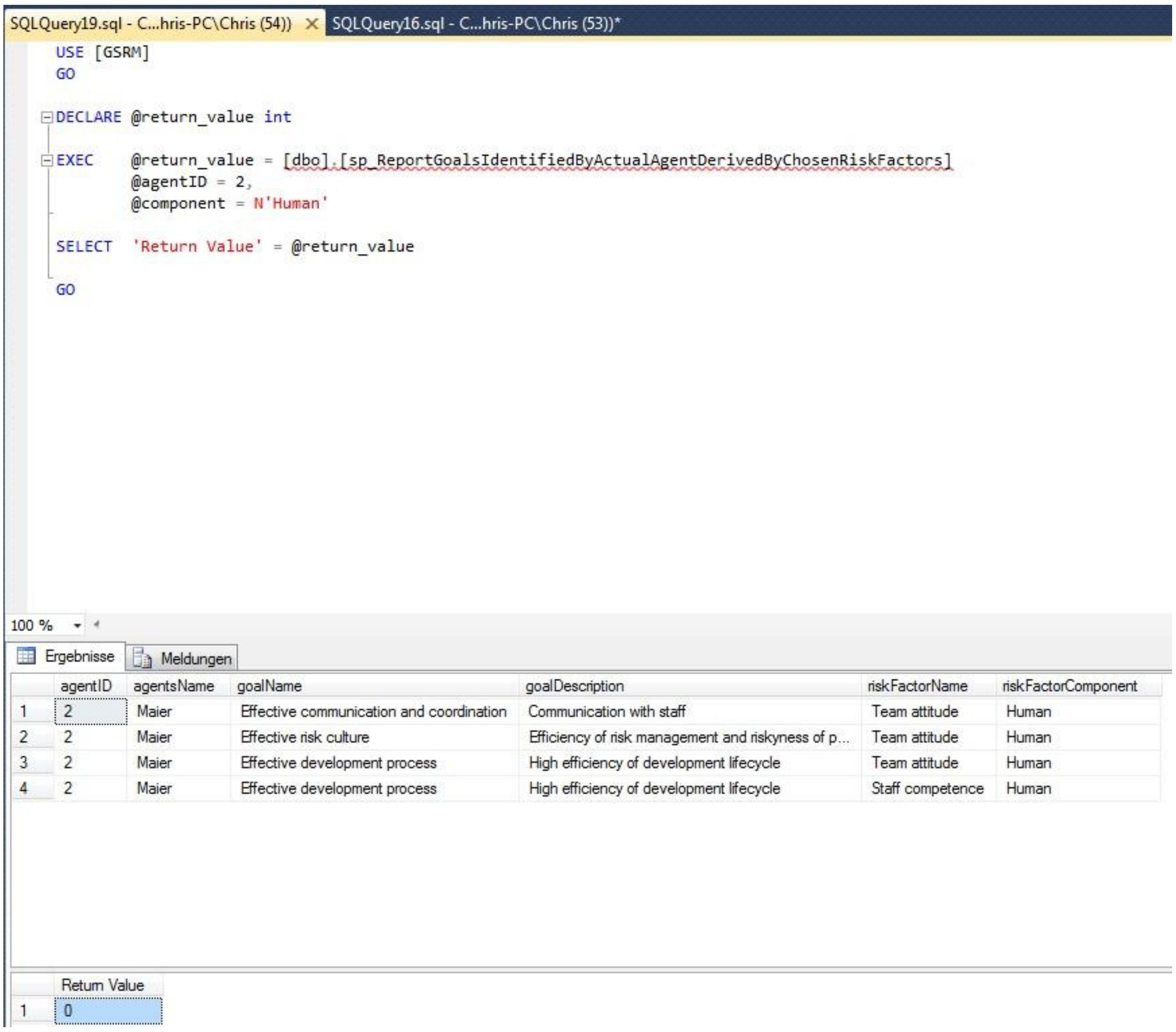

SQLQuery19.sql - C...hris-PC\Chris (54))
SQLQuery16.sql - C...hris-PC\Chris (53))*
100 %
Meldungen

```sql
USE [GSRM]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE sp_ReportGoalsIdentifiedByActualAgentDerivedByChosenRiskFactors
    @agentID int,
    @component nchar(100)
AS
BEGIN

    SET NOCOUNT ON;
SELECT        Agents.agentID, Agents.name AS agentsName, Goals.name AS goalName, Goals.description AS goalDescription, RiskFactors.name AS riskFactorName,
                         RiskFactors.component AS riskFactorComponent
FROM            Agents INNER JOIN
                         Goals ON Agents.agentID = Goals.agentID INNER JOIN
                         GoalsFactorsJunction ON Goals.goalID = GoalsFactorsJunction.GoalID INNER JOIN
                         RiskFactors ON GoalsFactorsJunction.FactorID = RiskFactors.factorID
WHERE        (Agents.agentID = @agentID) AND (RiskFactors.component = @component)
END
```

```
Befehl(e) wurde(n) erfolgreich abgeschlossen.
```

Figure 35- Stored Procedure ReportGoal Coding

The "select"-clause embodies attributes like agentID, name of each agent, name of each goal, description of each goal as well as name and component of each risk factor. Tables and concepts which are necessary for this procedure are incorporate in the "form"-clause like agents, goals and risk factors. Because goals and risk factors are associated with an junction table this table is also included in the "form-clause via a joined command. The "where"-clause therefore embodies the filter which is in this case the agentID having the value of the chosen agentID defined by the user at the procedure's execution and the risk factor component which equals similar to the agentID a user defined risk factor component.


SQLQuery19.sql - C...hris-PC\Chris (54))
SQLQuery16.sql - C...hris-PC\Chris (53))*
100 %
Ergebnisse
Meldungen

```
USE [GSRM]
GO

DECLARE @return_value int

EXEC    @return_value = [dbo].[sp_ReportGoalsIdentifiedByActualAgentDerivedByChosenRiskFactors]
        @agentID = 2,
        @component = N'Human'

SELECT  'Return Value' = @return_value

GO
```

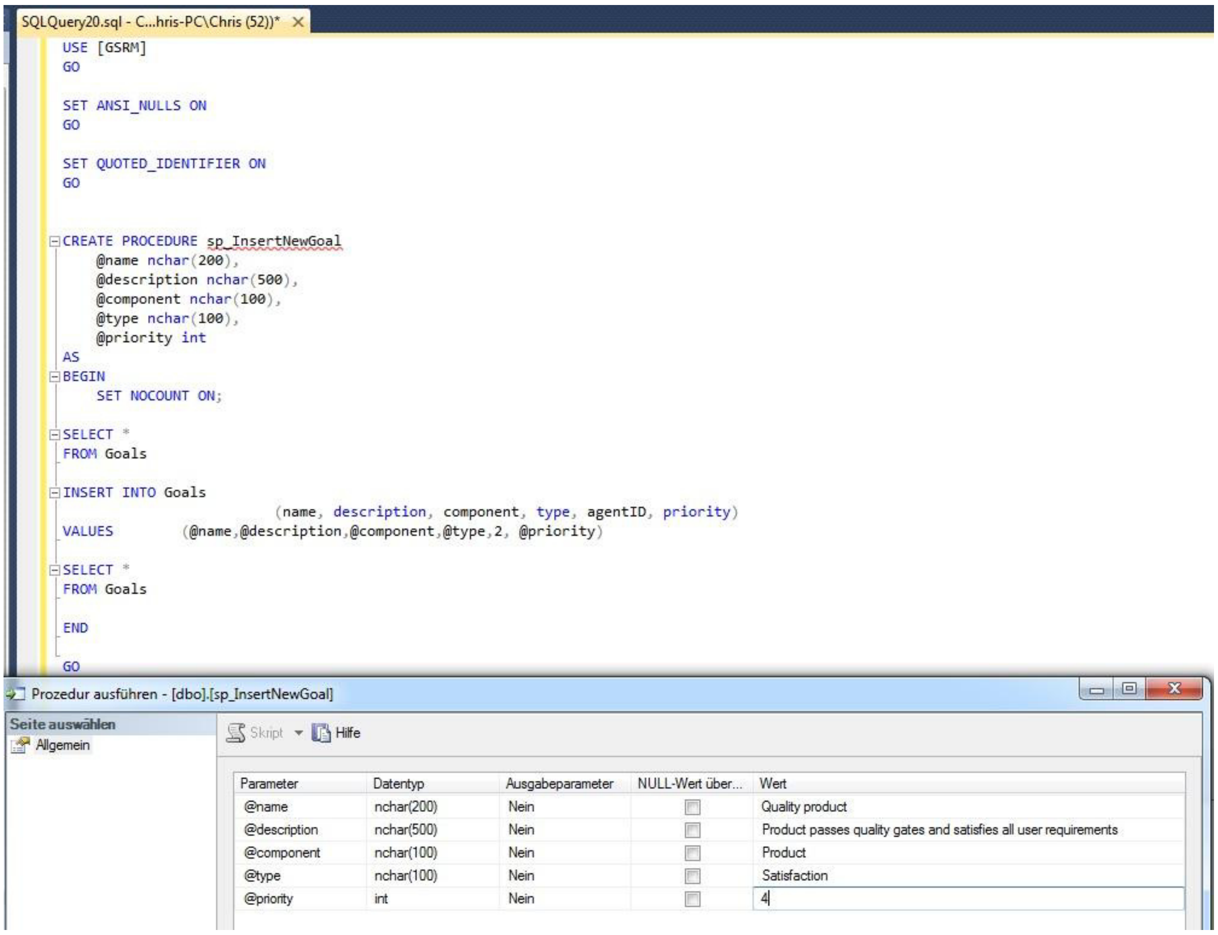
	agentID	agentsName	goalName	goalDescription	riskFactorName	riskFactorComponent
1	2	Maier	Effective communication and coordination	Communication with staff	Team attitude	Human
2	2	Maier	Effective risk culture	Efficiency of risk management and riskyness of p...	Team attitude	Human
3	2	Maier	Effective development process	High efficiency of development lifecycle	Team attitude	Human
4	2	Maier	Effective development process	High efficiency of development lifecycle	Staff competence	Human

	Return Value
1	0

Figure 36- Stored Procedure ReportGoal Execution

Figure36 represents the output after execution of the procedure. By execution the user defines the value of the agentID with "2" and the riskFactorComponent with "Human". The output shows that the data sample exists of four goals which are derived by human risk factors and identified of the specific agent with the name "Maier" and ID "2".

The next procedure incorporates an insert query. The coding and insert inputs are displayed in Figure37. The coding embodies the insert attributes including their types and default values if necessary i.e. agentID is valued with "2" which can be automatically defined by software applications i.e. based on userID transitioned in the background of the application.


SQLQuery20.sql - C...hris-PC\Chris (52))*
Prozedur ausführen - [dbo].[sp_InsertNewGoal]
Seite auswählen
Allgemein
Skript
Hilfe

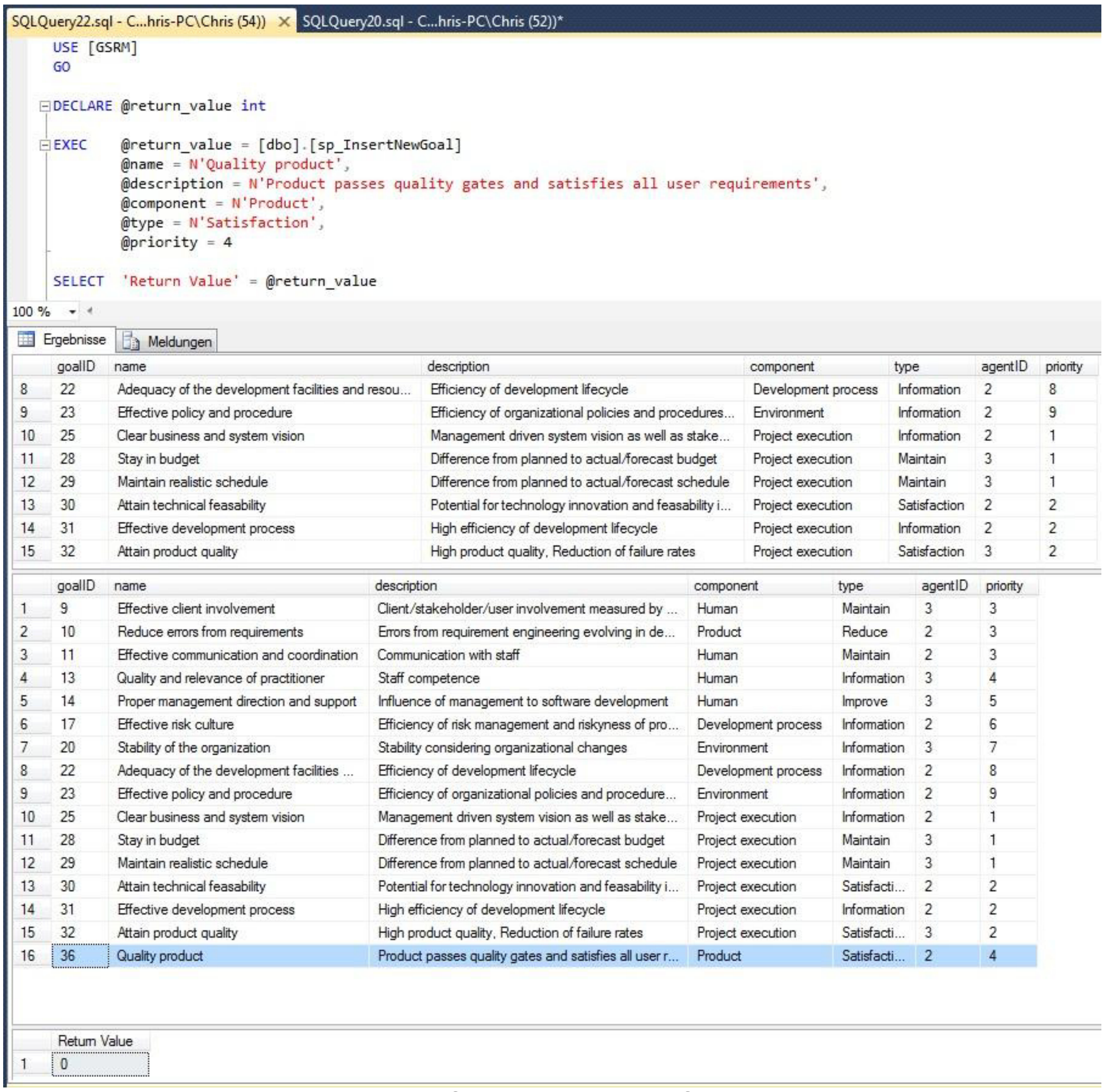
```
USE [GSRM]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


CREATE PROCEDURE sp_InsertNewGoal
    @name nchar(200),
    @description nchar(500),
    @component nchar(100),
    @type nchar(100),
    @priority int
AS
BEGIN
    SET NOCOUNT ON;

SELECT *
FROM Goals

INSERT INTO Goals
                        (name, description, component, type, agentID, priority)
VALUES         (@name,@description,@component,@type,2, @priority)

SELECT *
FROM Goals

END

GO
```

Parameter	Datentyp	Ausgabeparameter	NULL-Wert über...	Wert
@name	nchar(200)	Nein	☐	Quality product
@description	nchar(500)	Nein	☐	Product passes quality gates and satisfies all user requirements
@component	nchar(100)	Nein	☐	Product
@type	nchar(100)	Nein	☐	Satisfaction
@priority	int	Nein	☐	4

Figure 37- Stored Procedure InsertGoal Coding

The procedure also includes values for each attribute which has to be defined by the user i.e. the risk manager. In this case a new goal is defined called "quality product" with goal-specific attributes like name, description, component, type and priority.

The following Figure38 shows the execution of this insert procedure and further the result of this execution which can be retrieved by a simple query outputting all the goals in the data base. Therefore the first list states for the goals before insertion and the second list for afterwards. The new defined goal has been inserted via the stored procedure from Figure37 with respect to the attribute values chosen from the user. The goalID is defined as an identifier in the database therefore this attribute is automatically designed and an incremental value. The new defined goal is listed as last because it is sorted by ascending goalIDs on a default basis. Therefore the inserting of goals states for another valid concept of the domain model.



SQLQuery22.sql - C...hris-PC\Chris (54)) SQLQuery20.sql - C...hris-PC\Chris (52))*

```
USE [GSRM]
GO

DECLARE @return_value int

EXEC    @return_value = [dbo].[sp_InsertNewGoal]
        @name = N'Quality product',
        @description = N'Product passes quality gates and satisfies all user requirements',
        @component = N'Product',
        @type = N'Satisfaction',
        @priority = 4

SELECT  'Return Value' = @return_value
```

	goalID	name	description	component	type	agentID	priority
8	22	Adequacy of the development facilities and resou...	Efficiency of development lifecycle	Development process	Information	2	8
9	23	Effective policy and procedure	Efficiency of organizational policies and procedures...	Environment	Information	2	9
10	25	Clear business and system vision	Management driven system vision as well as stake...	Project execution	Information	2	1
11	28	Stay in budget	Difference from planned to actual/forecast budget	Project execution	Maintain	3	1
12	29	Maintain realistic schedule	Difference from planned to actual/forecast schedule	Project execution	Maintain	3	1
13	30	Attain technical feasability	Potential for technology innovation and feasability i...	Project execution	Satisfaction	2	2
14	31	Effective development process	High efficiency of development lifecycle	Project execution	Information	2	2
15	32	Attain product quality	High product quality, Reduction of failure rates	Project execution	Satisfaction	3	2

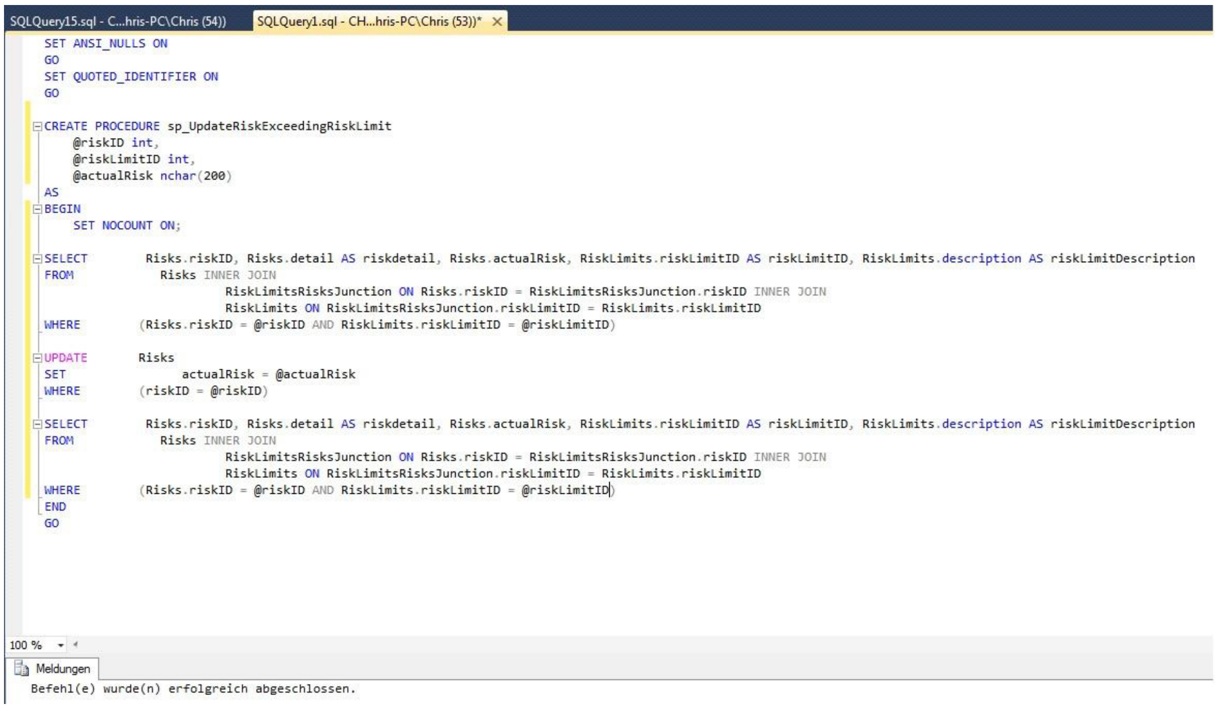
	goalID	name	description	component	type	agentID	priority
1	9	Effective client involvement	Client/stakeholder/user involvement measured by ...	Human	Maintain	3	3
2	10	Reduce errors from requirements	Errors from requirement engineering evolving in de...	Product	Reduce	2	3
3	11	Effective communication and coordination	Communication with staff	Human	Maintain	2	3
4	13	Quality and relevance of practitioner	Staff competence	Human	Information	3	4
5	14	Proper management direction and support	Influence of management to software development	Human	Improve	3	5
6	17	Effective risk culture	Efficiency of risk management and riskyness of pro...	Development process	Information	2	6
7	20	Stability of the organization	Stability considering organizational changes	Environment	Information	3	7
8	22	Adequacy of the development facilities ...	Efficiency of development lifecycle	Development process	Information	2	8
9	23	Effective policy and procedure	Efficiency of organizational policies and procedure...	Environment	Information	2	9
10	25	Clear business and system vision	Management driven system vision as well as stake...	Project execution	Information	2	1
11	28	Stay in budget	Difference from planned to actual/forecast budget	Project execution	Maintain	3	1
12	29	Maintain realistic schedule	Difference from planned to actual/forecast schedule	Project execution	Maintain	3	1
13	30	Attain technical feasability	Potential for technology innovation and feasability i...	Project execution	Satisfacti...	2	2
14	31	Effective development process	High efficiency of development lifecycle	Project execution	Information	2	2
15	32	Attain product quality	High product quality, Reduction of failure rates	Project execution	Satisfacti...	3	2
16	36	Quality product	Product passes quality gates and satisfies all user r...	Product	Satisfacti...	2	4

	Return Value
1	0

Figure 38- Stored Procedure InsertGoal Execution

Another stored procedure represents the update of a particular risk after exceeding the related risk limit. In this case it has to be assumed that the risk manager gets a notification about the exceeding of the risk i.e. via automated specifically checking tools as part of a comprehensive software application.

Nevertheless in this case it is necessary to define the riskID and the riskLimitID related to the exceeding. Afterwards the actual risk has to be remeasured and redefined in the database. Figure 39 shows the coding of such an update procedure including the coding of two lists which embodies the important risk and riskLimit attributes corresponding this exceeding before and after the update query.


SQLQuery15.sql - C...hris-PC\Chris (54))
SQLQuery1.sql - CH...hris-PC\Chris (53))*
100 %
Meldungen

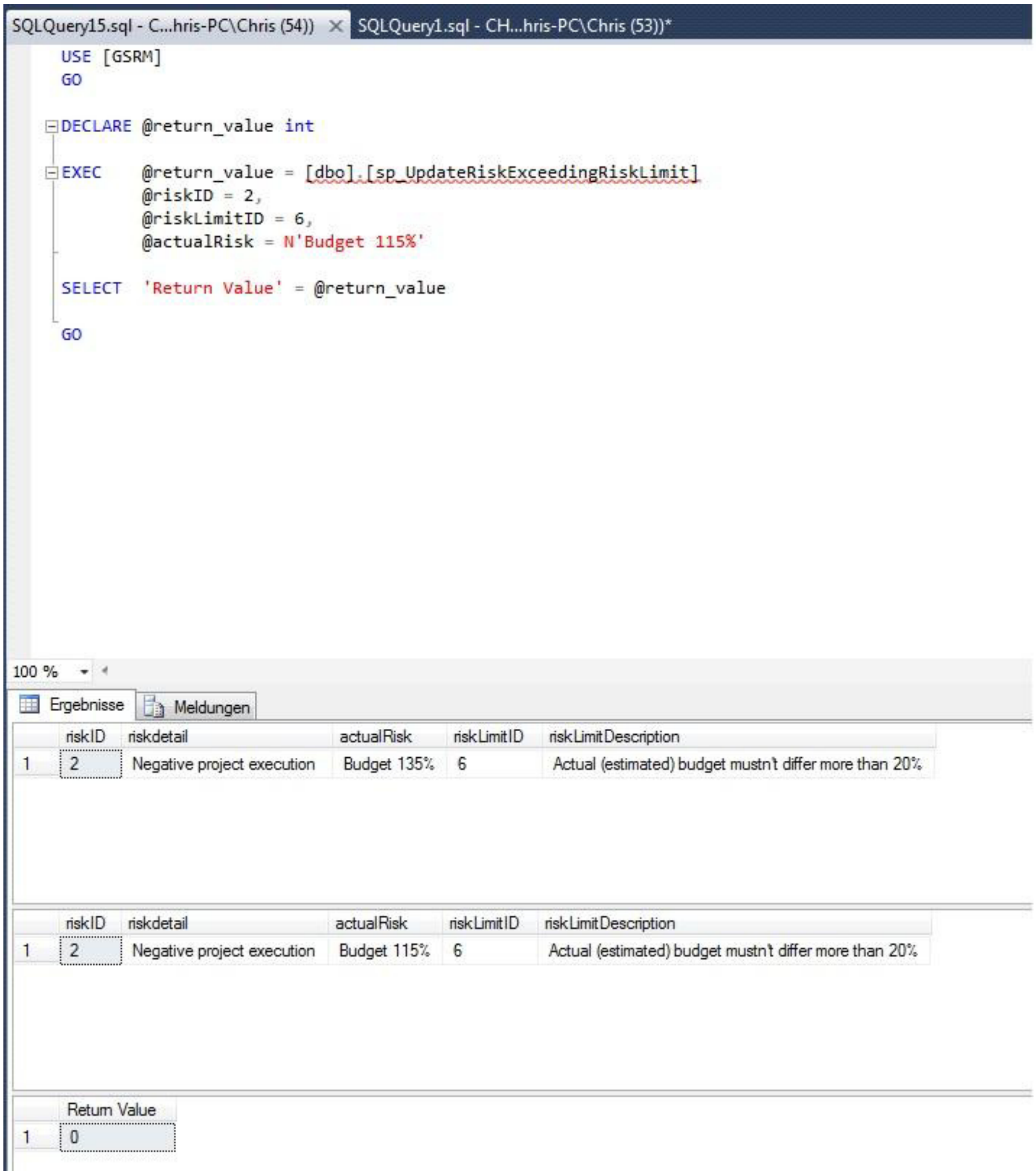
```sql
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE sp_UpdateRiskExceedingRiskLimit
    @riskID int,
    @riskLimitID int,
    @actualRisk nchar(200)
AS
BEGIN
    SET NOCOUNT ON;

SELECT        Risks.riskID, Risks.detail AS riskdetail, Risks.actualRisk, RiskLimits.riskLimitID AS riskLimitID, RiskLimits.description AS riskLimitDescription
FROM            Risks INNER JOIN
                         RiskLimitsRisksJunction ON Risks.riskID = RiskLimitsRisksJunction.riskID INNER JOIN
                         RiskLimits ON RiskLimitsRisksJunction.riskLimitID = RiskLimits.riskLimitID
WHERE        (Risks.riskID = @riskID AND RiskLimits.riskLimitID = @riskLimitID)

UPDATE       Risks
SET                actualRisk = @actualRisk
WHERE        (riskID = @riskID)

SELECT        Risks.riskID, Risks.detail AS riskdetail, Risks.actualRisk, RiskLimits.riskLimitID AS riskLimitID, RiskLimits.description AS riskLimitDescription
FROM            Risks INNER JOIN
                         RiskLimitsRisksJunction ON Risks.riskID = RiskLimitsRisksJunction.riskID INNER JOIN
                         RiskLimits ON RiskLimitsRisksJunction.riskLimitID = RiskLimits.riskLimitID
WHERE        (Risks.riskID = @riskID AND RiskLimits.riskLimitID = @riskLimitID)
END
GO
```

```
Befehl(e) wurde(n) erfolgreich abgeschlossen.
```

Figure 39- Stored Procedure Update Risk coding

The update query only updates a specifically defined risk in the risks table by defining a new value for the attribute actualRisk.

The following Figure40 shows represents the execution of this procedure and shows the update of the defined risk as well as the output of the queries before and after the update. At the beginning the actualRisk of the risk "Negative project execution" exceeds the related risk limit of allowed 20% difference to the estimated budget, whereas afterwards the actualRisk fulfills this criteria. The profound business processes and workflows are not addressed here. Tasks like risk treatment and risk re-measurement have to be done beforehand.



SQLQuery15.sql - C...hris-PC\Chris (54)) SQLQuery1.sql - CH...hris-PC\Chris (53))*

```
USE [GSRM]
GO

DECLARE @return_value int

EXEC    @return_value = [dbo].[sp_UpdateRiskExceedingRiskLimit]
        @riskID = 2,
        @riskLimitID = 6,
        @actualRisk = N'Budget 115%'

SELECT  'Return Value' = @return_value

GO
```

100 %

Ergebnisse | Meldungen

	riskID	riskdetail	actualRisk	riskLimitID	riskLimitDescription
1	2	Negative project execution	Budget 135%	6	Actual (estimated) budget mustn't differ more than 20%

	riskID	riskdetail	actualRisk	riskLimitID	riskLimitDescription
1	2	Negative project execution	Budget 115%	6	Actual (estimated) budget mustn't differ more than 20%

	Return Value
1	0

Figure 40- Stored Procedure UpdateRisk Execution

The previous stored procedures are essential for a validation proof of the domain model. Nevertheless the focus of this validation lies on the proof of the domain model not specifically on the validation of the database which in most cases implies this fact though. These four scenarios represent a sample of a more comprehensive field of application of the domain model and the implemented database. Their aim is to contribute to the support of project managers tasks corresponding to risk management in software project.

According to section 3.5 of this work and in more detail mentioned in [STW03] the domain model has to incorporate certain characteristics and fulfill criteria like accuracy, completeness, conflict-free and no-redundancy.

The accuracy validation of the model defines the real world representation of the model. Therefore stakeholder's or expert's proof would fulfill this criteria in the best way by checking the model against their requirements. Unfortunately for this thesis no stakeholders or experts are given. Nevertheless the accuracy check has been done by implementing the database according to the case study results of [Isla11] i.e. goals, risk factors and events. Also check constraints and simple "null value not allowed"-constraints are used to ensure that there must not exist a particular concept i.e. goal which is not accurate.

Completeness validation can also be proofed best by checking the model against stakeholder's requirements and in more detail if all the necessary concepts, attributes and relationships are embodied in the domain model. In this thesis the database implementation ensures that there exists no concept table without the appropriate attributes and relationships to associated concepts. The attributes are chosen to fulfill the completeness criteria at least by its minimum i.e. ID, name, related concept's IDs, etc. For relationship's completeness all the multiplicities are implemented by defining primary keys and foreign keys in the relevant concept tables and junction tables with embodied keys of the associated concept tables.

The database implementation also ensures the conflict-free criteria of the model as there exists no concept table which is in concurrency of any other.

Non-redundancy can also be proofed by the implemented database as there is no particular concept table which incorporates similar information of any other concept table. Each concept table exists on its own with exclusively assigned attributes and relations.

5. Critical reflection

This section includes a critical reflection of the relevant items of this thesis and discusses indentified open issues of this work. In more detail the domain model and database implementation are analyzed about open issues which has to be addressed and discussed to ensure future work's success. The section is separated into a subsection about general open issues and another subsection pointing out the database restrictions.

5.1 General open issues

The domain model includes now all the relevant relationships and concepts for the defined tasks. More important every relationship is defined at its name and multiplicities which are necessary to ensure that the model is valid. Some concepts has been neglected in this domain model i.e. different goal concepts like hard or soft goals and the abstraction of sub goals which are assumed and aligned in this thesis and in the database implementation as goal concept. Also the concept of obstacles isn't considered in this thesis according to the nature defined in [Isla11]. The reason is that in [Isla11] only events with negative impact are considered whereas in this thesis events can also have positive impacts and therefore risk factors cannot be driven exclusively from obstacles which characterization it is to hinder goals.

Another open issue is that the database implementation isn't able to ensure every semantic constraint and further issues is incorporated like the automatically insertion of the agentID in certain concepts i.e. goal's attribute "agentID" has to be defined by the database engineer which aims to generate a new goal and store it in the database. Such issues can only be considered in further software implementations where the software is responsible to transmit the particular ID and save it i.e. in a new defined goal object.

The non existence of stakeholder's or experts for a more adequate proof of validation of the model is also an open issue. Nevertheless the thesis is able to ensure the validation without stakeholders and experts by implementing a database prototype considering data samples from literature and author's expertise. This lack of stakeholder's existence reflects in the issue of use case definitions i.e. reports and insert queries which are implemented as stored procedures in section 4.4. More adequate use cases could be defined with expert's knowledge.

More test data sample can also be defined as open issue. The actual test data is taken from case study results of [Isla11] also to ensure the adequacy of the sample as basis for the implementation of the database prototype. Because of the small quantity of the case study results some further assumptions had to be made to fulfill the criteria of accuracy and completeness of the model.

5.2 Database restrictions

Simple constraints like attribute values and value range can be implemented via database creation without any restrictions. Unfortunately not every feature and constraint of the model is able to be implemented and incorporated in the database model. Not only to consider all those issues but also to embody the business logic in a more accurate way software implementations have to be developed and capture them completely. Further on a list of database restrictions which are both concept specific and concept overlapping is shown and discussed.

User Interface: The user interface, meaning the fields of data inputs, is not implementable in database design and in the prototype as such. For catching illegal inputs as well as displaying error messages user interfaces are essential because such features cannot be integrated and considered in database implementation. But not only error messages should be displayed. All the activities a user is integrated like goal planning//definition, risk measuring, risk checking and performing treatments, the user should be informed about depending concepts like particular rules and risk limits as well as messages when a certain risk is exceeding the corresponding risk limit.

Backend: The software backend has to incorporate the business logic which means restrictions not only about attribute values but more important about logically accurate inputs. Most important logical restrictions are about risks and opportunities definition. Because in the designed GSRM domain model risks and opportunities in their definition cannot be restricted only to negative respectively positive impacts. Another important issue is the consideration about deleting elements in the database. There are relations to other tables and concepts which has to be considered when deleting a certain element. Although this issue is considered also in the database implementation, the user should be informed about dependencies to other elements and in dropping a whole table the backend should have embodied a order of dropping tables which are depended and should be dropped first. The backend has also to transmit certain IDs in each action the user performs. For example in activities like goal definition, risk measuring, risk checking or performing treatments the user's ID should be transmitted in the background to define database elements' attributes like agentID with the corresponding userID.

6. Summary and future work

The thesis incorporates the design of a goal-driven software project risk management (GSRM) domain model on the basis of previous research from [Isla11]. The domain model is aligned to COSO's ERM Framework components [COSOII04] considering the activities and elements of the Cybernetic Management Framework [Schwa12]. For validation proof the domain model is prototyped as a database implementation with relevant test data mainly from case study results of [Isla11]. The designed domain model has been proofed of validation using Microsoft SQL Server Management Studio [MSMS14] and the concept of stored procedures. Those stored procedures the conceptual model and the database prototype ensure the further usage for risk managers or software project manager as well as for software engineers.

For future work the domain model can be used as basis for software implementations to help software project managers and risk managers working in this context to manage their risks and reach project success. Nevertheless the domain model is not restricted to goal-driven software project risk management and can, under alignment of different contexts and use cases, be used for other kinds of risk management as well.

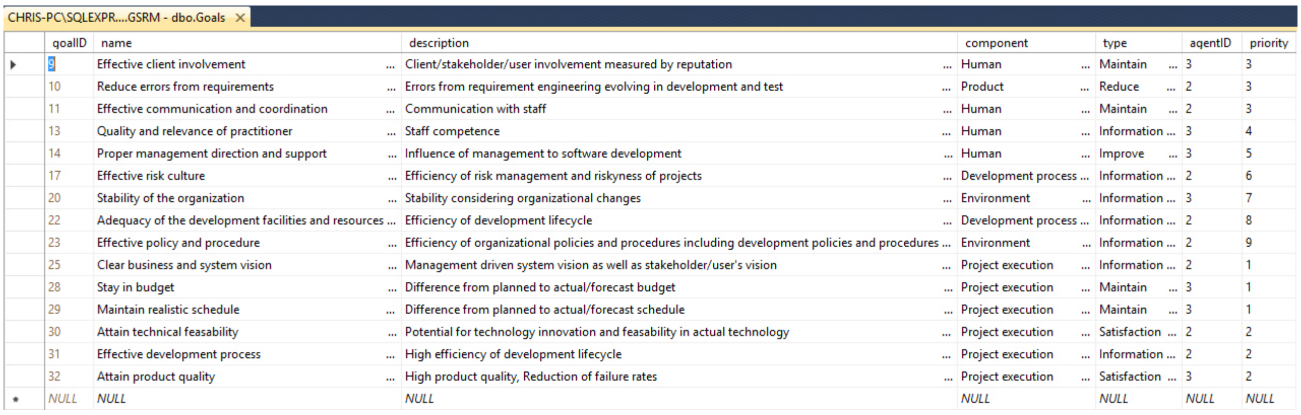
The addressed open issues from section 5.1 should also be considered in future work. In further researches some more adequate evaluation can be done by integrating stakeholders and experts to define more particular use cases as well as additional test data samples. Further software development have to be used to implement constrains and restriction as well as general features which could not be taken into account in the database prototype due to the lack of design features of the used database modeling tool or general database capabilities.

A more semantically issue for further approaches and future works would be the consideration of opportunity measuring. This thesis is designed in consideration of measuring only risks as opportunities are not assumed to eventuate in the first place.

7. Appendix

The appendix shows the database structure in detail and incorporates the used test sample of each concept extracted from the SQL Management Studio 2014 database. The last row of each sample shows the input parameters for a new element which is pre-described with NULL values.

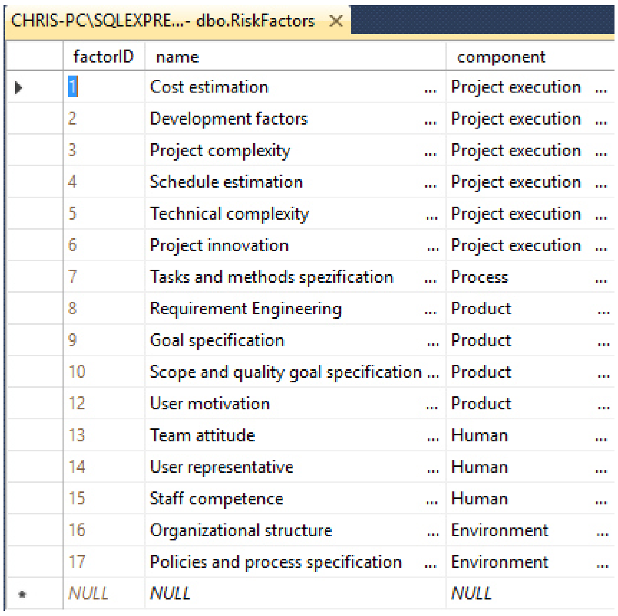
7.1 Appendix A: Test data samples from concepts



CHRIS-PC\SQLEXPR....GSRM - dbo.Goals

	goalID	name	description	component	type	agentID	priority
▶	9	Effective client involvement ...	Client/stakeholder/user involvement measured by reputation ...	Human ...	Maintain ...	3	3
	10	Reduce errors from requirements ...	Errors from requirement engineering evolving in development and test ...	Product ...	Reduce ...	2	3
	11	Effective communication and coordination ...	Communication with staff ...	Human ...	Maintain ...	2	3
	13	Quality and relevance of practitioner ...	Staff competence ...	Human ...	Information ...	3	4
	14	Proper management direction and support ...	Influence of management to software development ...	Human ...	Improve ...	3	5
	17	Effective risk culture ...	Efficiency of risk management and riskyness of projects ...	Development process ...	Information ...	2	6
	20	Stability of the organization ...	Stability considering organizational changes ...	Environment ...	Information ...	3	7
	22	Adequacy of the development facilities and resources ...	Efficiency of development lifecycle ...	Development process ...	Information ...	2	8
	23	Effective policy and procedure ...	Efficiency of organizational policies and procedures including development policies and procedures ...	Environment ...	Information ...	2	9
	25	Clear business and system vision ...	Management driven system vision as well as stakeholder/user's vision ...	Project execution ...	Information ...	2	1
	28	Stay in budget ...	Difference from planned to actual/forecast budget ...	Project execution ...	Maintain ...	3	1
	29	Maintain realistic schedule ...	Difference from planned to actual/forecast schedule ...	Project execution ...	Maintain ...	3	1
	30	Attain technical feasability ...	Potential for technology innovation and feasability in actual technology ...	Project execution ...	Satisfaction ...	2	2
	31	Effective development process ...	High efficiency of development lifecycle ...	Project execution ...	Information ...	2	2
	32	Attain product quality ...	High product quality, Reduction of failure rates ...	Project execution ...	Satisfaction ...	3	2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

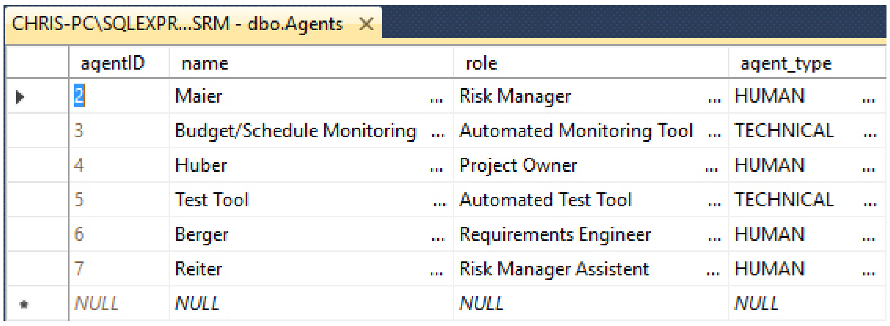
Figure 41- Goal test sample



CHRIS-PC\SQLEXPRE...- dbo.RiskFactors

	factorID	name	component
▶	1	Cost estimation ...	Project execution ...
	2	Development factors ...	Project execution ...
	3	Project complexity ...	Project execution ...
	4	Schedule estimation ...	Project execution ...
	5	Technical complexity ...	Project execution ...
	6	Project innovation ...	Project execution ...
	7	Tasks and methods spezification ...	Process ...
	8	Requirement Engineering ...	Product ...
	9	Goal specification ...	Product ...
	10	Scope and quality goal specification ...	Product ...
	12	User motivation ...	Product ...
	13	Team attitude ...	Human ...
	14	User representative ...	Human ...
	15	Staff competence ...	Human ...
	16	Organizational structure ...	Environment ...
	17	Policies and process specification ...	Environment ...
*	NULL	NULL	NULL

Figure 42- RiskFactor test sample

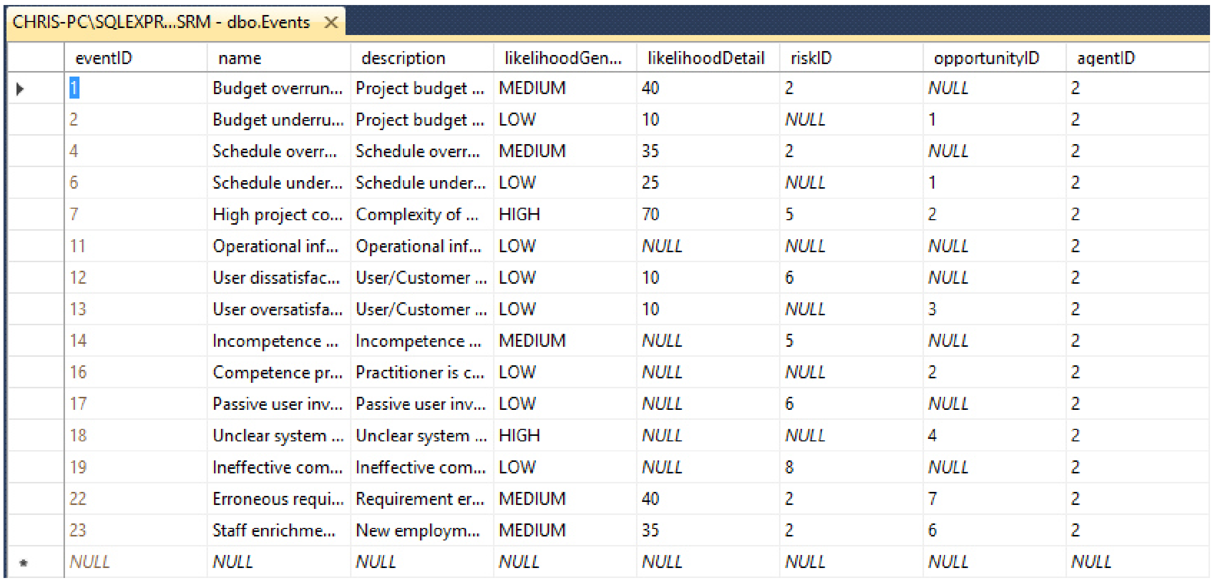


CHRIS-PC\SQLEXPR...SRM - dbo.Agents

	agentID	name	role	agent_type
▶	2	Maier ...	Risk Manager ...	HUMAN ...
	3	Budget/Schedule Monitoring ...	Automated Monitoring Tool ...	TECHNICAL ...
	4	Huber ...	Project Owner ...	HUMAN ...
	5	Test Tool ...	Automated Test Tool ...	TECHNICAL ...
	6	Berger ...	Requirements Engineer ...	HUMAN ...
	7	Reiter ...	Risk Manager Assistent ...	HUMAN ...
*	NULL	NULL	NULL	NULL

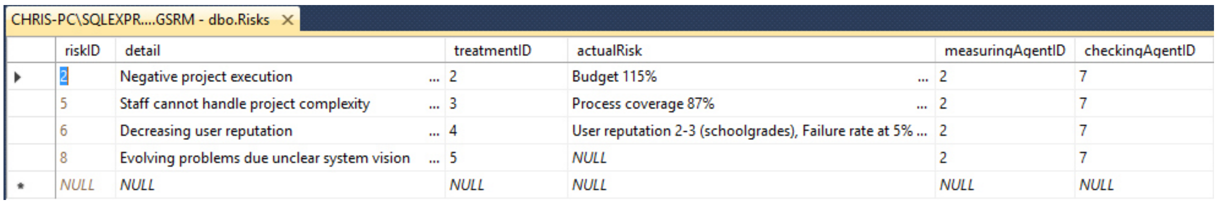
Figure 43- Agents test sample

7. Appendix



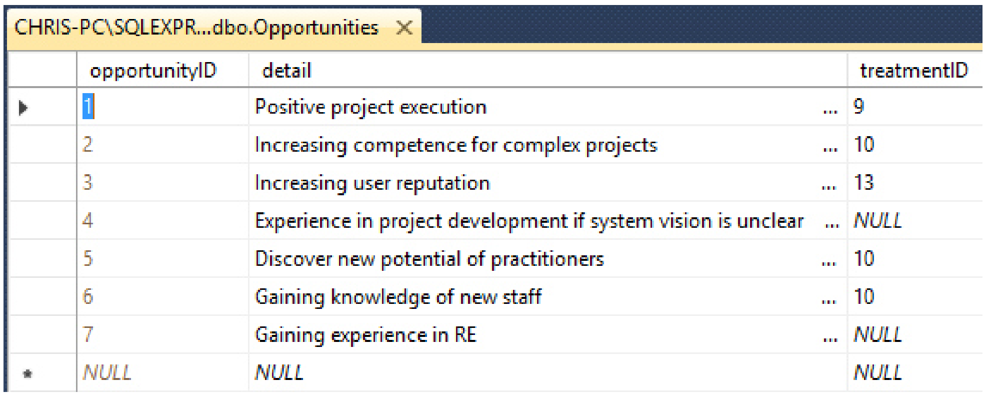
eventID	name	description	likelihoodGen...	likelihoodDetail	riskID	opportunityID	agentID
1	Budget overrun...	Project budget ...	MEDIUM	40	2	NULL	2
2	Budget underru...	Project budget ...	LOW	10	NULL	1	2
4	Schedule overr...	Schedule overr...	MEDIUM	35	2	NULL	2
6	Schedule under...	Schedule under...	LOW	25	NULL	1	2
7	High project co...	Complexity of ...	HIGH	70	5	2	2
11	Operational inf...	Operational inf...	LOW	NULL	NULL	NULL	2
12	User dissatisfac...	User/Customer ...	LOW	10	6	NULL	2
13	User oversatisfa...	User/Customer ...	LOW	10	NULL	3	2
14	Incompetence ...	Incompetence ...	MEDIUM	NULL	5	NULL	2
16	Competence pr...	Practitioner is c...	LOW	NULL	NULL	2	2
17	Passive user inv...	Passive user inv...	LOW	NULL	6	NULL	2
18	Unclear system ...	Unclear system ...	HIGH	NULL	NULL	4	2
19	Ineffective com...	Ineffective com...	LOW	NULL	8	NULL	2
22	Erroneous requi...	Requirement er...	MEDIUM	40	2	7	2
23	Staff enrichme...	New employm...	MEDIUM	35	2	6	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 44- Events test sample



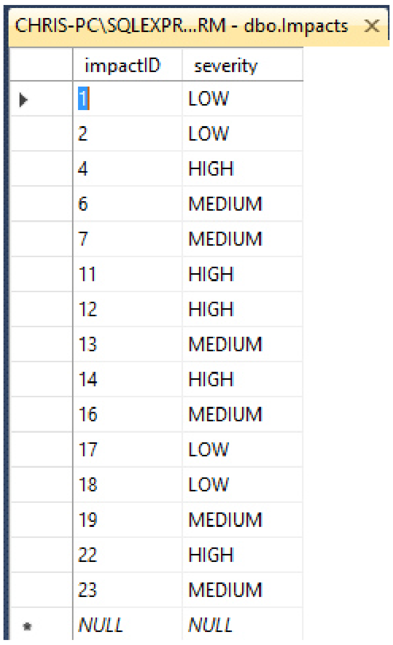
riskID	detail	treatmentID	actualRisk	measuringAgentID	checkingAgentID
2	Negative project execution ...	2	Budget 115% ...	2	7
5	Staff cannot handle project complexity ...	3	Process coverage 87% ...	2	7
6	Decreasing user reputation ...	4	User reputation 2-3 (schoolgrades), Failure rate at 5% ...	2	7
8	Evolving problems due unclear system vision ...	5	NULL	2	7
NULL	NULL	NULL	NULL	NULL	NULL

Figure 45- Risks test sample



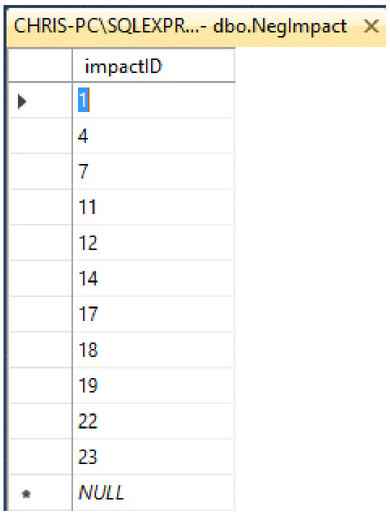
opportunityID	detail	treatmentID
1	Positive project execution ...	9
2	Increasing competence for complex projects ...	10
3	Increasing user reputation ...	13
4	Experience in project development if system vision is unclear ...	NULL
5	Discover new potential of practitioners ...	10
6	Gaining knowledge of new staff ...	10
7	Gaining experience in RE ...	NULL
NULL	NULL	NULL

Figure 46- Opportunities test sample



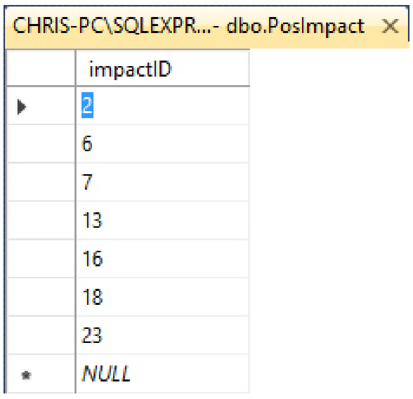
impactID	severity
1	LOW
2	LOW
4	HIGH
6	MEDIUM
7	MEDIUM
11	HIGH
12	HIGH
13	MEDIUM
14	HIGH
16	MEDIUM
17	LOW
18	LOW
19	MEDIUM
22	HIGH
23	MEDIUM
NULL	NULL

Figure 47- Impacts test sample


CHRIS-PC\SQLEXPR...- dbo.NegImpact

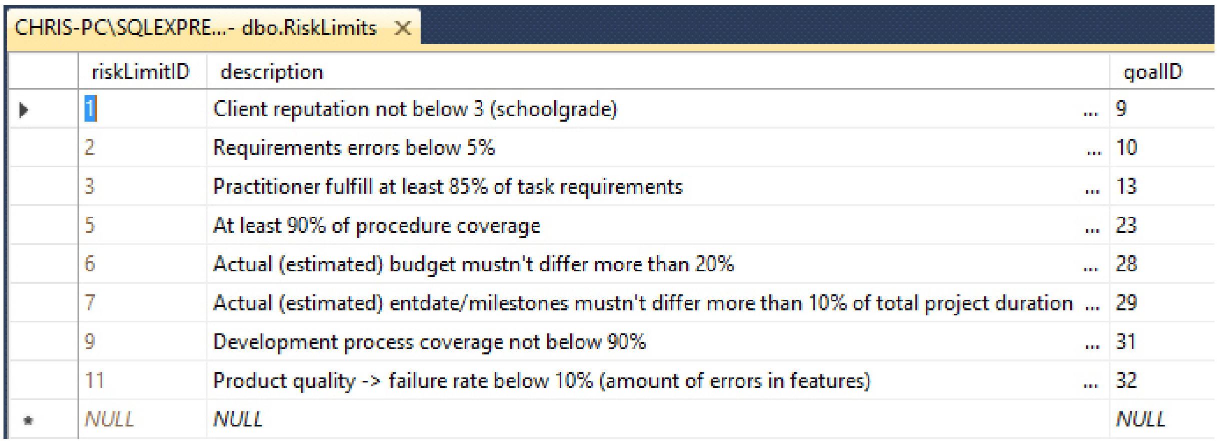
	impactID
▶	1
	4
	7
	11
	12
	14
	17
	18
	19
	22
	23
*	NULL

Figure 48- Negative Impacts test sample


CHRIS-PC\SQLEXPR...- dbo.PosImpact

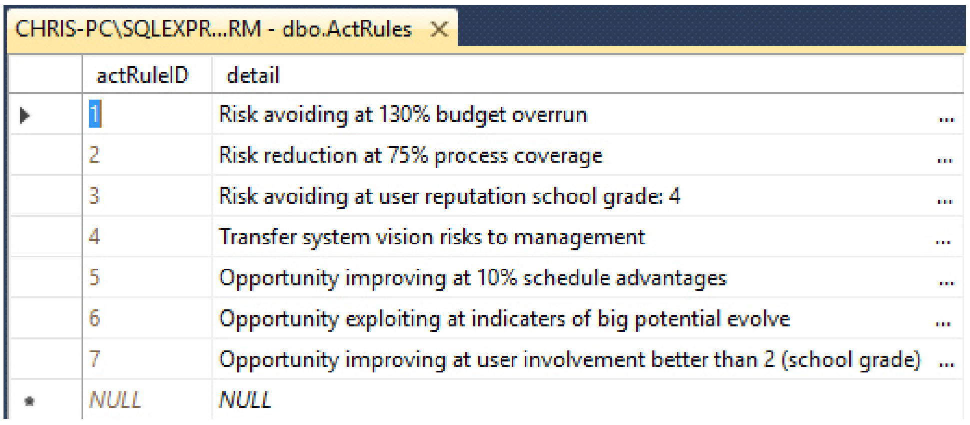
	impactID
▶	2
	6
	7
	13
	16
	18
	23
*	NULL

Figure 49- Positive Impacts test sample


CHRIS-PC\SQLEXPRE...- dbo.RiskLimits

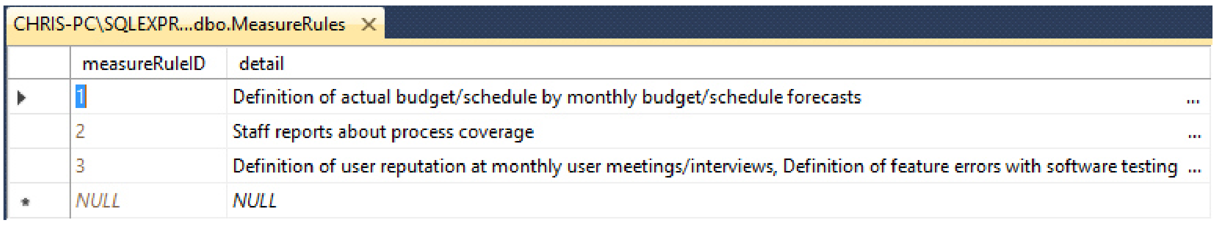
	riskLimitID	description	goalID
▶	1	Client reputation not below 3 (schoolgrade) ...	9
	2	Requirements errors below 5% ...	10
	3	Practitioner fulfill at least 85% of task requirements ...	13
	5	At least 90% of procedure coverage ...	23
	6	Actual (estimated) budget mustn't differ more than 20% ...	28
	7	Actual (estimated) entdate/milestones mustn't differ more than 10% of total project duration ...	29
	9	Development process coverage not below 90% ...	31
	11	Product quality -> failure rate below 10% (amount of errors in features) ...	32
*	NULL	NULL	NULL

Figure 50- RiskLimit test sample


CHRIS-PC\SQLEXPR...RM - dbo.ActRules

	actRuleID	detail
▶	1	Risk avoiding at 130% budget overrun ...
	2	Risk reduction at 75% process coverage ...
	3	Risk avoiding at user reputation school grade: 4 ...
	4	Transfer system vision risks to management ...
	5	Opportunity improving at 10% schedule advantages ...
	6	Opportunity exploiting at indicaters of big potential evolve ...
	7	Opportunity improving at user involvement better than 2 (school grade) ...
*	NULL	NULL

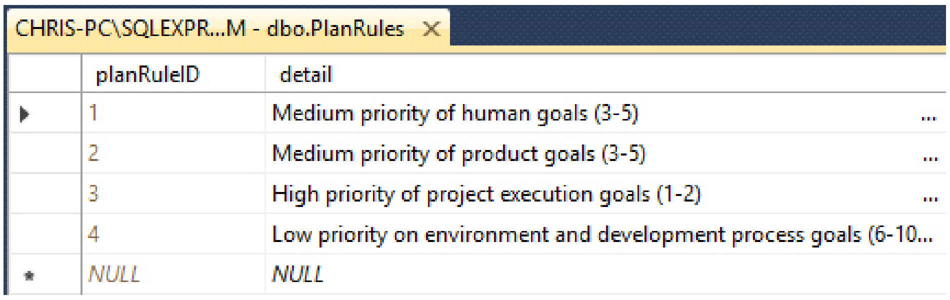
Figure 51- ActRules test sample



CHRIS-PC\SQLEXPR...dbo.MeasureRules

	measureRuleID	detail
▸	1	Definition of actual budget/schedule by monthly budget/schedule forecasts ...
	2	Staff reports about process coverage ...
	3	Definition of user reputation at monthly user meetings/interviews, Definition of feature errors with software testing ...
*	NULL	NULL

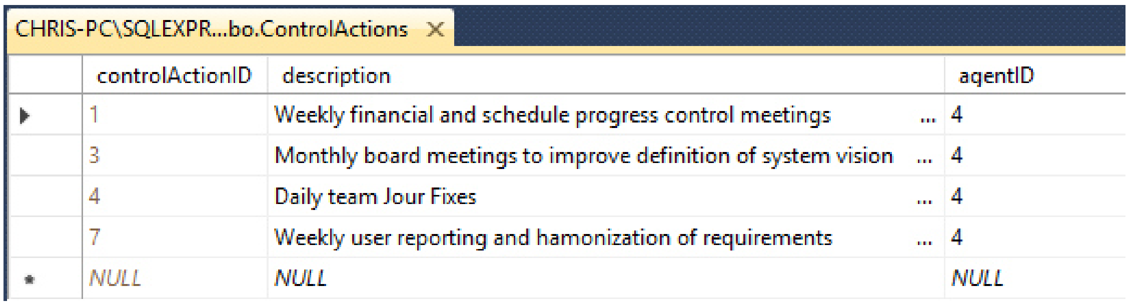
Figure 52- MeasureRules test sample



CHRIS-PC\SQLEXPR...M - dbo.PlanRules

	planRuleID	detail
▸	1	Medium priority of human goals (3-5) ...
	2	Medium priority of product goals (3-5) ...
	3	High priority of project execution goals (1-2) ...
	4	Low priority on environment and development process goals (6-10...
*	NULL	NULL

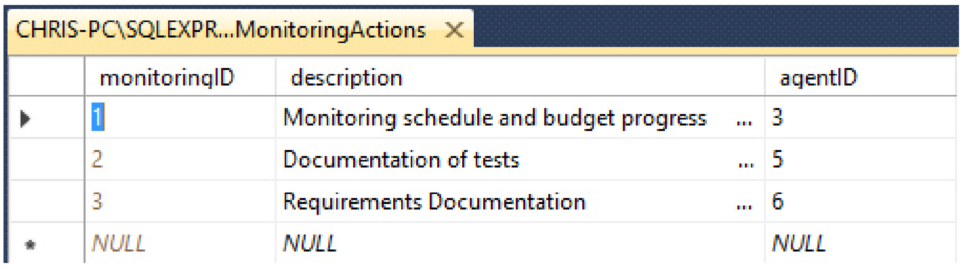
Figure 53- PlanRules test sample



CHRIS-PC\SQLEXPR...bo.ControlActions

	controlActionID	description	agentID
▸	1	Weekly financial and schedule progress control meetings ...	4
	3	Monthly board meetings to improve definition of system vision ...	4
	4	Daily team Jour Fixes ...	4
	7	Weekly user reporting and hamonization of requirements ...	4
*	NULL	NULL	NULL

Figure 54- ControlActions test sample



CHRIS-PC\SQLEXPR...MonitoringActions

	monitoringID	description	agentID
▸	1	Monitoring schedule and budget progress ...	3
	2	Documentation of tests ...	5
	3	Requirements Documentation ...	6
*	NULL	NULL	NULL

Figure 55- MonitoringActions test sample

8. Bibliography

Alberts C. J., Dorofee A. J., Higuera R., Murphy R. L., Walker J. A. and R., Williams C. [ADH+96]: Continuous Risk Management Guidebook. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA., 1996.

Akehurst D.H. and Bordbar B. [AkBo01]: On Querying UML Data Models with OCL. University of Kent at Canterbury. In: M. Gogolla and C. Kobryn (Eds.): UML 2001, LNCS 2185, pp. 91-103, 2001. Springer-Verlag Berlin Heidelberg 2001.

Basel Committee on Banking Supervision [Basel2–06]: International Convergence of Capital Measurement and Capital Standards – A Revised Framework – Comprehensive Version, http://www.bis.org/publ/bcbs128.htm, June 2006.

Bell D. [Bell03]: UML Basics: Part II: The activity diagram. The Relational Edge. E-Zine for the rational community. 2003.

Bell D. [Bell04]: UML Basics: The class diagram. An introduction to structure diagrams in UML 2. IBM Corporation, developerWorks, Technical topics. 15 September 2004.

Ballou B., Heitger D. [BH05]: A building-block approach for implementing COSO's Enterprise Risk Management Integrated Framework. Management Accounting Quarterly Winter 2005 Vol.6 No.2.

Bisbal J., Grimson J. [BiGr02]: Consistent database sampling as a database prototyping approach. J. Softw. Maint. Evol.: Res. Pract. 2002; 14:447–459 (DOI: 10.1002/smr.263). 2002.

Boehm B. W. [Boe91]: Software risk management: Principles and practices. IEEE Software, 8(1):32–41, 1991.

Boehm B. W., Ross R. [BoRo89]: Theory-W Software Project Management Principles and Examples, IEEE Transactions on Software Engineering, v.15 n.7, p.902-916, July 1989.

Bresciani P., Perini A., Giorgini P., Giunchiglia F., and Mylopoulos J. [BPG+04]: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems, 8:203–236, 2004.

BrooksF.P. [Bro95]: The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Professional, 1995.

Carr M., Konda S., Monarch I., Ulrich C., and Walker C. [CKM+93]: Taxonomy based risk identification (cmu/sei-93-tr-6, ada266992). Technicalreport, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA., 1993.

Connolly, T. and Begg, C. [CoBe02]: Database Systems: A Practical Approach to Design, Implementation, and Management, 3rd Ed. Addison-Wesley, Harlow, England, 2002.

Committee of Sponsoring Organizations of the Treadway Commission [COSOII04]: Enterprise Risk Management – Integrated Framework, September 2004, http:// www.coso.org/-ERM.htm.

Committee of Sponsoring Organizations of the Treadway Commission [CSAT04]: Enterprise Risk Management – Integrated Framework Application Techniques, September 2004.

Dubois E., Heymans P., Mayer N., Matulevicius R. [DHMM10]: A systematic approach to define the domain of information system security risk management, S. Nurcan, C. Salinesi, C. Souveyet, J. Ralyte (Eds.), Intentional perspectives on information systems engineering, Springer Berlin Heidelberg (2010), pp. 289–306.

Dumas M. and ter Hofstede A. [DuHo01]: UML Activity Diagrams as a Workflow Specification Language. Cooperative Information Systems Research Centre Queensland University of Technology. In: M. Gogolla and C. Kobryn (Eds.): UML 2001, LNCS 2185, pp. 76-103, 2001.Springer-Verlag Berlin Heidelberg 2001.

Dardenne A., van Lamsweerde A., and Fickas S. [DvLF93]: Goal-directed requirements acquisition. Science of Computer Programming, 20(1-2):3– 50, 1993.

Evaristo R., Prikladnicki R., Yamaguti M. H., Audy J. L. N. [EPYA06]: Risk management in distributed it projects: Integrating strategic, tactical, and operational levels. International Journal of e-Collaboration, 2:1–18, 2006.

Foerster H.v. [Foer03]: Cybernetics of Cybernetics, in: Foerster H.v.: Understanding Understanding – Essays on Cybernetics and Cognition, Springer, New York, 2003, 283–286.

Fowler, M. [Fow03]: Patterns of Enterprise Application Architecture. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2002.

Geerts G., McCarthy W.F. [GeMc02]: An ontological analysis on the economic primitive of the extended REA enterprise information architecture. International Journal of Accounting Information Systems. 3, 1-16. 2002.

Islam S. [Isla09]: Software Development Risk Management Model – A Goal Driven Approach. ESEC/FSE Doctoral Symposium '09 Proceeding of the doctoral symposium for ESEC/FSE Doctoral Symposium. P.5-8. New York, NY, USA. 2009.

Islam S. [Isla11]: Software Development Risk Management Model- a goal-driven approach. Diss. Technische Universität München, Institute für Informatik, 2011. https://mediatum.ub.tum.de/doc/1002328.

ISO-Management System Standards [ISO-09]: https://www.iso.org/obp/ui/#iso:std:iso:guide:73:ed-1:v1:en:term:1.1

ISO-Management System Standards [ISO-MSS11]: Download – November 2011 http://www.iso.org/iso/iso_catalogue/management_and_leadership_standards/

management_system_basics.

Jaafar J., Iqbal U., Lai F. [JIL15]: Software Effective Risk Management: An Evaluation of Risk Management Process Models and Standards. In: K.J. Kim (ed.), Information Science and Applications, Lecture Notes in Electrical Engineering 339, pp. 837-844. Springer Verlag Berlin Heidelberg 2015.

Karolak D.W. [Kar95]: Software Engineering Risk Management. IEEE Computer Society Press, 1995.

Kontio J. [Kon01]: Software Engineering Risk Management: A Method, Improvement Framework and Empirical Evaluation. PhD thesis, Helsinki University of Technology, 2001.

Kop C. [Kop12]: Checking Feasible Completeness of Domain Models with Natural Language Queries. Proceedings of the Eighth Asia-Pacific Conference on Conceptual Modeling (APCCM 2012), Melbourne, Australia. P.33-42. 2012.

Lin L, Nuseibeh B, Ince D, Jackson M [LNIJ04]: Using Abuse Frames to Bound the Scope of Security Problems. In: Proceedings of the 12th IEEE International Conference on Requirements Engineering (RE '04), pp. 354-355, IEEE Computer Society. 2004.

McCarthy W. [McC82]: The REA Accounting Model – A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review, LVII(3), 554-578. 1982.

Mouratidis H, Giorgini P, Manson GA, Philp I [MGMP02]: A Natural Extension of Tropos Methodology for Modeling Security. In: Proceedings of the Agent Oriented Methodologies Workshop (OOPSLA'02). 2002.

Prikladnicki R., Nicolas A., Jorge L., and Evaristo R. [PNJE06]: A reference model for global software development: Findings from a case study. In ICGSE '06: Proceedings of the IEEE international conference on Global Software Engineering, pages 18–28, Washington, DC, USA, 2006. IEEE Computer Society.

Risk Management Standard [RMS09] (ISO 31000:2009): Risk Management – Principles and guidelines, 1st edition, 2009–11–15.

Schwaiger W., Abmayer M. [ScAb13]: Accounting and Management Information Systems: A Semantic Integration, International Conference on Information Integration and Web-based Applications & Services (iiWAS2013), pp.346-353, Association for Computing Machinery, Vienna (2013).

Schwaiger W.S. [Schw12]: Risk Management: Comprehensive Integration into the Enterprise Management. In Frick R., Gantenbein P. and Reichling P. (editors), Asset Management. Haupt, Berlin, Stuttgart and Vienna, pp. 420-459.

Selic B. [Sel07]: A Systematic Approach to Domain-Specific Language Design Using UML IBM Canada. In: Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07). 2007.

Simsion, Graeme. C. & Witt, Graham. C. [SGWG05]: Data Modeling Essentials.3rd Edition. Morgan Kauffman Publishers. ISBN 0-12-644551-6, 2005.

Sisti F. and Joseph S. [SJ94]: Software risk evaluation method version 1.0. Technical report, SEI, Carnegie -Mellon University, 1994.

Sehrawat, Munsi, Jain [SMJ14]: Risk Management in Software Projects. IJCSMC, Vol. 3, Issue. 10, October 2014, pg.845 – 849 2014.

SQL Server 2014 Management Studio [SSMS14]: https://msdn.microsoft.com/en-us/library/ms188430.aspx. Microsoft® 2014.

ShanksG., Tansley E., Weber R. [STW03]: Using Ontology to validate conceptual model. Communication of the ACM Vol.46, No.10, P.85-89. October 2003.

Unified Modeling Language [UML11]: Superstructure, Version 2.4.1, 2011–08–06, www.uml.org.

van Lamsweerde A. [vL09]: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, 2009.

van Lamsweerde A [vLam04]: Elaborating Security Requirements by Construction of Intentional Anti-Models. In: Proceedings of the 26th International Conference on Software Engineering (ICSE '04), pp 148-157, IEEE Computer Society. 2004.

Wiener N. [Wiener48]: Cybernetics: Or the Control and Communication in the Animal and the Machine, MIT-Press, Cambridge, 1948.

WarmerJ., Kleppe A. [WK98]: The Object Constraint Language: Precise Modeling With Uml. Addison-Wesley Object Technology Series. 13 October 1998.

Zachman J. A. [Zach87]: *A framework for information systems architecture*. In: *IBM Systems Journal*. 26, Nr. 3, 1987, S. 277-293.