1 Introduction

This master thesis concludes the study of computer science at the Rijks Universiteit of Leiden. The mentor for this project is dr. L.P.J. Groenewegen. The topic addressed in this master thesis is the investigation of how to combine the ISO-9000 quality standards with software process modeling. The software engineering group in Leiden, with its software process formalism SOCCA, is a member of the European research group PROMOTER. As an extension to the models given so far in this software process formalism we shall concentrate on combining ISO-9000 quality standards with SOCCA.

PROMOTER (Process modeling techniques) is a European project, financed by ESPRIT, wherein a number of universities participate to exchange ideas on software process modeling techniques. No single existing specification formalism is sufficiently suitable to model different and complex software process situations, which models both data, behavior, as well as the coordination of human and non-human agents. To this aim the university of Leiden proposed a specification formalism, which combines the best parts of different formalisms. The proposal is published in paper [1]. The same paper is also covered in book [9]. The formalism discussed in the paper is SOCCA, which consists out of the following parts:

- Extended entity relationship diagrams (EER), based on object oriented modeling for the data perspective;
- State transition diagrams + PARADIGM (STD + PARADIGM), for the behavior and communication perspective;
- Object flow diagrams (OFD), for the process perspective.

The paper concentrates on object oriented class diagrams, state transition diagrams and PARADIGM, as well as on their integration. It briefly sketches the use of object flow diagrams.

Due to the complexity of most software design processes it is necessary that higher level models precede lower level models, until the completion of the final computer program. From a project point of view the entire development process involves many activities. It is believed that through process modeling it is possible to design and implement a computer model for the life cycle of an arbitrary software engineering project. Such a computer model should support all in the

software engineering process involved human and non human agents and retrieve all kinds of information from it.
For instance, from a project management perspective, this information ranges from topics like: the current project status; the tasks and the people to whom these tasks are assigned; the available documents and their versions; to topics that answer questions like: what happens to the project if a different approach is taken; or what happens if the current model is changed into a new one.

In the world of software process modeling the ISWP-6 [4] and ISWP-7 [3] examples present a problem description. They provide a standard problem, to which all modeling approaches can be applied and thus can be evaluated and compared to one another. In brief these problems describe a structure of a software project.

The critical success factors of software engineering projects are the delivery of products within agreed costs, time and quality. Upon delivery, it is quite easy to measure the actual costs and time versus the agreed costs and time. However at this stage of the project there are little opportunities left (within the remaining budget and time) to take corrective actions if so required. More specific, any errors made during the earlier phases of the project become more expensive to correct when discovered at later stages of the project. Measuring the quality of the product at delivery (so-called ‘black-box testing’) is often impossible. At most the product can be measured against its specification and some agreed test plan may be carried out to verify whether the product meets its specifications.
For instance the final computer program can meet its specifications and pass the test according to the agreed test plan. However the underlying database may not be properly normalized. Certain software routines may compensate the resulting inconsistencies or duplication of data. Such poor designs result in huge amounts of work in case modifications are needed during the life cycle of the product. This means that at earlier stages of the project quality checks are required, such as design reviews.

The quality of an arbitrary software engineering project can be classified as:

a) Quality standards for the software development process. This type of quality relates to the business processes that are needed to develop the software product within agreed

costs, time and specifications. It is believed that these quality aspects also result in a software product of high internal product quality.

b) Internal product quality. This is the 'technical' quality of the design and the software program to be delivered. SOCCA, which uses the concepts of EER, STD and Paradigm, and OFD, addresses some of these quality aspects, like consistency, design verification and efficiency. However further research is needed to define other quality aspects.

This master thesis addresses the quality of an arbitrary software engineering project as expressed under (a) above. It is believed that the quality of the software engineering process largely determines the quality of the end product, and the delivery of the product within agreed costs and time. The thesis of this paper is to model the tasks between people (business processes) with SOCCA during an arbitrary software engineering project and whereby these tasks comply with certain quality standards. Concerning these quality standards the focus is on ISO-9000.
The ISO-9000 quality standards specify the quality system requirements. These quality system requirements primarily focus on the various project stages (from design to delivery and servicing) which are needed to ultimately deliver the product according to the customer's expectations. These standards are primarily aimed to prevent nonconformity through all project stages. The costs of the actions to prevent nonconformity are in general recovered by the much lower costs of the otherwise required corrective actions. This illustrates the importance of a quality system which aims at preventing nonconformity.

The organization of this master thesis is as follows:

- ISO-9000 standards and their application to an arbitrary software engineering project.
- Modeling the tasks between people (business processes) during an arbitrary software engineering project with the concepts of SOCCA.
- Conclusion and recommendations for further research topics.

The idea is that the concepts of SOCCA and the quality standards of ISO-9000 set the framework for modeling an arbitrary software engineering project. This approach extends the ISWP-6 and ISWP-7 examples, since it also considers the quality management aspects of a project. The concluding

section addresses the following aspects of the paper (and which is also the focus of this study):

- Conclusions on quality management during an arbitrary software engineering project.
- Conclusions on the ISWP-6 and ISWP-7 examples regarding their compliance to the ISO-9000 requirements.
- The fitness of the SOCCA concepts to model such a process. Since no standard example (reference example) is used, the modeling techniques with SOCCA shall not be evaluated against other modeling techniques.

2 ISO-9000

2.1 Introduction

Over the last number of years companies pay more attention to quality. In this context quality is not just a pure objective measurable property of the product or service, but relates to the expectations of the customer. This quality notion embodies much more than the manufacturing of a product or providing a service of so-called high ‘technical’ quality, since it includes well-described business processes, organizational structures, procedures and responsibilities needed to deliver the product or service to the customer’s expectations. This kind of quality is called a quality system.

ISO-9000 describes and explains the requirements of a quality system. However ISO-9000 does not prescribe any specific model for a quality system. Therefore companies need to design their quality system (that is describe their business processes, organizational structure, procedures and responsibilities) as they see fit for their organization.

If the quality system of a company meets the ISO-9000 quality

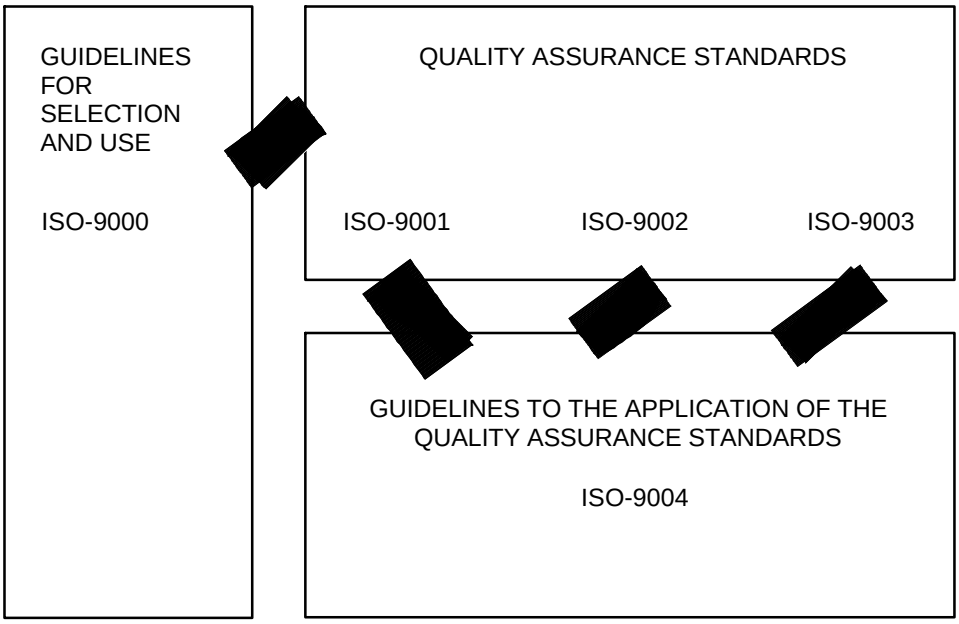

Figure 1 ISO-9000 structure

requirements, then that quality system can be ISO-9000 certified. The consequence is however, that at regular intervals the company is audited to ensure they adhere to their quality system. If the company fails to meet the requirements of their certified quality system, then they must either take corrective actions to comply with ISO-9000 or otherwise they will lose the ISO-9000 certification.
The advantage of having an ISO-9000 certification is that customers often evaluate products or services, besides topics like prices, on whether the manufacturer or service provider can ensure that the product or service is in compliance with the agreed quality. Moreover a well-defined quality system guarantees a constant product or service quality, thus preventing unexpected results by the customer.
On the other hand customers consider companies not having an ISO-9000 certification better (provided they have some quality system in place) compared to companies who lose their certification. Customers regard companies who lose their ISO-9000 certification as companies with organization problems and who cannot guarantee delivering products or services with constant and agreed quality anymore.

The term ‘ISO-9000’ (as used in the text so far and commonly referred to) actually refers to the ISO-900x quality standards. Figure 1 shows the structure of the ISO-900x standards. The ISO-9001 / 9002 / 9003 series contain the quality requirements. The ISO-9000 series provide the guidelines for the selection and use of the ISO quality requirements and the ISO-9004 series explain the ISO quality requirements. Besides these standards are ISO-8402 (which is the quality vocabulary) and ISO-10011 (which contains the guidelines for auditing quality systems). In the appendix the available ISO-900x quality standards are listed.

The European Committee for Standardization (CEN) have accepted the ISO-900x quality standards. Thus the ISO-900x standards are published as EN-2900x standards. The member countries of CEN have committed themselves to incorporate the European standards into their national standards. In The Netherlands the ISO-900x quality standards are translated into Dutch and published as NEN-ISO-900x.

To illustrate the importance of a quality system the costs incurred by a failure will be considered. The costs to correct a failure increases progressively when the time increases between when the failure occurred and when the failure is discovered. For instance a failure during the

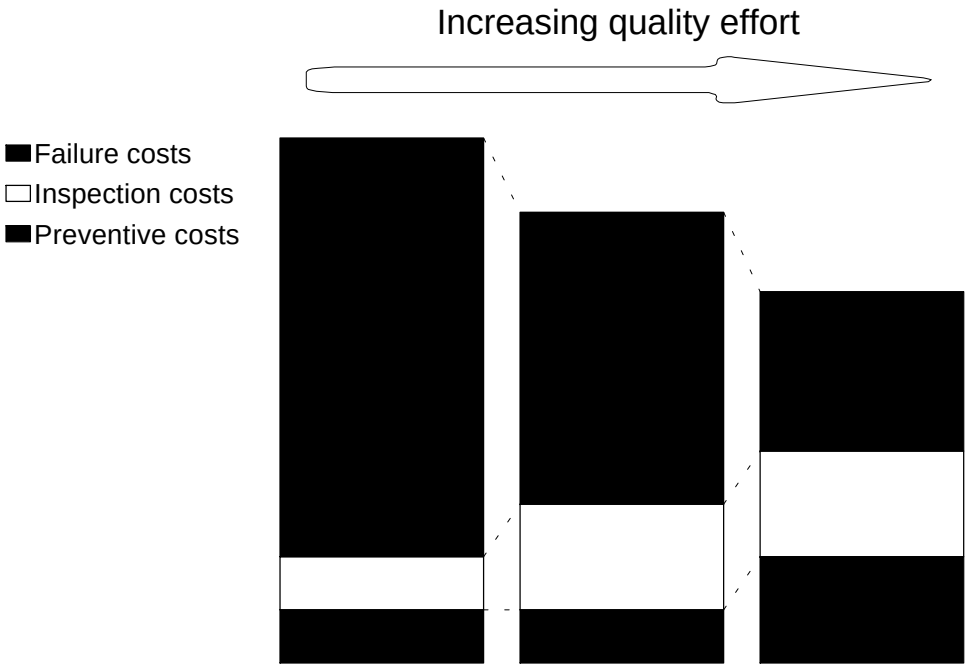

Figure 2 Costs

analysis phase can be corrected relatively easy during the same phase but may be very expensive to correct when discovered during the programming phase or worse at delivery of the final product. On the other hand a programming error will be easier to correct during testing, but again will become more expensive to correct once the product is delivered (due to the costs of redistribution, loss of image and such). To reduce these failure costs it is essential that defects are discovered as soon as possible. As a consequence during the execution of a project the following issues need to be considered:

a) preventive actions;
b) inspection.

The preventive actions focus on the processes needed to achieve the deliverables, whereas inspection focuses on the deliverables itself. Both preventive actions and inspection have their costs, but reduce the costs of failures as shown in figure 2. In this context failures are defined as any defect which need some corrective action. From the same figure it can also be derived that it is important to measure

all costs to determine the effectiveness of any quality improvement action.

ISO-9000 primarily focuses on preventing nonconformity through all project stages and pays less attention to inspection and inspection techniques. The same holds for quality improvement activities. Quality improvement activities result in a change of quality procedures and according to ISO-9000 these changes need to be documented. But apart from the need of documenting these changes, ISO-9000 pays little attention to quality improvement once a quality system complies with its standards.

2.2 Scope and Definitions

As mentioned in the previous chapter and shown in figure 1, the ISO-900x standards are structured around the following standards:

- ISO-9001 / 9002 / 9003, which contain the actual quality assurance requirements. A quality system is always certified against one of these standards.
- ISO-9000 series, which provide the guidelines for the selection and use of the ISO-9001 / 9002 / 9003 standards.
- ISO-9004 series, which explain the quality assurance requirements.

In my opinion the latter two series contain quite some overlap.

Besides the above quality standards are:

- ISO-8402, which contains the quality vocabulary.
- ISO-10011 series, which contain the guidelines for auditing quality systems.

In the appendix a complete list of all the available ISO-900x standards are given.

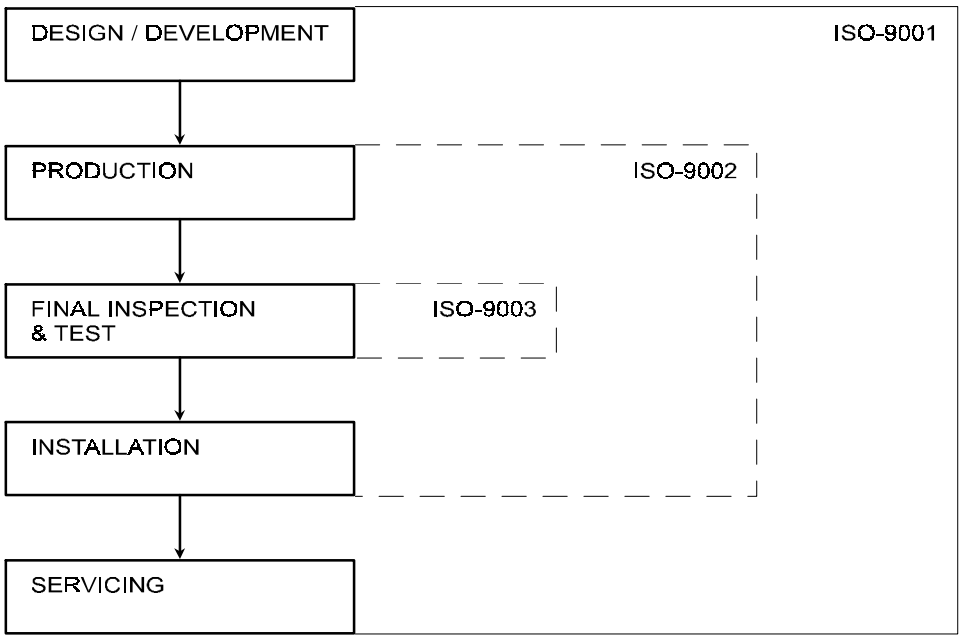

Figure 3 ISO-9000 coverage

The ISO quality assurance standards, to which a quality system can be certified, are covered in one of the following standards (see figure 3):

ISO-9001	Model for quality assurance, which covers: design and development, production, final inspection & testing, installation and servicing.
ISO-9002	Model for quality assurance, which covers: production, final inspection & testing and installation.
ISO-9003	Model for quality assurance, which covers: final inspection & testing.

As can be concluded by these titles, the standards overlap one another. The range of ISO-9001 is considerably larger compared to the other 2 standards (the range of ISO-9003 is the smallest of these standards). Certification is always against one of these standards. For an ISO certification the ISO requirements that are not met, must be explicitly stated with explanations why they are not applicable. For this reason it is more convenient to certify for a lower standard, if applicable, than for the more prestigious ISO-9001 standard.

The ISO quality standards can be applied to many different types of companies, such as manufactures of any product or service companies. Typically the quality system applies to and interacts with all activities relevant to the quality of a product or service. It involves all phases from initial product or service identification to the final satisfaction of customer requirements and expectations. These phases and activities may include (depending on the company) the following:

a) identify customer needs (e.g. with market research);
b) design / engineering and product development;
c) procurement;
d) process planning and development;
e) production;
f) inspection and testing;
g) packaging and storage;
h) sales and distribution;
i) installation and operation;
j) technical assistance and maintenance;

k) disposal after use.

Such a model (from *identifying customer needs* until *disposal after use*) is called a quality loop (see figure 4).

It is also possible that within a company only some departments have an ISO quality certification, and possibly independent from one another. Therefore a company may have several ISO quality certificates. This does however not mean that all departments or sections within a company have a quality system in place.
For instance within a company a certain production train (say for the manufacturing of certain chemicals) may have an ISO-9001 certificate (possibly with some appropriate exclusions), another supporting section may have an ISO-9002 certificate, whereas other sections may have no certificate at all (or are in the process of obtaining a certificate).

For the IT-sector (hardware and software companies, but also for an IT section within large companies) the following ISO standards are important:

ISO-9000-3	Quality management and quality assurance standards - Part 3; Guidelines for the application of ISO-9001 to the development, supply and maintenance of software.
ISO-9001	Quality systems - Model for quality assurance in design / development, production, final inspection & testing, installation and servicing.
ISO-9004	Quality management and quality system elements - Guidelines.
ISO-9004-2	Quality management and quality system elements - Part 2: Guidelines for services.
ISO-8402	Vocabulary

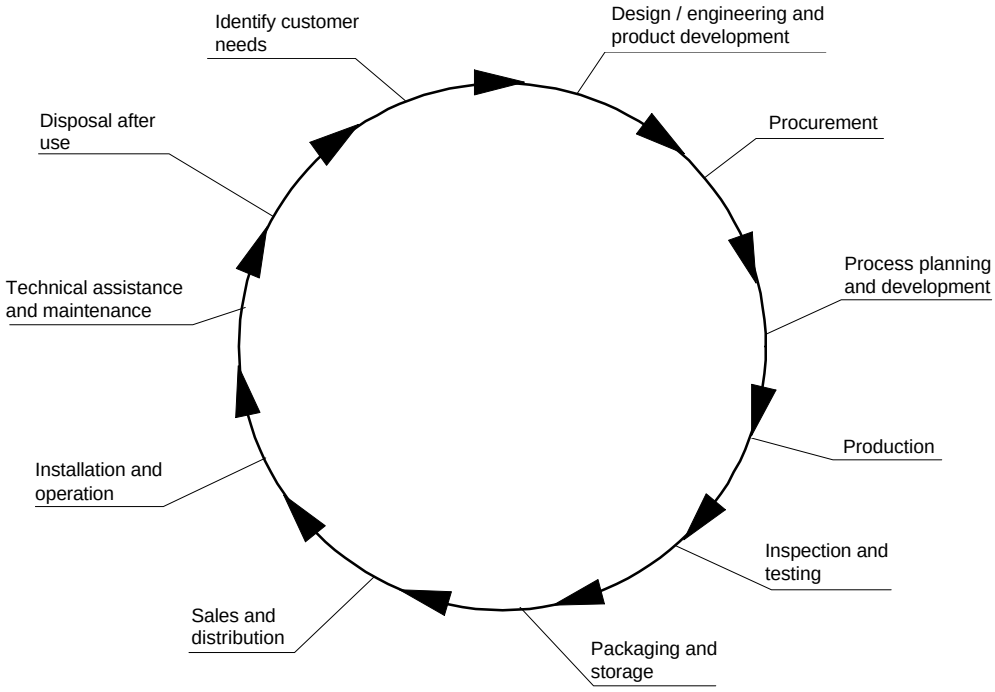

Figure 4 Quality loop

The actual quality system requirements are stated in ISO-9001 in 20 articles, which covers the following subjects:

1) Management responsibility
2) Quality system
3) Contract review
4) Design control
5) Document control
6) Purchasing
7) Purchaser supplied product
8) Product identification and traceability
9) Process control
10) Inspection and testing
11) Inspection, measuring and test equipment
12) Inspection and test status
13) Control of non-conforming product
14) Corrective action
15) Handling, storage, packaging and delivery
16) Quality records
17) Internal quality audits
18) Training
19) Servicing
20) Statistical techniques

ISO-9002 and ISO-9003 contain the same, but fewer articles. Unfortunately the reviewed standards do not use the same numbering and terminology, which is confusing from time to time. In 1994 new ISO standards were published, wherein the ambiguity and numbering mismatch was resolved.

The next two chapters discuss the ‘application of the ISO-9000 quality standards’ and ‘the quality guidelines for the development, supply and maintenance of software’. The first chapter presents an overview in what is involved in a quality system and discusses its key factors. The second chapter covers the guidelines that are specific for the IT sector.
This approach gives a better understanding of the ISO requirements, compared to discussing the ISO-9001 standard itself. The ISO-9001 standard is a 7 page document that only lists the 20 quality requirement articles and it gives some definitions. The interpretation of these articles and their fitness for the business is left to the reader.

2.3 Application of ISO-9000 Quality Standards

2.3.1 Overview

The central theme in the quality management guidelines is that a company, to meet its objectives, should organize itself in such a way that the technical, administrative and human factors affecting the quality of its products or services are under control. As stated in ISO-9004, the quality management system has two interrelated aspects:

a) For the company there is a business need to obtain and to maintain the desired quality at an optimum cost. The realization of this quality aspect is related to the effective utilization of the technological, human and material resources available to the company. The company needs to consider:
   - Risks related to deficient products or services, which may lead to loss of image or reputation, loss of market, claims, waste of human and financial resources.
   - Costs due to design deficiencies, including unsatisfactory materials, rework, repair, replacement, loss of production, warranties and field repair.
   - Benefits, such as increased profitability and market share.

b) For the customer there is a need for confidence in the ability of the company to deliver the desired quality as well as the consistent maintenance of that quality. The customer needs to consider:
   - Risks such as those connected with the health and safety of people, dissatisfaction with products or services, availability, marketing claims and loss of confidence.
   - Costs such as safety precaution actions, acquisition, operating, maintenance, downtime, repair, and possible disposal costs.
   - Benefits, such as reduced costs, improved fitness for use, increased satisfaction and growth in own confidence.

Each of the above quality aspects requires objective evidence in the form of information and data concerning the quality of the quality management system and the quality of the company’s products.

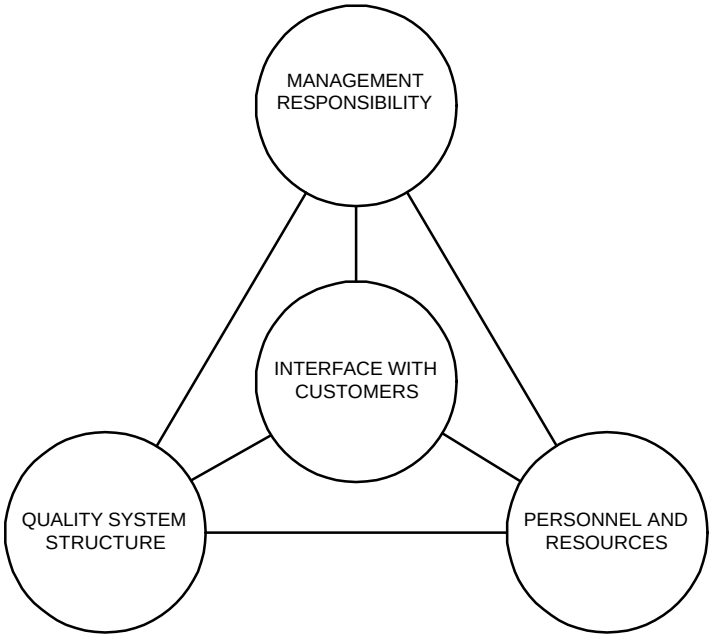

Figure 5 ISO-9000 key factors of a quality system

An effective quality management system should be designed to satisfy the customer needs and expectations, while protecting the company's interests. Figure 5 shows the key factors of a quality system (source ISO-9004-2). It shows the interface with the customers in the center of the 3 main aspects of a quality system. The customer's needs and expectations can only be satisfied if the management responsibility, personnel and material resources as well as the quality system structure are properly balanced to one another.

2.3.2 Management responsibility

Management is responsible for the quality policy and the commitment thereto. The quality policy states the objectives concerning the quality that the company wants to achieve, as well as the available resources to achieve that quality. A successful implementation of a quality system depends on the involvement of management. The management responsibility involves the following tasks:

a) Management needs to develop and document the quality policy, relating to:
   - quality level of the product or service to be provided;
   - image and reputation of the organization for quality;

- objectives for product or service quality;
- approach to be adopted in pursuit of quality objectives;
- function of personnel responsible for implementing the quality policy.

Management must ensure that the organization understands, executes and maintains the quality policy.

b) The realization of the quality policy requires the identification of the primary objectives for establishing the quality objectives. These primary objectives include:
   - customer satisfaction with the product or service in accordance with appropriate standards and ethical codes;
   - continuous improvement of the product or service;
   - protection of the environment and society at large in respect of manufacturing or service activities;
   - efficiency in manufacturing or providing the service.

   Management must translate these primary objectives in a series of quality objectives and activities.

c) To achieve the quality objectives, management should set up an effective structure for the quality system to enable an effective control, evaluation and improvement of the product or service quality. Management should designate representatives who are responsible for ensuring that the quality system is developed, established, audited, continuously measured and reviewed for improvement.

2.3.3 Personnel and resources

Management should provide sufficient and appropriate resources essential for the implementation of the quality system and the achievement of the quality objectives. These are:

a) Personnel, where special attention needs to be paid to motivation, training and career development, and communication skills.
b) Material resources, such as accommodation provisions and tools.

2.3.4 Quality system structure

The quality system structure emphasizes preventive actions that avoid the occurrence of problems while not sacrificing the ability to respond to and correct failures should they

occur. The following aspects are involved in a quality system structure:

a) The quality system as it applies to and interacts with the activities appropriate to the quality of the product or service:
   It involves all phases from initial product or service identification to the final satisfaction of customer requirements and expectations. The model of these phases is called a quality loop (see also figure 4).

b) The documentation of all the elements of the quality system adopted by the company:
   The appropriate quality system documentation includes the following:
   - The quality manual, which contains the quality policy, the quality objectives, an adequate description of the quality system and the quality practices of the organization. It serves as a reference in the implementation and maintenance of the quality system.
   - For projects relating to new products, services or processes a quality plan needs to be written which is consistent with all other requirements of the company's quality system. These quality plans define:
     - the quality objectives to be attained;
     - the allocation of responsibilities during the different phases of the project;
     - the specific procedures, methods and work instructions which apply to the project;
     - suitable testing, inspection, examination and audit programs at appropriate phases of the project;
     - a method for modifications as a project proceeds.
   - The quality system requires that quality records are maintained to demonstrate achievement of the required quality and verify effective operation of the quality system. These records may include test data, audit reports, quality cost reports, and so on.
   - Methods need to be established for document change control, such as check on correctness and adequacy, version control, removal of obsolete documentation, and so on.

2.3.5 Interface with customers

Effective communication between the company's personnel and the customer is crucial to the quality of the product or service perceived. This effective communication embodies:

a) Provide customers with a clear description of the product or service, when it is available and at what costs.
b) Explaining the customers what the consequences are in case problems arise, and how they will be solved.
c) Ensuring that customers are aware of the contribution they can make to the product or service quality.
d) Providing adequate and readily accessible facilities for effective communication to and from the customer.
e) Determining the relationship between the offered product or service and the real needs of the customer.

The perception that the customer has of the company's quality ability (and thus of the quality of the product or service that the company can provide) is often acquired through the effectiveness of the communication between the customer and the company. In essence an effective communication reveals the company's commitment to deliver the product or service to customer's needs and expectations.

2.4 Quality Guidelines for the Development, Supply and Maintenance of Software

2.4.1 ISO-9000-3

The emphasis of this chapter is on the most important clauses of ISO-9000-3. This standard contains the guidelines to facilitate the application of ISO-9001 to organizations developing, supplying and maintaining software. This ISO standard should be read in conjunction with the other standards relevant to the IT-sector (see page 9). In broad terms ISO-9000-3 is structured around the following quality system topics:

- life cycle activities (these activities relate to the various phases of a quality loop);
- supporting activities (these activities support the life cycle activities and are reasonably applicable throughout all phases of the quality loop).

This ISO standard deals primarily with situations where specific software is developed as part of a contract according to customer's specifications. It is intended to provide guidance where a contract (or other type of agreement) between two parties requires the demonstration of the supplier's capability to develop, supply and maintain software products. It describes suggested controls and methods for producing software that meets customer's requirements. This is done primarily by preventing nonconformity at all stages from development through to maintenance.

This ISO standard does not prescribe any specific life cycle model for a software development project. Therefore quality related activities can be planned and implemented with respect to the nature of the life cycle model used.

On the basis of the quality system requirements of this ISO standard, the life cycle of an arbitrary software engineering project will be modeled with SOCCA. To support this modeling activity the most important ISO clauses are described in the subsequent two chapters. The subjects are presented in the same sequence as in ISO-9000-3 and the text may be taken quite literally from this standard, but may be supplemented or condensed by the discretion of the author. This is however necessary to set the scene for the subsequent modeling activities with SOCCA in the remainder part of this paper.

The clauses concerning document control (6.2) and quality planning (5.4) are used for modeling with SOCCA. As a result of this activity these clauses have been revised considerable, since it was necessary to be more precise in making the distinction between the ISO-9001 requirements (which are modeled) and the ISO-9000-3 guidelines.

It should be emphasized that ISO-9000-3 contains the quality guidelines and ISO-9001 contains the actual quality requirements. Therefore it is acceptable to deviate from the ISO-9000-3 clauses, as long as such deviations are within the boundaries of the ISO-9001 requirements.

2.4.2 Life cycle activities

The following topics are taken quite literally from ISO-9000-3, but may be supplemented or condensed by the discretion of the author. For easy reference the ISO-9000-3 chapter numbers are included between brackets.

Contract reviews (5.1):

The supplier should establish and maintain procedures for contract review and for the coordination of these activities. The contract should be reviewed by the supplier to ensure that:

a) The requirements are adequately defined and documented.
b) Possible contingencies and risks are identified.
c) Propriety information is adequately protected.
d) A method is in place to solve any requirements differing from those in the tender.
e) The supplier can meet the contractual requirements. The supplier should also determine that the purchaser can meet the contractual obligations.
f) The supplier’s responsibility with regard to subcontracted work is defined.
g) The terminology is agreed by both parties.

Among other issues, the following are frequently found to be relevant in the contract:

a) Acceptance criteria.
b) Handling of the changes in purchaser’s requirements during development.

c) Handling of problems detected after acceptance including quality related claims and purchaser's complaints.
d) Activities carried out by the purchaser, especially the purchaser's role in requirements specification, installation and acceptance.
e) Facilities and tools to be provided by the purchaser.
f) Standards and procedures to be used.
g) Replication requirements.

Purchaser's requirements specification (5.2):

To continue with software development, the supplier needs a complete, unambiguous set of functional specifications. In addition these requirements should include all aspects necessary to satisfy purchaser's need. These may include, but are not limited to:

- performance;
- safety;
- reliability;
- security;
- privacy;
- Interfaces with other systems.

These requirements should be stated precisely enough to allow validation during product acceptance.

If the specification documentation is not provided by the customer, then the supplier should develop these requirements in close cooperation with the customer and the supplier should obtain customer's approval before entering the development stage. The requirements specification is subject to documentation control and configuration management as part of the development documentation.

The following issues need attention in case the supplier develops the customer specification:

a) Assignment of persons (on supplier and customer sides) who are responsible for establishing the specification.
b) Methods for agreeing on requirements and approving changes.
c) Efforts to prevent misunderstandings, such as definition of terms, explanation of background of requirements.
d) Recording and reviewing discussion results on both sides.

Development planning (5.3):

A development plan should cover the following:

a) A description of the project including a statement of its objectives and references to other related purchaser's or customer's projects.
b) The organization of the project resources including the team structure, responsibilities, use of subcontractors and non-human resources.
c) Project phases.
d) The project schedule, such as the identification of the tasks needed to complete the project, the relationship between these tasks, the time required and the resources needed to carry out these tasks.
e) Identification of related plans, such as:
   - Quality plan;
   - Configuration management plan;
   - Integration plan;
   - Test and acceptance plan.

The development plan should be updated as development progresses.

The development plan should define a methodology for transforming the customer's specification into a software product. This may involve segmenting the work into phases and the identification of:

a) Development phases to be carried out.
b) Required inputs for each phase.
c) Required outputs from each phase.
d) Verification procedures to be carried out at each phase.
e) Analysis of the potential problems associated with the development phases and with the achievement of the specified requirements.

The development plan should define how the project is to be managed including the identification of:

a) Development schedule, implementation and associated deliveries.
b) Progress control.
c) Organizational responsibilities, resources and work assignment.
d) Organizational and technical interfaces between different groups.

The development plan should identify methods for ensuring that all activities are carried out correctly. This may include:

a) Rules, practices and conventions for development.
b) Tools and techniques for development.
c) Configuration management.

Progress reviews should be planned, held and documented to ensure that outstanding resource issues are resolved and to ensure effective execution of development plans.

The required input for each development phase should be defined and documented so that its achievement can be verified. Incomplete, ambiguous or conflicting requirements should be resolved with those responsible for drawing up the requirements.

The required output for each development phase should be defined and documented. The output from each phase should be verified and should:

a) Meet the requirements specified at the start of the development phase.
b) Contain or reference acceptance criteria for forwarding to subsequent phases.
c) Conform to appropriate development practices and conventions whether or not these have been stated in the input information.
d) Identify those characteristics of the product that are crucial to its safe and proper functioning.
e) Conform to applicable regulatory requirements.

The supplier should draw up a plan for verification of all development phase outputs at the end of each phase. Development verification should establish that development phase outputs meet the corresponding input requirements by means of development control measures such as:

a) Holding development reviews at appropriate points in the development phases.
b) Comparing a new design with a proven similar design, if available.
c) Undertaking tests and demonstrations.

The verification results and any further actions required to ensure that the specification requirements are met, should be recorded and checked when the actions are complete. Only verified development outputs should be submitted to configuration management and accepted for subsequent use.

Quality planning (5.4):

The ISO-9000-3 guidelines refer with this article to the ISO-9001 article 'Quality System'. This ISO-9001 article is actually very short and mainly states that the supplier needs to establish and maintain a documented quality system as a means of ensuring that the product conforms to the specified requirements. Establishing a documented quality system involves the following activities:

a) The preparation of documented quality system procedures and instructions in accordance with the requirements of this standard (ISO-9001).
   (This article refers to the same standard. In my opinion it means that the contents of the document must comply with the standard, but as an activity it refers to the document control procedures.)
b) The effective implementation of the documented quality system procedures and instructions.

The ISO-9000-3 guidelines state the same, but is more specific concerning the development, supply and maintenance of software. It states:

The supplier should prepare and document a quality plan to implement quality activities for each software development on the basis of the quality system, and make it understood and observed by the organizations concerned.

The quality plan should be updated along with the progress of the development and items concerned with each phase should be completely defined when starting the phase.

These articles actually covers the earlier mentioned ISO-9001 articles a) and b).

The quality plan should be formally reviewed and agreed by all organizations concerned in its implementation. This article refers to the document control procedures.

The ISO-9000-3 guidelines gives a guidance to which items should be covered in a quality plan and states the following: The quality plan should specify or reference the following items:

a) Quality objectives, expressed in measurable terms whenever possible.
b) Identification of types of test, verification and validation activities to be carried out on product

specifications, plans and test specifications together with the methods and tools to be employed.

c) Defined entry and exit criteria for each development phase.

d) Detailed planning of test verification and validation activities to be carried out, including schedules, resources and approval authorities.

e) Specific responsibilities for quality activities such as
   - Inspections, reviews and tests,
   - Configuration management and change control,
   - Defect control and corrective action.

The document that describes the quality plan may be an independent document (entitled Quality Plan) or a part of another document, or composed of several documents including the development plan.

Design and implementation (5.5):

The design and implementation activities are those which transform the purchaser's requirements specification into an executable software product. Because of the complexity of software products, it is imperative that these activities be carried out in a disciplined manner, in order to produce a product of suitable quality, rather than depending on the test and validation activities for assurance and quality.

In addition to the requirements common to all development phases the following aspects inherent to the design activities should be taken into account:

a) Identification of design considerations.
   Besides the input and output specifications, aspects such as design rules and internal interface definitions should be examined.

b) Design methodology.
   A systematic design methodology appropriate to the type of software product being developed should be used.

c) Use of past design experiences.
   Using lessons learned from past design experiences, the supplier should avoid recurrence of the same or similar problems.

d) Subsequent processes.
   The product should be designed to the extend practical to facilitate testing, maintenance and use.

In addition to the requirements common to all development activities, the following aspects should be considered in each implementation activity:

a) Rules.
   Rules such as programming rules, programming languages, consistent naming conventions, coding and adequate commentary rules should be specified and observed.
b) Implementation methodologies.
   The supplier should use appropriate implementation methods and tools to satisfy purchaser requirements.

The supplier should carry out reviews or inspections to ensure that the requirements are met and the above methods are correctly carried out. The design or implementation process should not proceed until all deficiencies are satisfactorily corrected or the risk of proceeding otherwise is known and accepted.

Testing and validation (5.6):

This article refers to a number of articles from ISO-9001, that is: Design control, Inspection and testing, and Control of nonconforming product. However this article does not cover the mentioned ISO-9001 articles completely, since other articles from the ISO-9000-3 guidelines cover the remaining part. The guidelines state the following:

Testing may be required at several levels from the individual software module, to the integrated system. There are several different approaches to testing and integration.

In some instances validation, field testing and acceptance testing may be the same activity.

Test planning:
The supplier should establish and review the test plans, specifications and procedures before starting testing activities. These should include the following:

a) The integration, system test and acceptance test plans.
b) Test cases, test data, and expected results.
c) Types of tests to be performed, e.g. functional tests, boundary tests, performance tests, use ability tests.
d) Test environment, tools and test software.
e) The criteria on which the completion of the test will be judged.
f) Completeness and accuracy of user documentation.

g) Personnel required for testing and associated training requirements.

These items seem to overlap with some of the items mentioned earlier under ‘Quality planning’. However it should be noted that a quality plan may be an independent document, or part of another document, or composed of several documents. A test plan is part of a quality plan. If the test plan is a small document then it may be incorporated in the quality plan, otherwise the quality plan should refer to the documents describing the test plan.

Testing:
Special attention should be paid to the following aspects of testing:

a) The test results should be recorded to the level defined in the relevant specification.
b) Any discovered problems and their possible impacts to any other parts of the software should be noted and those responsible notified so the problems can be tracked until they are solved.
c) Areas impacted by any modifications should be identified and re-tested.
d) Test adequacy and relevancy should be evaluated.
e) The hardware and software configuration during test should be considered.

Validation:
Before offering the product for delivery and purchaser acceptance, the supplier should validate its operation as a complete product (including deliverable documentation) and under conditions similar to the application environment as specified in the contract.

Field testing:
Where testing under field conditions is required, the following concerns should be addressed:

a) The features to be tested in the field environment.
b) The specific responsibilities of the supplier and purchaser for carrying out and evaluating the test.
c) Restoration of the user environment (after test).

Acceptance (5.7):

When the supplier is ready to deliver the validated product, the purchaser should judge whether the product is acceptable

according to previous agreed criteria and in a manner specified in the contract.

The method of handling problems detected during the acceptance procedure and their disposition should be agreed between the purchaser and supplier and documented.

Before carrying out acceptance activities the supplier should assist the purchaser to identify the following:

a) Time schedule
b) Procedures for evaluation
c) Software/hardware environments and resources
d) Acceptance criteria

Replication is a step which should be conducted prior to delivery. The following should be considered:

a) Number of copies of each software item to be delivered.
b) Type of media for each software item, including format and version in human readable form.
c) The stipulation of required documentation (training manuals, user guides, etc.)
d) Copyright and licensing concerns addressed and agreed to.
e) Custody of master and backup copies where applicable, including disaster recovery plans.
f) Period of obligation of supplier to supply copies.

The supplier should deliver as specified in the contract. Provisions should be made for verifying the correctness and completeness of copies of the software product delivered.

The roles, responsibilities and obligations of the supplier and purchaser should be clearly established taking into account the following:

a) Schedule, including off hour and weekends.
b) Access to purchaser’s facilities (security badges, passwords, escorts);
c) Availability of skilled personnel.
d) Availability and access to purchaser’s systems and equipment.
e) The need for validation as part of each installation should be determined contractually.
f) A formal procedure for approval of each installation upon completion.

Maintenance (5.8):

When maintenance of the software product is required by the purchaser, after initial delivery and installation, this should be stipulated in the contract. The supplier should establish and maintain procedures for performing maintenance activities and verifying that such activities meet the specified requirements for maintenance.

Maintenance activities for software products are typically classified into the following:

a) Problem resolution.
b) Interface modification.
c) Functional expansion or performance improvements.

The items to be maintained, and the period of time for which they should be maintained, should be specified in the contract. The following are examples of such items:

a) Programs
b) Data and their structures.
c) Specifications
d) Documents for purchaser and/or user.
e) Documents for supplier’s use.

All maintenance activities should be carried out and managed according to a maintenance plan defined and agreed beforehand by the supplier and purchaser. The plan should include the following:

a) Scope of maintenance
b) Identification of the initial status of the product.
c) Support organization
d) Maintenance activities
e) Maintenance records and reports.

The initial status of the product to be maintained at the beginning of the maintenance stage should be defined, documented and agreed to by both supplier and purchaser.

It may be necessary to establish an organization, with representatives from both supplier and purchaser to support the maintenance activities. Since activities in the maintenance stage can not always be carried out on a scheduled basis, this organization should be flexible enough to cope with unexpected occurrences of problems. It may also be necessary to identify facilities and resources to be used for the maintenance activities.

All changes to the software (for reasons of problem resolution, interface modifications, functional expansions or performance improvement) carried out during maintenance should be made in accordance with the same procedures, as far as possible, used for development of the software product. All such changes should also be documented according to the procedures for document control and configuration management.

Problem resolution involves the detection, analysis and correction of software faults causing operational problems. When resolving problems, temporary fixes may be used to minimize downtime and permanent modifications carried out later.

Interface modifications may be required when additions or changes are made to the hardware system, or components controlled by the software.

Functional expansion or performance improvement of existing functions may be required by the purchaser in the maintenance stage.

All maintenance activities should be recorded in pre-defined formats and retained. Rules for the submission of maintenance reports should be established and agreed upon by the supplier and purchaser.

The maintenance records should include the following items for each software item being maintained:

a) List of requests for assistance or problem reports that have been received and the current status of each.
b) Organization responsible for responding to requests for assistance or implementing the appropriate corrective actions.
c) Priorities that have been assigned to the corrective actions.
d) Results of the corrective actions.
e) Statistical data on fault occurrences and maintenance activities.

The record of maintenance activities may be used for evaluation and enhancement of the software product and for improvement of the quality system itself.

The supplier and purchaser should agree and document procedures for incorporating changes in a software product resulting from the need to maintain performance.

These procedures should include:

a) Ground rules to determine where localized “patches” may be incorporated or release of a complete updated copy of the software product is necessary.
b) Descriptions of the types (or classes) of releases depending on their frequency and/or impact on the purchaser’s operations and ability to implement changes at any point in time.
c) Methods by which the purchaser will be advised of current or planned future changes.
d) Methods to confirm that changes implemented will not introduce other problems.
e) Requirements for records indicating which changes have been implemented and at what locations for multiple products or sites.

2.4.3 Supporting activities

The following topics are taken quite literally from ISO-9000-3, but may be supplemented or condensed by the discretion of the author. For easy reference the ISO-9000-3 chapter numbers are included between brackets.

Configuration management (6.1):

Configuration management provides a mechanism for identifying, controlling and tracking the official versions of each software item. In many cases earlier versions still in use must also be maintained and controlled.

The configuration management system should:

a) Uniquely identify the official versions of each software item.
b) Identify the versions of each software item which together constitute a specific version of a complete product.
c) Identify the build status of software products in development or delivered and installed.
d) Control simultaneous updating of a given software item by more than one programmer.
e) Provide coordination for updating of multiple products in one or more locations as required.

f) Identify and track each change request from suggestions through release.

The supplier should develop and implement a configuration management plan which includes the following:

a) The organizations involved in configuration management and responsibilities assigned to each of them.
b) Configuration management activities to be carried out.
c) Configuration management tools, techniques and methodologies to be used.

The supplier should establish and maintain procedures for identifying software items during all phases starting from specification through development, replication and delivery. Where required by contract these procedures may also apply after delivery of the product. each individual software item should have a unique identification.

Procedures should be applied to ensure that the following can be identified for each version of a software item:

a) The functional and technical specifications.
b) All development tools which affect the functional and technical specifications.
c) All interfaces to other software items and to hardware.
d) All documents and computer files related to the software item.

The identification of a software item should be handled in such a way that the relationship between the item and the contract requirements can be demonstrated.

For released products there should be procedures to facilitate traceability of the software item or product.

The supplier should establish and maintain procedures to identify, document, review and authorize any changes to the software items under configuration management. All changes to software items should be carried out according to these procedures.

Before a change is made official, its validity should be confirmed and the effects on other items should be identified and thoroughly examined.

Methods to notify the changes to those concerned and to show the traceability between changes and modified parts of software items should be provided.

The supplier should establish and maintain procedures to record, manage and report on the status of software items, of change requests and of the implementation of approved changes.

Document control (6.2):

The ISO-9001 standard state the following requirements:

The supplier should establish and maintain procedures to control all documents that relate to the ISO requirements. This covers:

a) The determination of those documents which should be subject to the document control procedures.
b) The document approval and issuing procedures.
c) The document change procedures including withdrawal of obsolete documents and, as appropriate, new releases.

All documents should be reviewed and approved by authorized personnel prior to issue. Procedures should exist to ensure that:

a) The pertinent issues of appropriate documents are available at appropriate locations where operations essential to the effective functioning of the quality system are performed.
b) Obsolete documents are promptly removed from appropriate points of issue or use.

Changes to documents should be reviewed and approved by the same functions/organizations that performed the original review and approval unless specifically designated otherwise. The designated organizations should have access to pertinent background information upon which to base their review and approval.

Where practical, the nature of the change should be identified in the document or the appropriate attachments. A master list or equivalent document control procedure should be established to identify the current version of documents to preclude the use of non-applicable documents.

Documents should be reissued after a practical number of changes have been made.

In addition to the ISO-9001 requirements the ISO-9000-3 guidelines include the following:

The document control procedures should be applied to relevant documents including:

a) Procedural documents describing the quality system to be applied in the software life-cycle.
b) Planning documents describing the planning and progress of all supplier activities and his interactions with the purchaser.
c) Product documents describing a particular software product, including:
   - Development phase input
   - Development phase outputs
   - Verification plans and results
   - Documentation for purchaser and user
   - Maintenance documentation

Where use is made of computer files special attention should be paid to appropriate approval, distribution and archiving procedures.

Quality records (6.3):

The supplier should establish and maintain procedures for identification, collection, indexing, filing, storage, maintenance and disposition of quality records.

Quality records should be maintained to demonstrate achievement of the required quality and the effective operation of the quality system. Pertinent subcontractor quality records should be an element of these data.

All quality records should be legible and identifiable to the product involved. Quality records should be stored and maintained in such a way that they are readily retrievable in facilities that provide a suitable environment to minimize deterioration or damage and prevent loss. Retention times of quality records should be established and recorded. When agreed contractually, quality records should be made available for evaluation by the purchaser or his representative for an agreed period.

Measurements (6.4):

There are currently no universally accepted measures of software quality. However, at a minimum, some metrics should be used which represents reported field failures or defects from the customer's viewpoint. Selected metrics should be described such that the results are comparable.

The supplier of software products should collect and act on quantitative measures of the quality of these software products. These measurements should be used for the following purposes:

a) To collect data and report metric values on a regular basis.
b) To identify the current level of performance on each metric.
c) To take remedial action if metric levels grow worse or exceed established target levels.
d) To establish specific improvement goals in terms of the metrics.

Metrics should be reported and used to manage the development and delivery process and should be relevant to the particular software product or application.

The supplier should have quantitative measures of the quality of the development and delivery process. These metrics should reflect:

a) How well the development process is being carried out in terms of milestones and in-process quality objectives being met on schedule.
b) How effective the development process is at reducing the probability that faults are introduced or that any faults introduced go undetected.

Here, as for product metrics the important thing is that levels are known and used for process control and improvement and not what specific metrics are used. The choice of metrics should fit the process being used and if possible have a direct impact on the quality of the delivered software. Different metrics may be appropriate for different software products produced by the same supplier.

Rules, practices and conventions (6.5):

The supplier should provide rules, practices and conventions to make the quality system defined in the ISO-9000-3 guideline effective. The supplier should review these rules, practices and conventions and revise them as required.

Tools and techniques (6.6):

The supplier should provide software tools, facilities and techniques to make the quality system specified in the ISO-9000-3 guideline effective. These tools, facilities and techniques can be effective for management purposes as well as for product development. The supplier should improve these tools and techniques as required.

Purchasing (6.7):

The supplier should ensure that purchased product conforms to specified requirements.

Purchasing documents should contain data clearly describing the product ordered. The supplier should review and approve purchasing documents for adequacy of specified requirements prior to release.

Note: A purchased product may be a software and/or hardware item intended for inclusion in the required end product or a tool intended to assist in the development of the required product.

The supplier should select subcontractors on the basis of their ability to meet subcontract requirements, including quality requirements. The supplier should establish and maintain records of acceptable subcontractors.

The selection of subcontractors and the type and extent of control exercised by the supplier, will be dependent upon the type of product and where appropriate, on records of the subcontractor’s previously demonstrated capability of performance.

The supplier should ensure that the quality system controls are effective.

The supplier is responsible for the validation of subcontracted work. This may require the supplier to conduct design and other reviews in line with the supplier's own quality system and if so, such requirements should be included in the subcontract. Any requirements for acceptance testing of the subcontracted work by the supplier should be similarly included.

Where specified in the contract the purchaser or his representative should be afforded the right to determine at source, or upon receipt, that purchased product conforms to specified requirements. Validation by the purchaser may not absolve the supplier of the responsibility to provide acceptable product nor may it preclude subsequent rejection.

When the purchaser or his representative elects to carry out validation at the subcontractor premises, such validation should not be used by the supplier as evidence of effective control of quality by the subcontractor.

Included software product (6.8):

The supplier may be required to include or use software product supplied by the purchaser or by a third party. The supplier should establish and maintain procedures for validation, storage, protection and maintenance of such product. Consideration should be given to the support of such software product in any maintenance agreement related to the product to be delivered.

Purchaser supplied product that is found unsuitable for use should be recorded and reported to the purchaser. Validation by the supplier does not absolve the purchaser of the responsibility to provide acceptable product.

Training (6.9):

The supplier should establish and maintain procedures for identifying the training needs and provide for the training of all personnel performing activities affecting quality. Personnel performing specific assigned tasks should be qualified on the basis of appropriate education, training and/or experience, as required.

The subjects to be addressed should be determined considering the specific tools, techniques, methodologies and computer resources to be used in the development and management of the software product. It might also be required to include the training of skills and knowledge of the specific field the software is to deal with.

Appropriate records of training/experience should be maintained.

2.5 Conclusion

The ISO-900x standards contain the requirements of a quality system. These standards apply to the life cycle phases of a product or service (quality loop) and deal with establishing a quality system, ensuring that quality is maintained (quality assurance) by quality control procedures. Especially the quality control procedures receive much attention.

The key objectives of the ISO quality requirements are that the quality system is beneficial to:

- the supplier (business results);
- the customer (customer's needs and expectations).

The ISO quality standards require that the business processes are well described and adhered to and that the results from these processes are appropriately documented. Its belief is that a quality organization delivers products or provides services that meet customer's expectations. Its prime focus is on preventing nonconformity through all stages until delivery of the product and servicing.

These standards do not address issues like:

- influence of manufacturer or service provider on society;
- business constraints (e.g. legislation);
- personnel satisfaction;
- any methodology to acquire or improve a quality system or any part of a quality system;
- metrics to measure the quality of the delivered products or services;

3 Modeling ISO-9000

3.1 Introduction

As explained in the previous chapter, the ISO-9000 standards state and explain the requirements of a quality system. However these standards do not give a model for a quality system. In this chapter a number of models will be designed that support some of the ISO-9000 quality requirements. In this context a model is a coherent set of drawings (figures) that describe a single or at most a few ISO requirements, but which are very closely related to one another. In other words each single model describes the data perspective, the behavior and communication perspective, and the process perspective of the ISO requirement(s) under review. Each model will be discussed in a separate subchapter.

As mentioned in the previous chapter, an effective quality management system satisfies customer's needs and expectations, and protects the company's interests. With this statement, the following questions arise:

- Who are the customers and what are their needs and expectations?
- What are the company's interests?

It is obvious that the answers to the above questions depend on the type of company considered and as a result the appropriate quality system for such a company will differ. To design an appropriate quality system for the IT sector (but limited to software engineering projects only), the companies will be classified into the following categories:

a) Software companies who's core business is the production of commercial software packages.
b) Software companies who's core business is software engineering projects for identified customers. This includes companies who supply basic packages and tailor them to customer's needs.
c) Companies who's core business is not in the IT sector, but have an IT department to support their business. The IT department may either develop their own packages, or interact with companies of type a) or b) above.

All the above companies can design one or more quality loops and the corresponding life-cycle activities for their software engineering projects as is appropriate for their

business. The identified and relevant life cycle activities may be implemented quite differently depending on the development model chosen. For instance the appropriate development model is quite different in the following types of software engineering projects:

- Development of commercial packages, which involves market research and delivery within certain time limits to beat the competition.
- Client / Server development, which involves development on different and geographical spread platforms and integrating them into a coherent system.
- Implementation of package solutions, which involves the identification of the business needs, evaluation of commercial packages against these business needs, and the possible additional required development in case packages do not satisfy all requirements.

In the following chapters some quality related activities will be modeled with SOCCA. These activities will be modeled according to the quality requirements of ISO-9001. The aim is to implement a general model that can be applied to all the above identified companies. A further customization of these models can be applied and which are more appropriate to the type of company considered.

In the following chapters some of the ISO-9001 requirements are modeled. In a previous chapter the quality guidelines of ISO-9000-3 (Quality guidelines for the development, supply and maintenance of software) were described. During the modeling activities it was necessary to make a sharper distinction between the ISO-9001 requirements and the ISO-9000-3 guidelines. This resulted in some revisions on the previous chapter to highlight the distinction between the ISO guidelines and requirements more precisely.

3.2 Document Control

3.2.1 Introduction

During the life cycle of an arbitrary software engineering project many documents are produced. In short ISO-9001 states the following requirements concerning document control:

a) The documents shall be reviewed and approved for adequacy by authorized personnel. The same also applies to document revisions.
b) The pertinent issues of the appropriate documents are available at all locations which need these documents for their business.
c) Obsolete documents are promptly removed from all points of issue.
d) A master list or equivalent document control procedure shall be established to identify the current revision of documents in order to preclude the use of non-applicable documents.
e) Where applicable the nature of the change shall be identified in the document or the appropriate attachments. Documents shall be re-issued after a practical number of changes have been made.

ISO-9001 only states that the above requirements are applicable to documents and data that relate to the requirements of the ISO standard. It does not explicitly state which documents are subject to these requirements. ISO-9000-3 (the guidelines) states that these requirements should be applied to the following document types:

- Procedure documents.
- Project execution plans.
- Input and output documents of all project phases. Depending on the type of project different types of documents may be applicable. For instance integrating different software packages into a coherent system requires a different approach compared to developing a real-time process control system.
- Contract documents. Again depending on the type of project many different types of contract documents may exist. For instance the supplier has a contract with the customer, but at the same time may have some parts of the project contracted out to other suppliers (in which case the main supplier is also a customer).

Apart from the above document types, other types of documents may be produced during a software engineering project, but which are not subject to the ISO-9001 requirements. However during the project it may still be necessary to have some form of document control for these documents, like registering the issue of these documents and maintaining a distribution list. Requirements like document approval or version control may not be applicable to these documents. Examples of these documents are:

- Minutes of meeting.
- Progress reports.

3.2.2 Class diagrams

At the start of the project the types of documents that will be produced need to be identified. Concerning document control the relevant document classes, relationships, attributes and their operations are shown in the figures **6**, **7** and **8**. These classes serve as superclasses.

All document types can be attached as subclasses to either ISO_Document or Non_ISO_Document (for instance the example class Design_Document is a subclass of ISO_Document). If a document does not contain any additional relevant characteristics compared to these classes, then the document

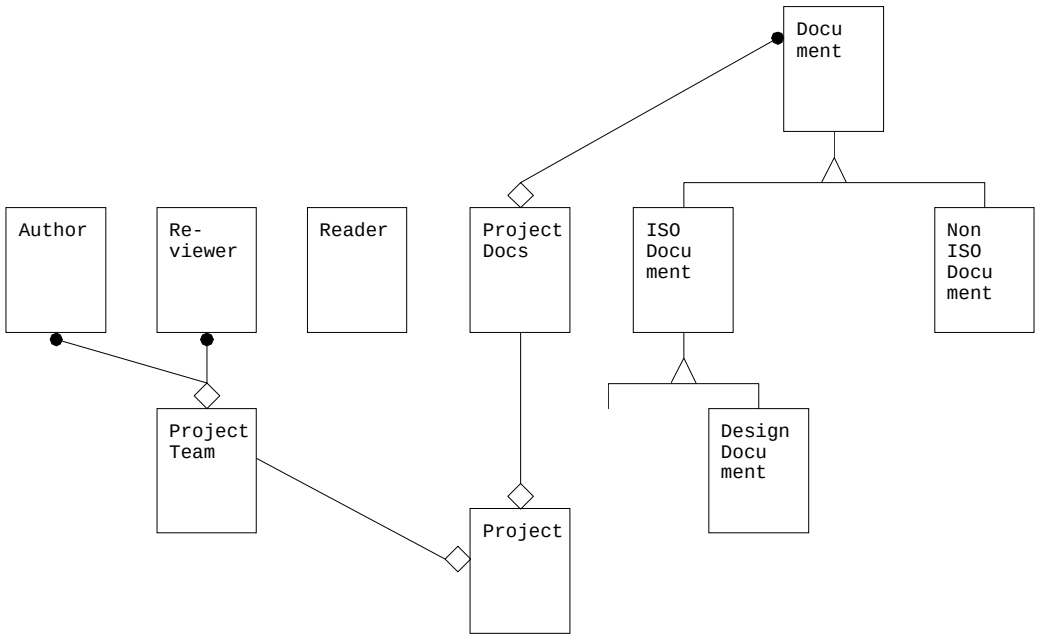

Figure 6 Class diagram: document classes and is-a and part-of relationships

Document	ISO_Document	Non_ISO_Document	Project_Docs
fDocumentId fDocumentTitle fAuthor fIssueDate fContent	fReviewedBy[1..n] fReviewDate[1..n] fRevcomment[1..n] fApprovedBy fApprovalDate fVersionNumber fStatus	fCustomizedAttr.	
create_first create_next open_modify modify close_modify issue read	withdraw review review_ok review_not_ok close_review reset_review approve do_not_approve		create_version

Author	Reviewer	Reader	
modify_doc withdraw_doc	review_doc approve_doc		

Figure 7 Class diagram: document attributes and operations

can be defined as an instance of either the subclass ISO_Document or Non_ISO_Document. The subclass Non_ISO_Document does not contain any additional specific operations or attributes (since non ISO requirements are outside the scope of this master thesis). Therefore this class is redundant and any instance of a ‘non ISO-document’ can also be defined as an instance of its superclass Document. In the context of this paper the subclass Non_ISO_Document is used as a separate class to highlight the distinction between ‘ISO-documents’ and ‘non ISO-documents’.

The classes Author, Reviewer, and Reader represent roles people have in a project with regard to documents. The class Reader (or user of a document) does not have any ‘is-a’ or ‘part-of’ relationships with other classes in figure **6**, because a ‘reader’ is not necessarily part of a project team. Because of the ISO requirement which states that obsolete documents must be promptly removed from all points of issue, it is important to include the class Reader in the model.

The class ISO_Document contains the array attributes fReviewedBy[1..n], fReviewDate[1..n] and fRevComment[1..n].

The ISO-9001 requirements state that the document must be reviewed and authorized for adequacy by authorized personnel. It is the company's responsibility to determine how the document is reviewed. It is assumed that the document may also be reviewed by more people (for instance by a review team). Each reviewer can enter his or her name, review date and comments. Depending on the review results the document is either approved or rejected. In case the document is approved the name of the approver and the approval date is entered.

The attribute fStatus with possible values not available, readable or obsolete in the class ISO_Document serves to indicate the status of the document from a reader's perspective. Until the document is issued the document is not available, after the document is issued it is readable and it becomes obsolete when the document is withdrawn.

Note that the document classes do not apply to paper documents only, but also apply to electronic data (such as software code on an electronic medium).

So far the above discussed classes only contain the static data, so information can be retrieved about the status of the active project documents and their current version number. The information contained in these classes comply with the ISO requirements, but may be extended as a company sees fit for its purpose. In addition the dynamic behavior needs to be modeled. The import/export operations (uses relations) and the import list are shown in the figures **9** and **10**. It is important to note that author and reviewer represent roles, which can be called by some other class (for instance a projectmanager). This class is not shown here. Furthermore reader acts as an independent class and may read any document available to him.

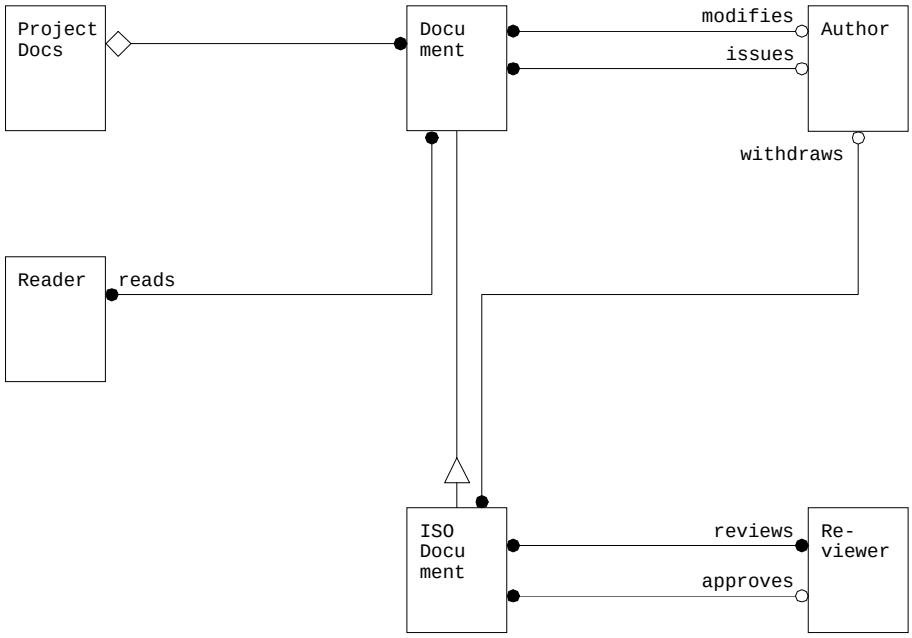

Figure 8 Class diagram: classes and general relationships

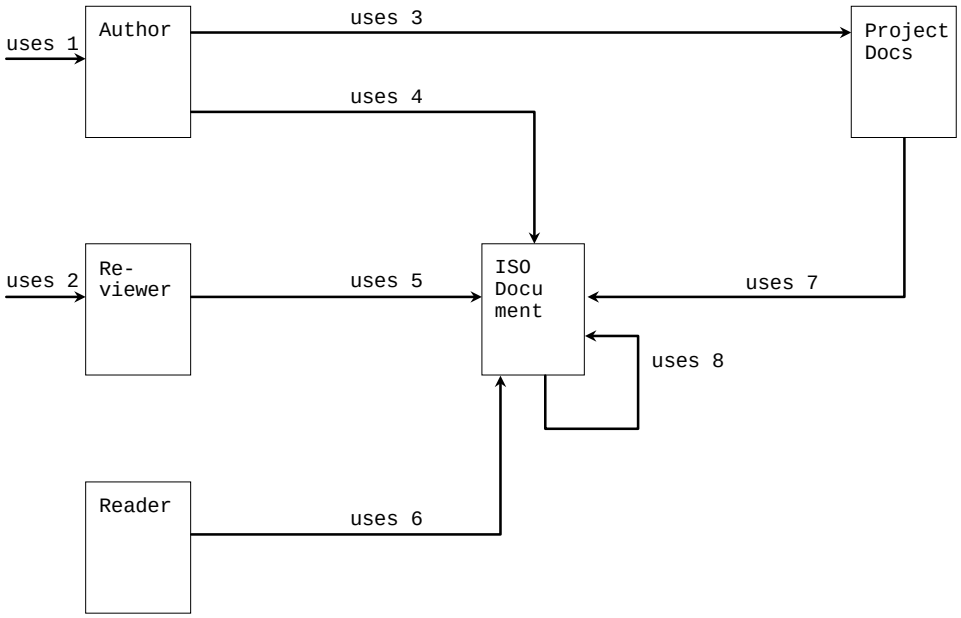

Figure 9 Import/export diagram

```
uses 1
  modify_doc
  withdraw_doc
uses 2
  review_doc
  approve_doc
uses 3
  create_version
uses 4
  open_modify
  modify
  close_modify
  issue
  withdraw
uses 5
  review
  review_ok
  review_not_ok
  approve
  do_not_approve
uses 6
  read
uses 7
  create_first
  create_next
uses 8
  close_review
  reset_review
```

Figure 10 Import list

3.2.3 STD of the external behavior

The STD's of the external behaviors are shown in the subsequent figures.

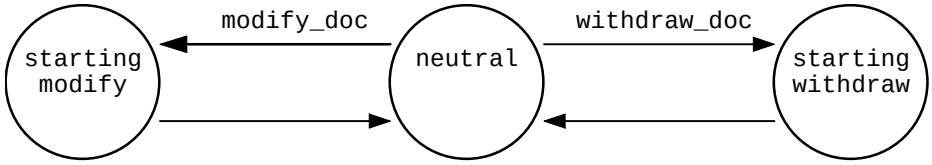

Figure 11 Author; STD of the external behavior

Besides the export operation modify_doc the operation withdraw_doc is explicitly modeled. The assumption is made that two or more version of the same document may coexist (for instance two program versions may be operational and therefore both versions are valid). The decision whether a document version (or any version of a specific document) is obsolete is made outside the modeled classes. Therefore the export operation withdraw_doc needs to be explicitly modeled.

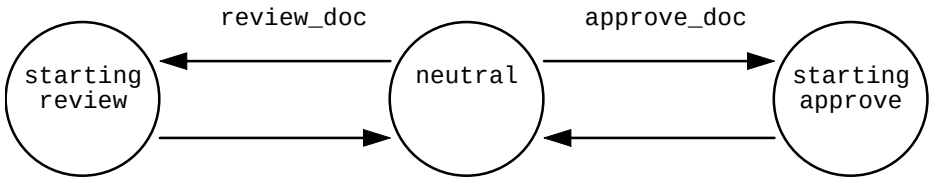

Figure 12 Reviewer; STD of the external behavior

ISO explicitly states that documents must be reviewed and approved by authorized personnel. However ISO does not state how a document should be reviewed, nor that the reviewer and approver are the same person or different persons. In the model the following review options are considered:

- Review by a single person or by a review team acting as a single body (for instance a fagan inspection).
- Review by many people, who review the document in sequence of one another or simultaneously (for instance the document is circulated for comments).

Only one person approves or rejects the document based on the results of the reviews. If the review is OK, then the

document is approved, otherwise it is up to the discretion of the approver to approve or reject the document. The reviewer and approver may be the same person. The model allows that the reviewer only reviews, approves, or reviews and approves a document.

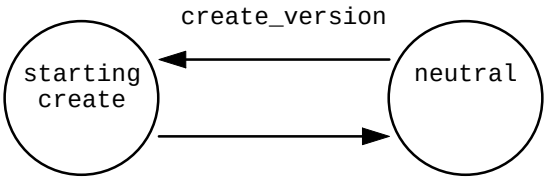

Figure 13 Project_Docs; STD of the external behavior

Because Project_Docs is a container class of all documents, only this class is able to create documents with valid version numbers.

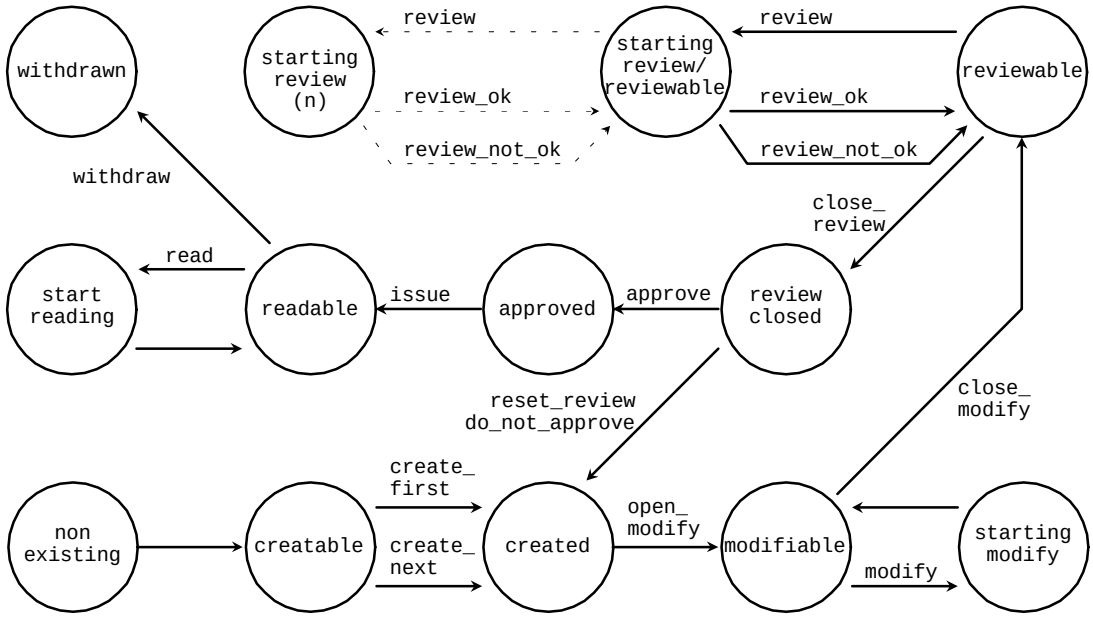

Figure 14 ISO_Document; STD of the external behavior

The STD of the external behavior of ISO_Document in figure **14** is more complicated and needs further explanation. When the document is created, it can be modified (open_modify, modify, close_modify). Thereafter the document can be reviewed by any number of reviewers (1 . . n) and the results are reported by x OK's and y not-OK's, where x + y = n (review_ok, review_not_ok). End review is activated by the operation close_review in the class ISO_Document. It signals the end of the review by the following condition:

- The number (n) of required reviewers is pre-determined. When
  x OK's + y not-OK's = n
  are received the operation close_review is activated and the review ends.

- An alternative condition is a time-out signal:
  The operation close_review is activated when the allowable review time has passed. However the review can only be ended in the state reviewable.
  In case the review must be ended after the allowable review time has elapsed, irrespective whether the document is currently under review, than the STD of ISO_Document needs some modification. All transitions review_ok and review_not_ok need to be connected from all states starting_review to reviewable. The number of OK's and not-OK's no longer serve as a counter to determine whether the review may be ended, but are only required to determine whether the document may be approved or should be rejected.

Once the review ends, the document can either be approved or rejected, depending on the review results. Each reviewer entered his comments in the ISO_document attribute fRevComment. The approval decision can be based on these comments or the number of review_not_ok's and review_ok's. After the document is approved it will be issued and read (issue, read). When the document becomes obsolete, it must be withdrawn (withdraw).

3.2.4 STD of the internal behavior

The succeeding figures show the STD of the following internal behaviors:

- modify_doc;
- review_doc;
- approve_doc;
- withdraw_doc;
- ... (w.r.t. call read, which is explained later);
- create_version.

It appears that only these internal behaviors are relevant in the sense that they cover all the operations discussed in the external behaviors, as far as they are calling them from within the corresponding classes. The exception is read. It is called from some other external behavior, which is not further modeled. Only part of the internal behavior of read is modeled, since its trap process is used by ISO_Document.

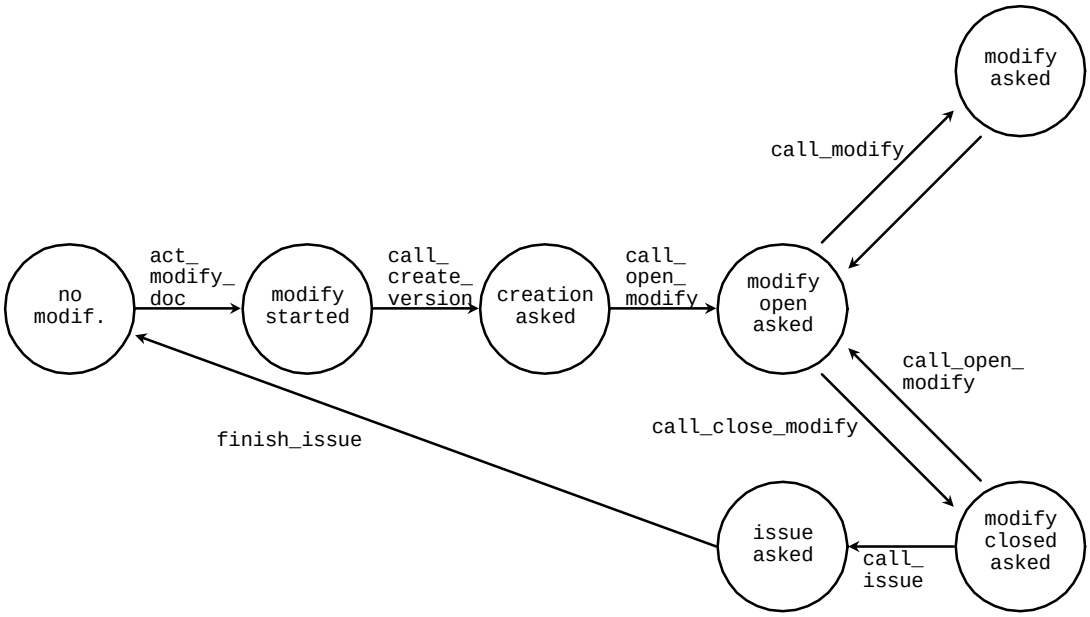

Figure 15 int_modify_doc; STD of the internal behavior

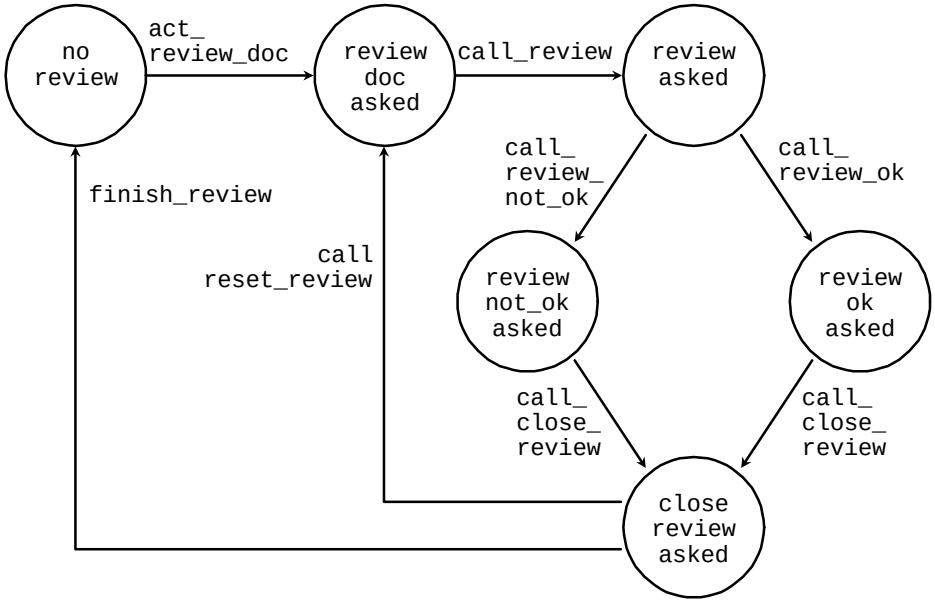

Figure 16 int_review_doc; STD of the internal behavior

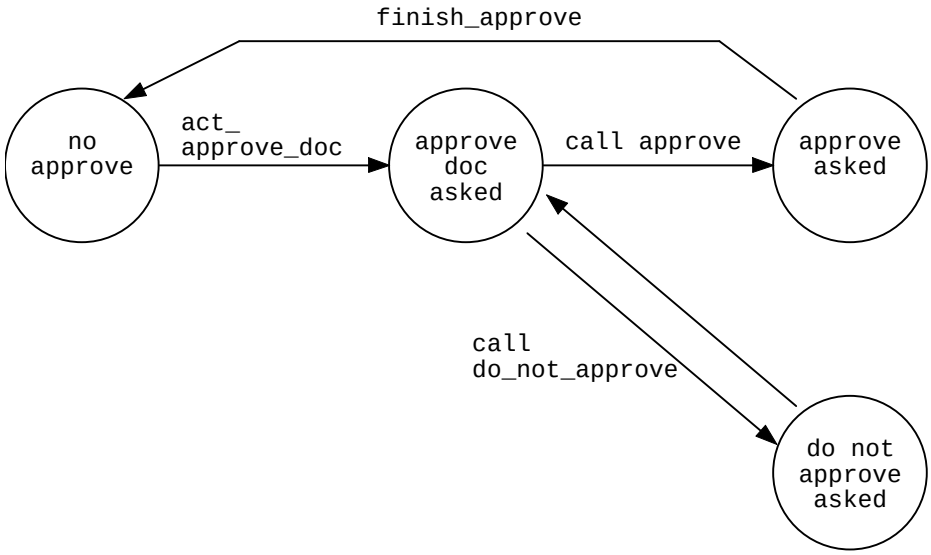

Figure 17 int_approve_doc; STD of the internal behavior

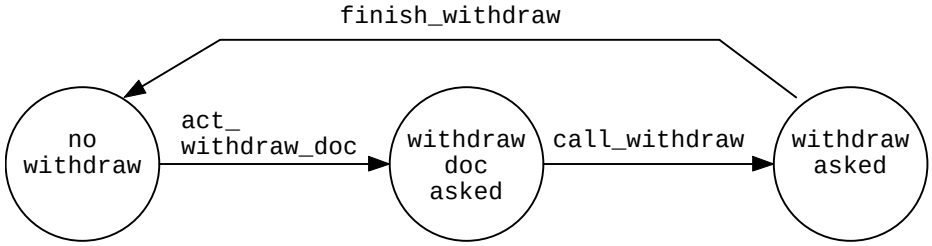

Figure 18 int_withdraw_doc; STD of the internal behavior

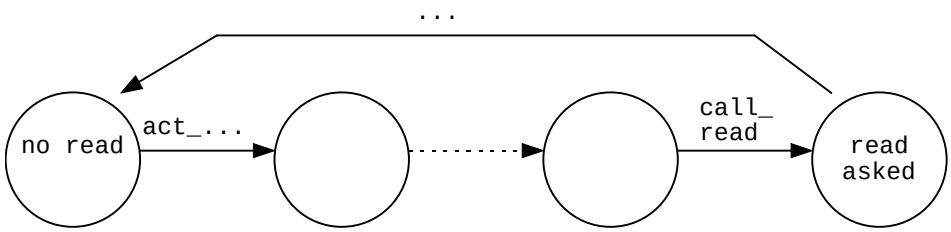

Figure 19 int_. . .; STD of the internal behavior

The internal behavior of int_... is not completely modeled, because the only important operation regarding document control is call_read. ISO-9001 requires that obsolete documents are withdrawn and thus cannot be accessed by readers.

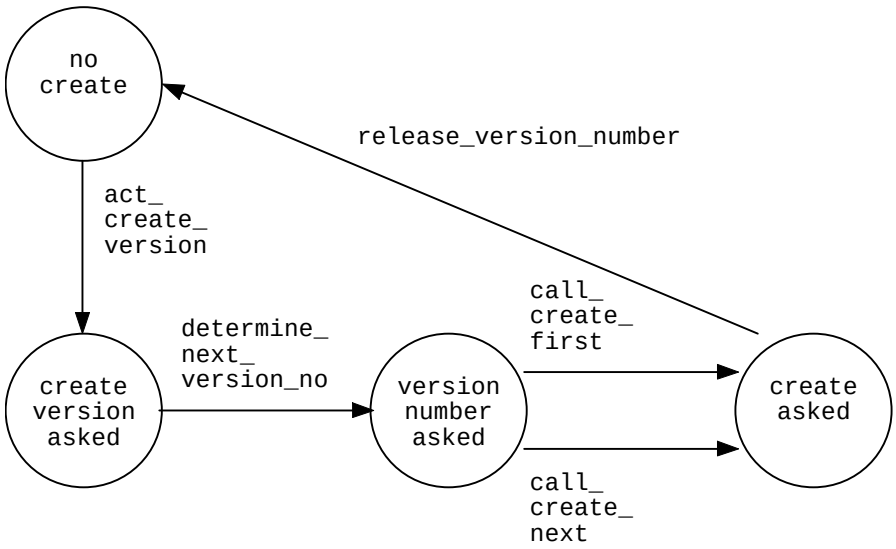

Figure 20 int_create_version; STD of the internal behavior

Each STD of the external and internal behaviors is a strictly sequential operation. Together all STD’s are parallel. The coordination of the parallel operations will be modeled with Paradigm. The STD’s of the external behaviors will be used for the manager processes and the STD’s of the internal behaviors will be used to model the employee processes. So the latter category consists of STD’s for which subprocesses and traps are to be indicated.

3.2.5 Subprocesses and traps

In this chapter Paradigm will be used to model the coordination of the parallel processes. The previously modeled STD's of the external behaviors become the manager processes of the also previously modeled STD's of the internal behaviors. The internal behaviors however now contain the subprocesses and traps concerning their manager process.

The subprocesses and traps are indicated with S-# and t-# (# is a number). The manager processes show the possible states and transitions (a transition is only possible when a subprocess has entered its trap).

The following manager processes are modeled:

- Author
- Reviewer
- Project_Docs
- ISO_Document

Author

Author is modeled as a role class, so it can be used by many other classes (for instance one of the tasks of a designer is writing a document and wherein he will also assume the role of author). Obviously an author must be able to modify a document, which is activated by modify_doc. ISO-9001 explicitly states that an obsolete document must be withdrawn and which is activated by withdraw_doc.

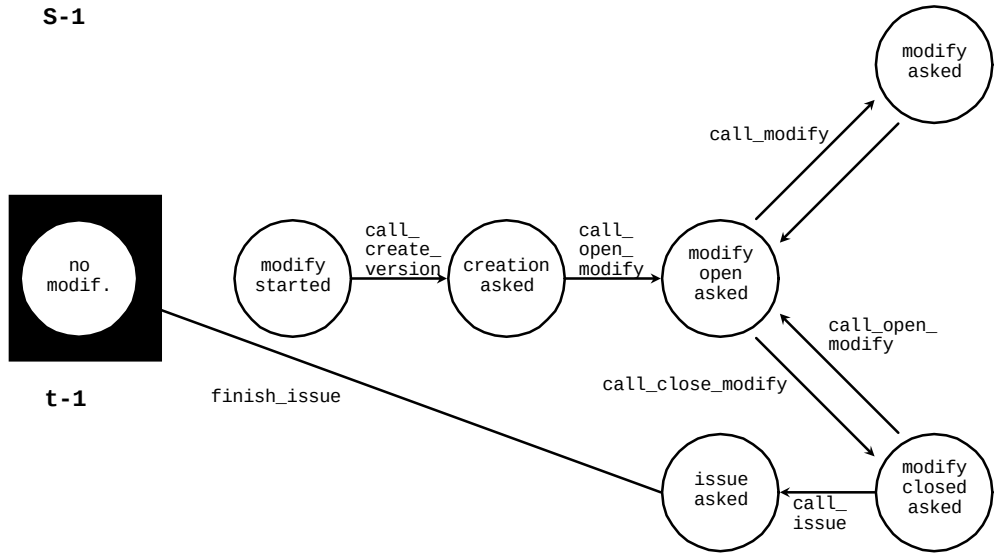

Figure 21 int_modify_doc's subprocesses and traps w.r.t. Author

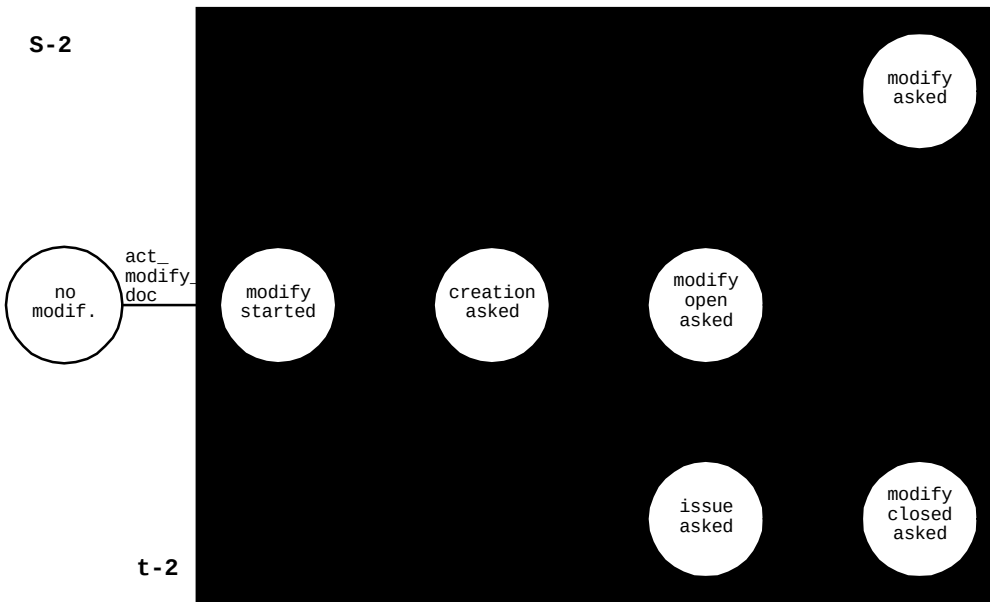

Figure 22 int_modify_doc's subprocesses and traps w.r.t. Author

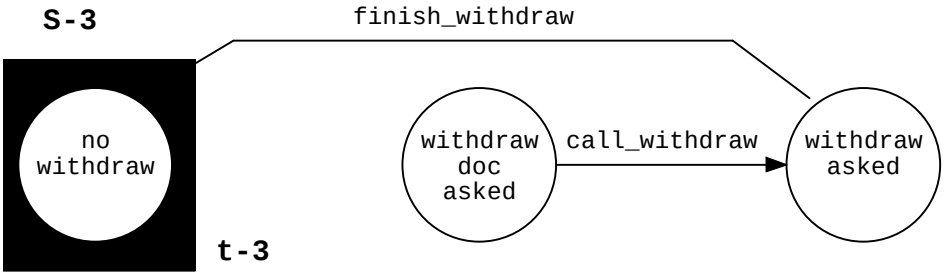

Figure 23 int_withdraw_doc’s subprocesses and traps w.r.t. Author

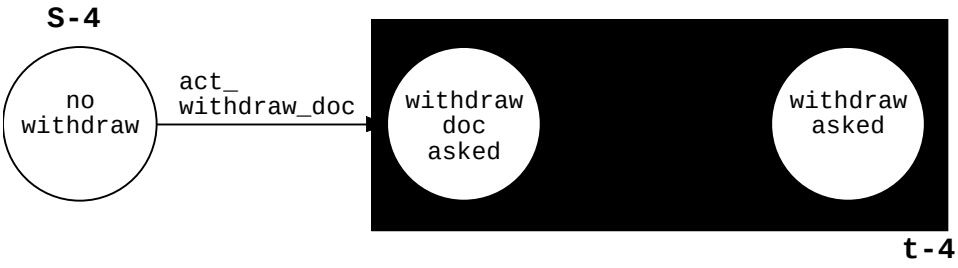

Figure 24 int_withdraw_doc’s subprocesses and traps w.r.t. Author

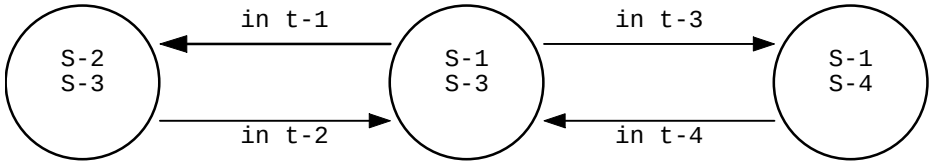

Figure 25 Author; manager of int_modify_doc and int_withdraw_doc

Author is the manager process of the employee processes int_modify_doc and int_withdraw_doc. Both employee processes are the internal behaviors of the same object the manager belongs to. The state {S-1, S-3} is the neutral state of author and as soon as the employee processes have entered the traps t-1 or t-3 a transition is possible to either state {S-2,S-3} or {S-1,S-4} (modify_doc or withdraw_doc). As soon as the employee processes enter the traps t-2 or t-4 the manager returns to its neutral state.

Reviewer

Identical to author, reviewer is modeled as a role class, so it can be used by many other classes. ISO-9001 requires that a document should be reviewed by authorized personnel before it is issued. It is up to the organization to determine who is authorized to review and approve a document. It is very likely that depending on the document type (for instance a design document or a quality plan document) the authorized people are not the same. A document can be reviewed by one or many persons (depending on the document type and the quality procedures of the organization). Depending on the outcome of the review the document may be approved or rejected. Approval will always be done by one person and depending on the quality procedures of the organization the approver may or may not take part in the review process.

The employee processes of reviewer are int_review_doc and int_approve_doc.

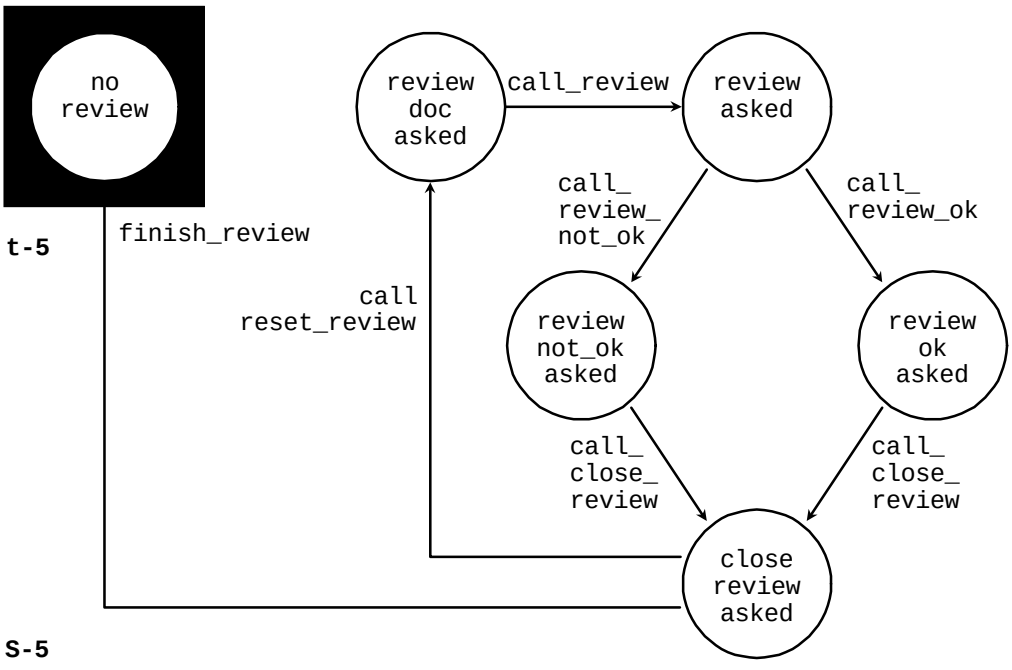

Figure 26 int_review_doc's subprocesses and traps w.r.t. reviewer

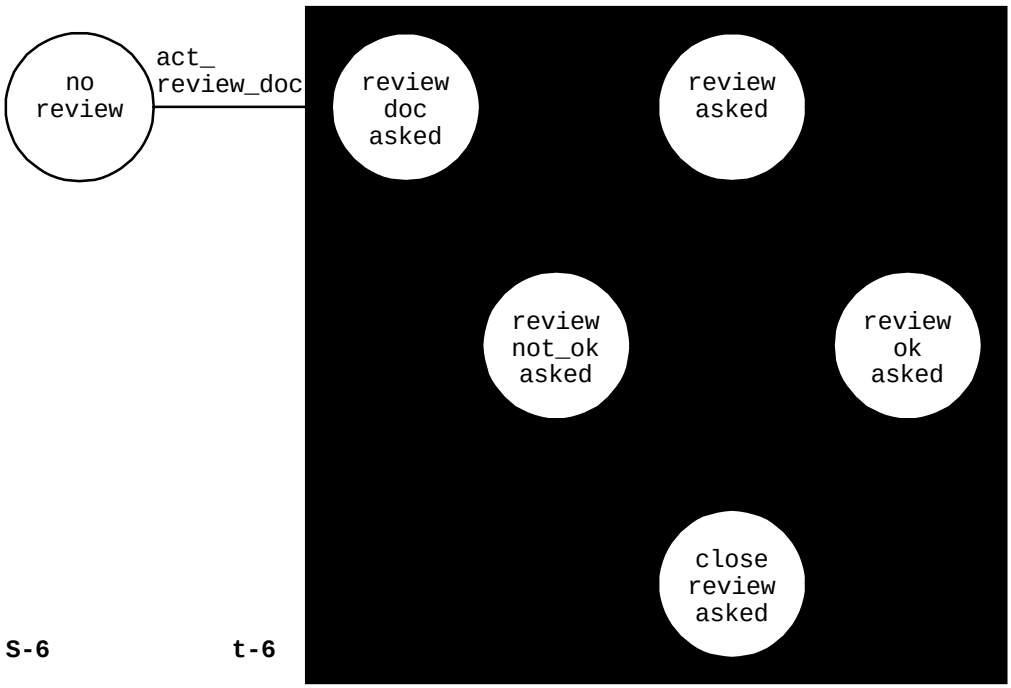

Figure 27 int_review_doc's subprocesses and traps w.r.t. reviewer

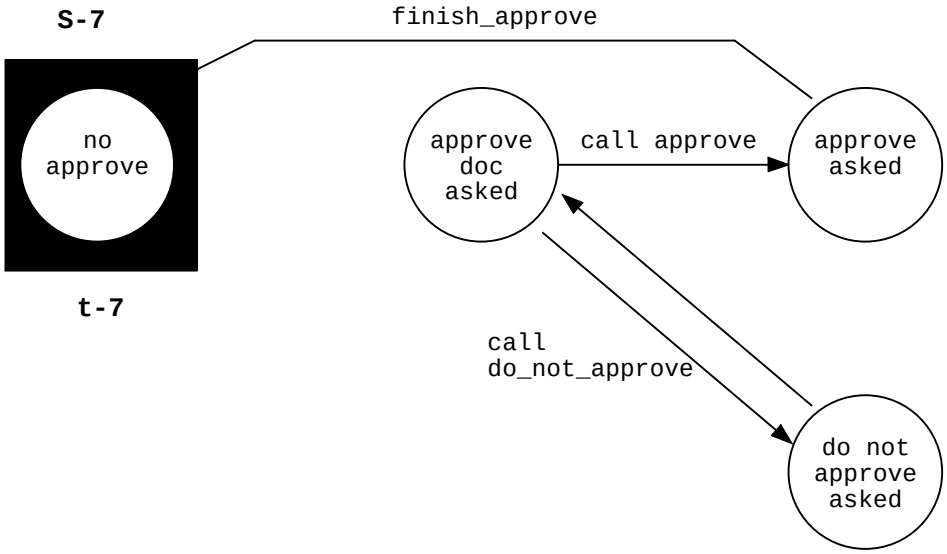

Figure 28 int_approve_doc's subprocesses and traps w.r.t. reviewer

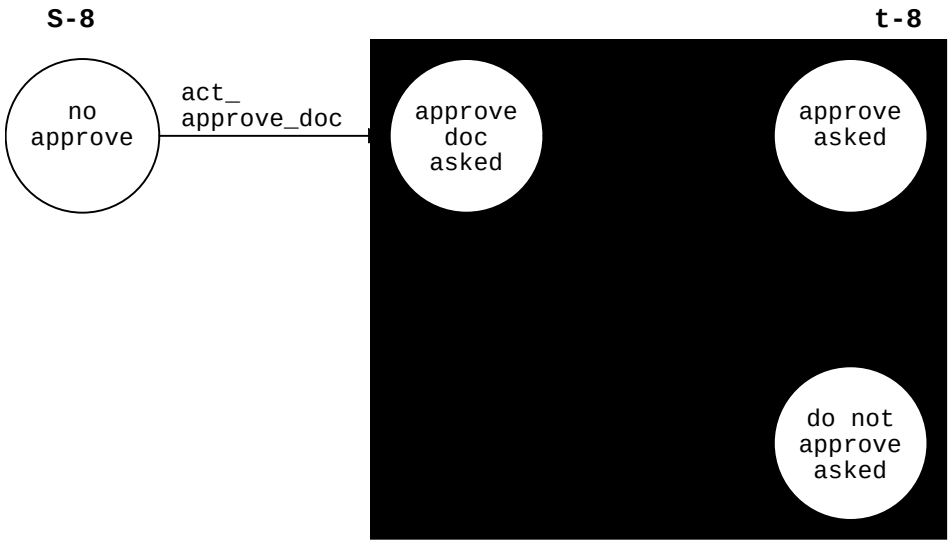

Figure 29 int_approve_doc's subprocesses and traps w.r.t. reviewer

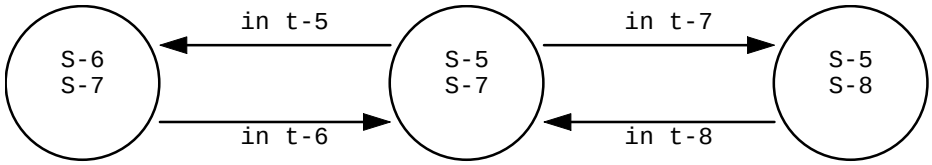

Figure 30 Reviewer; manager of int_review_doc and int_approve_doc

Reviewer is the manager process of the employee processes int_review_doc and int_approve_doc. Both employee processes are the internal behaviors of the same object the manager belongs to. The state {S-5, S-7} is the neutral state of reviewer and as soon as the employee processes enter the traps t-5 or t-7 a transition is possible to either state {S-6, S-7} or {S-5, S-8} (review_doc or approve_doc). As soon as the employee processes enter the traps t-6 or t-8 the manager returns to its neutral state.

Project_Docs

Project_Docs is the manager of the employee processes int_modify_doc and int_create_version. However only the employee process int_create_version is the internal behavior of the same object the manager belongs to. The employee process int_modify_doc is the internal behavior of another object calling create_version, and thus calling the export operation of this manager.

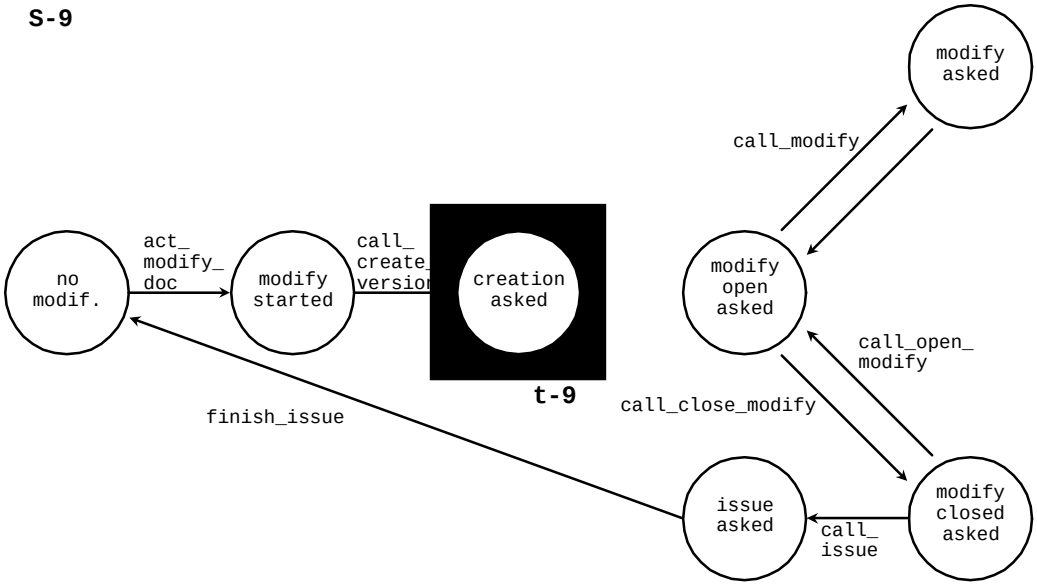

Figure 31 int_modify_doc's subprocesses and traps w.r.t. Project_Docs

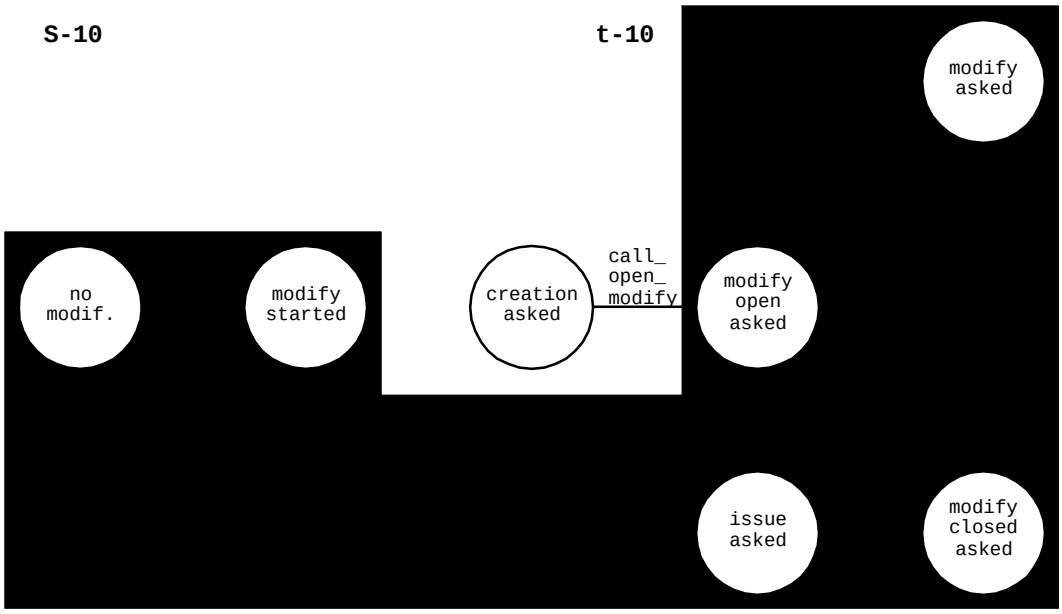

Figure 32 int_modify_doc's subprocesses and traps w.r.t. Project_Docs

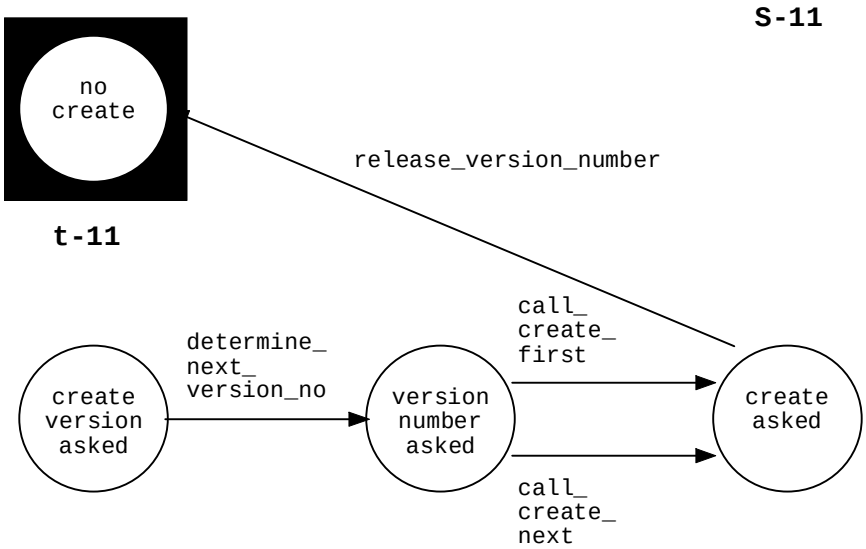

Figure 33 int_create_version's subprocesses and traps w.r.t. Project_Docs

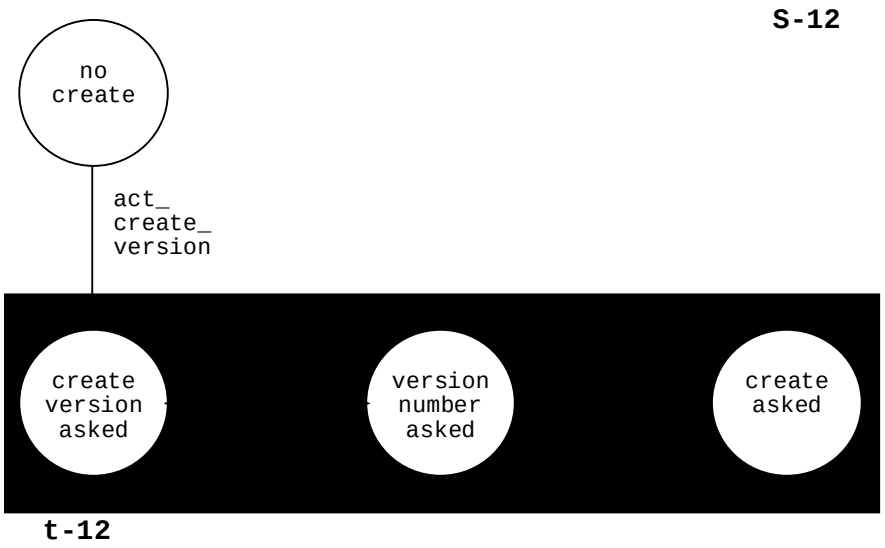

Figure 34 int_create_version's subprocesses and traps w.r.t. Project_Docs

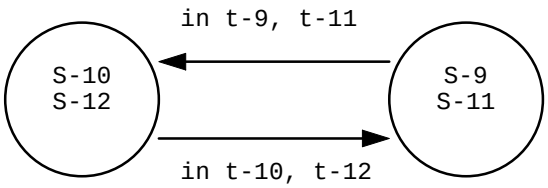

Figure 35 Project_Docs; manager of int_modify_doc and int_create_version

The state {S-9, S-11} represents the neutral state of Project_Docs and as soon as both employee processes have entered the traps t-9 and t-11 a transition is possible to the state {S-10, S-12} (create_version). As soon as both employee processes enter the traps t-10 and t-12 the manager returns to its neutral state. The transition from one state to another is now dependent on 2 traps. Note that the trap t-9 is a call from another object to create a version (create_version).

ISO_Document

ISO_Document is the manager process of the employee processes:

- int_create_version;
- int_modify_doc;
- int_review_doc;
- int_approve_doc;
- int_... (w.r.t. call read, which is explained later);
- int_withdraw_doc.

All employees are the internal behaviors of other objects calling the various export operations of this manager.

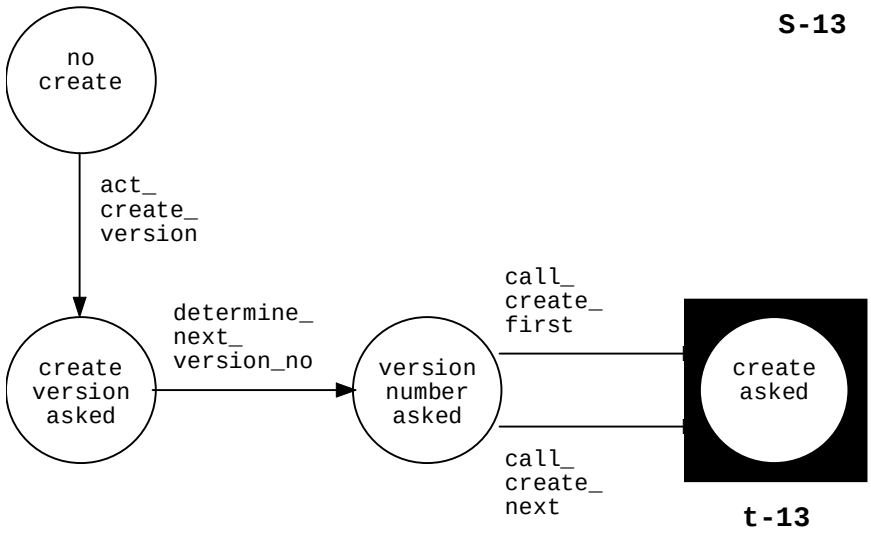

Figure 36 int_create_version's subprocesses and traps w.r.t. ISO_Document

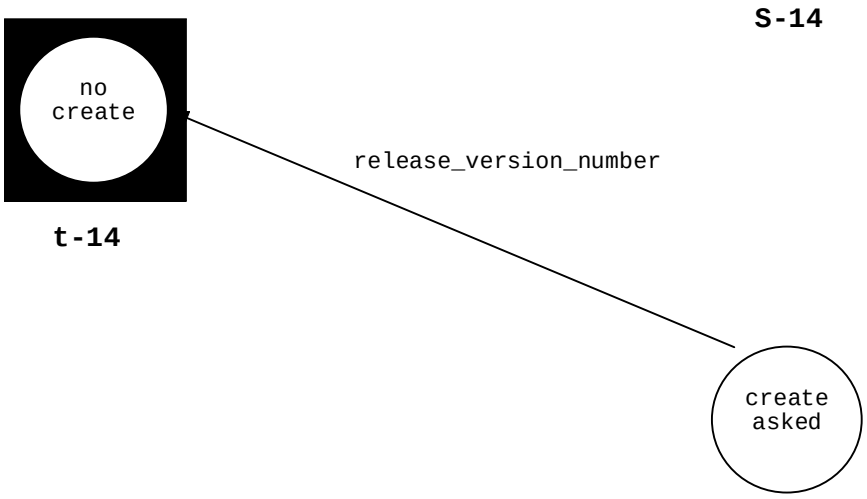

Figure 37 int_create_version's subprocesses and traps w.r.t. ISO_Document

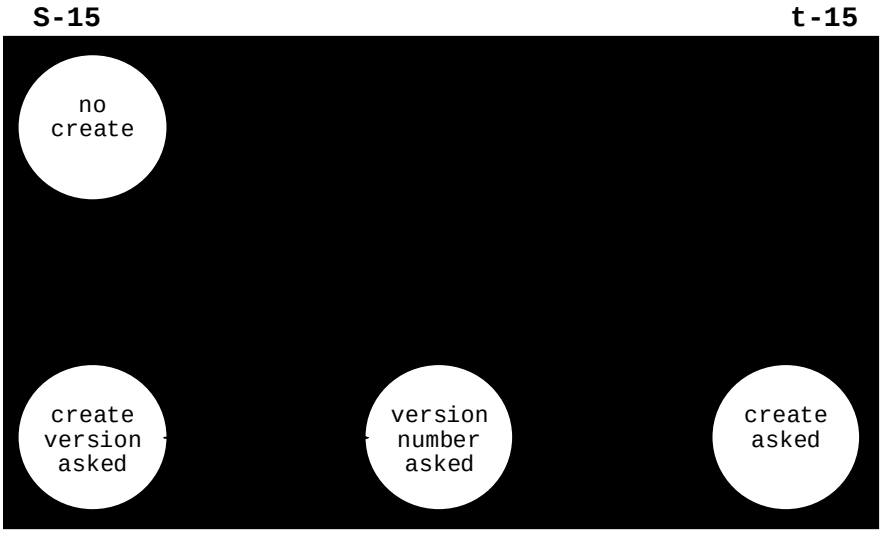

Figure 38 int_create_version's subprocesses and traps w.r.t. ISO_Document

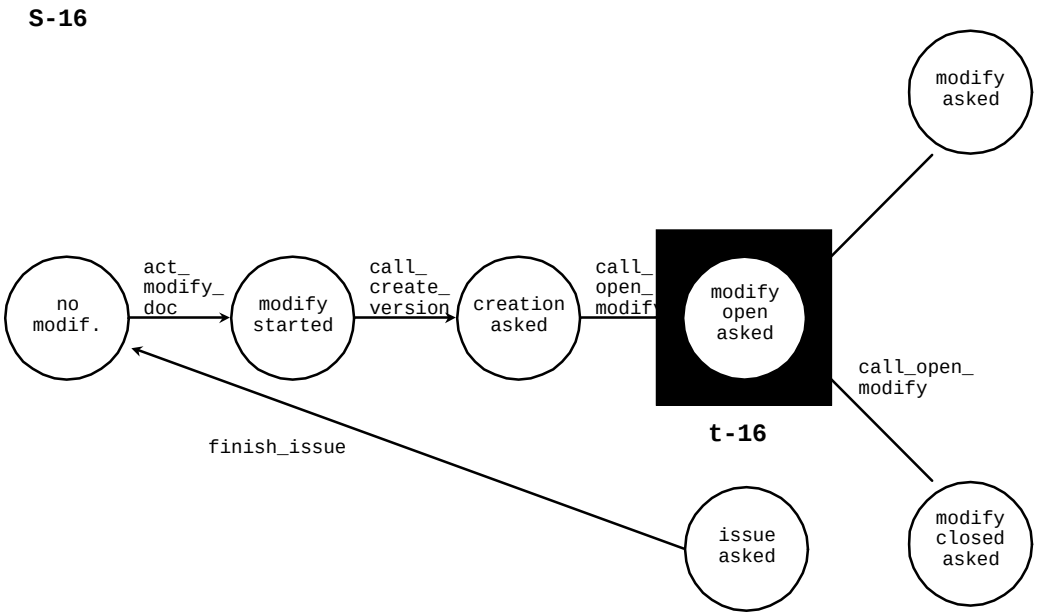

Figure 39 int_modify_doc's subprocesses and traps w.r.t. ISO_Document

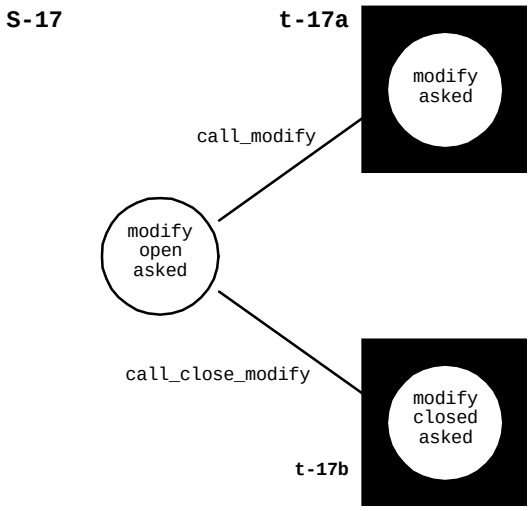

Figure 40 int_modify_doc's subprocesses and traps w.r.t. ISO_Document

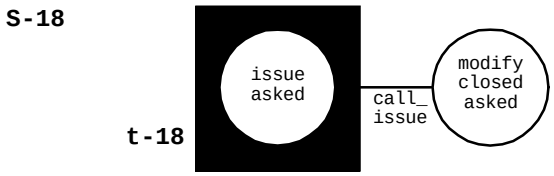

Figure 41 int_modify_doc's subprocesses and traps w.r.t. ISO_Document

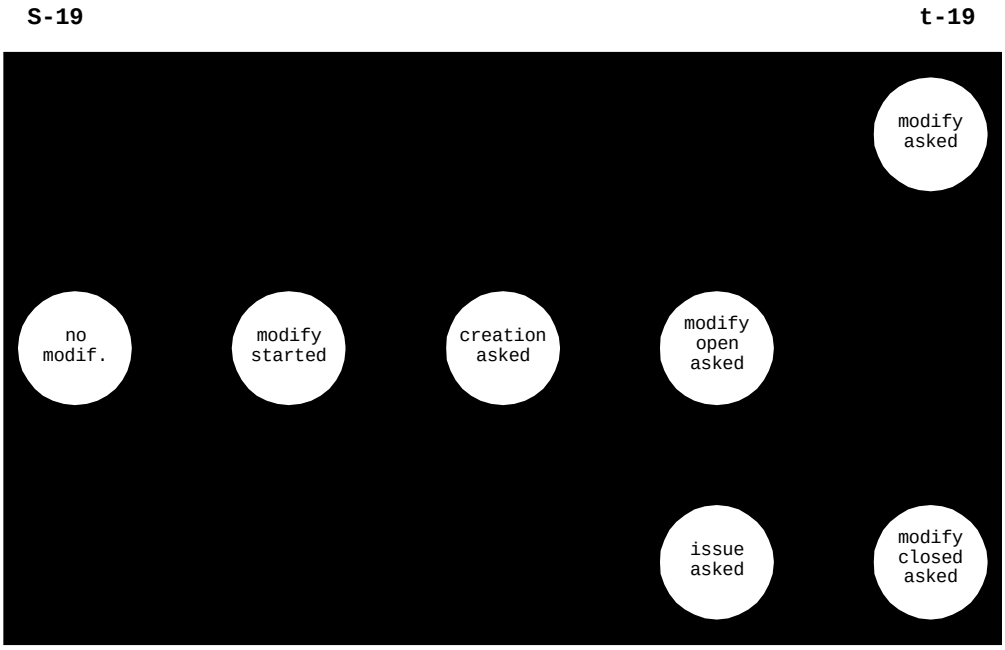

Figure 42 int_modify_doc's subprocesses and traps w.r.t. ISO_Document

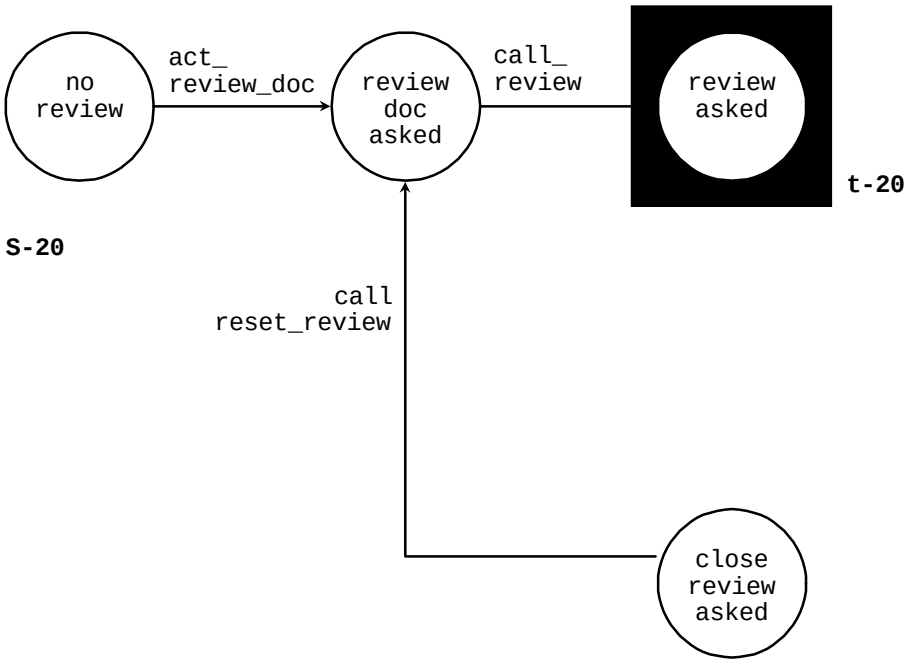

Figure 43 int_review_doc's subprocesses and traps w.r.t. ISO_Document

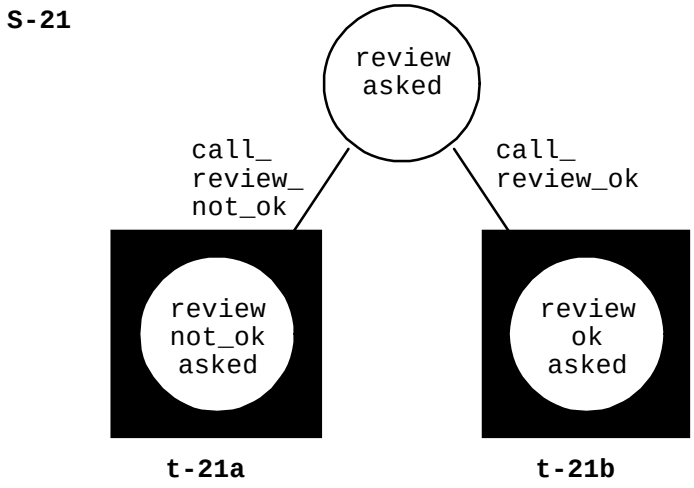

Figure 44 int_review_doc's subprocesses and traps w.r.t. ISO_Document

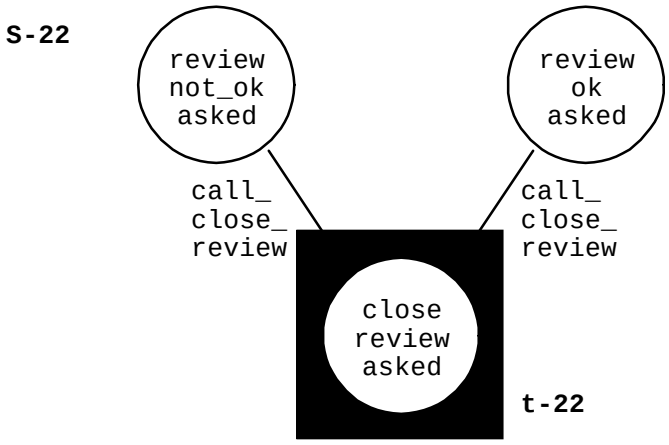

Figure 45 int_review_doc's subprocesses and traps w.r.t. ISO_Document

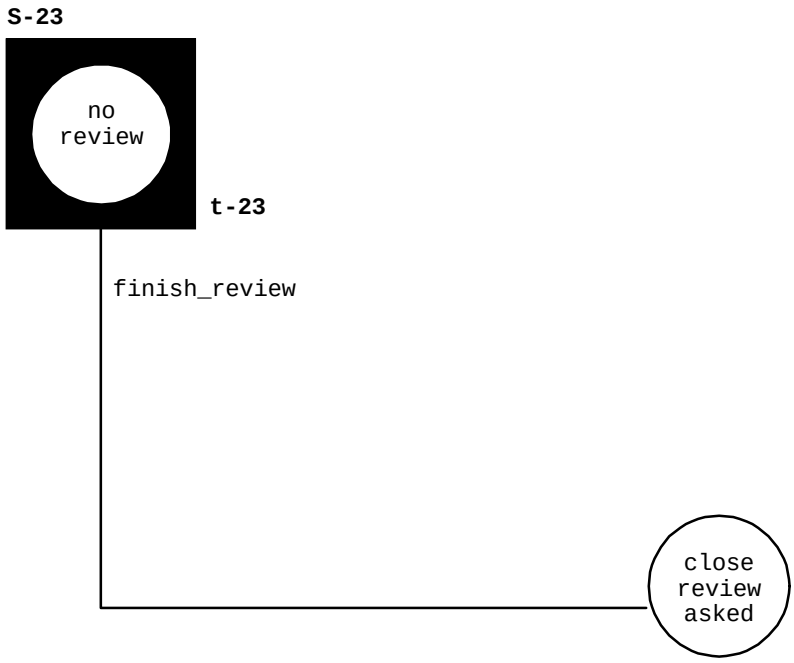

Figure 46 int_review_doc's subprocesses and traps w.r.t. ISO_Document

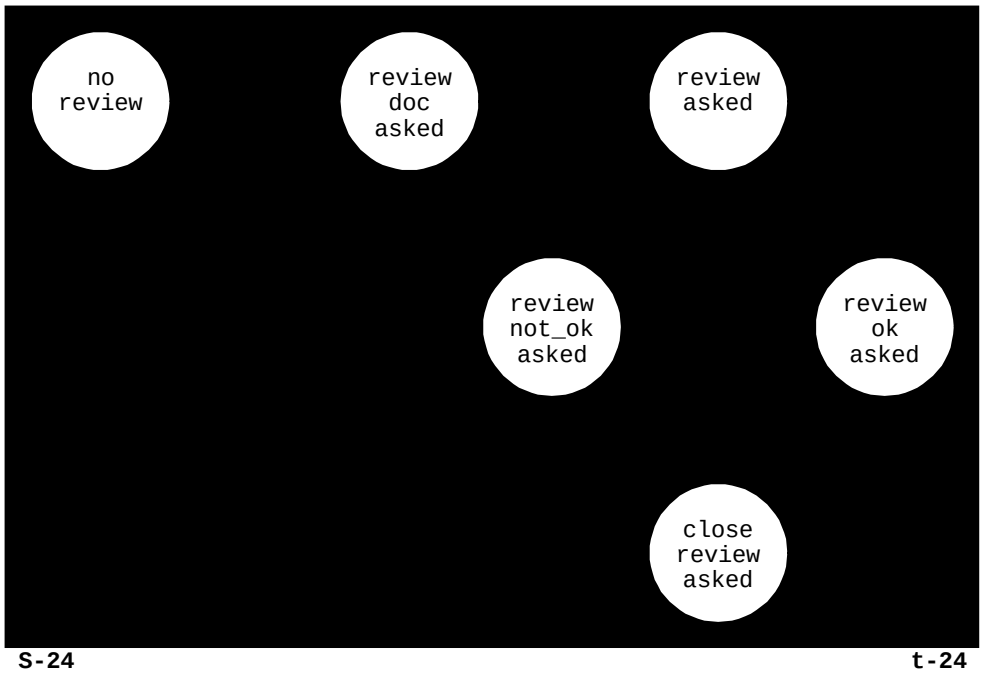

Figure 47 int_review_doc's subprocesses and traps w.r.t. ISO_Document

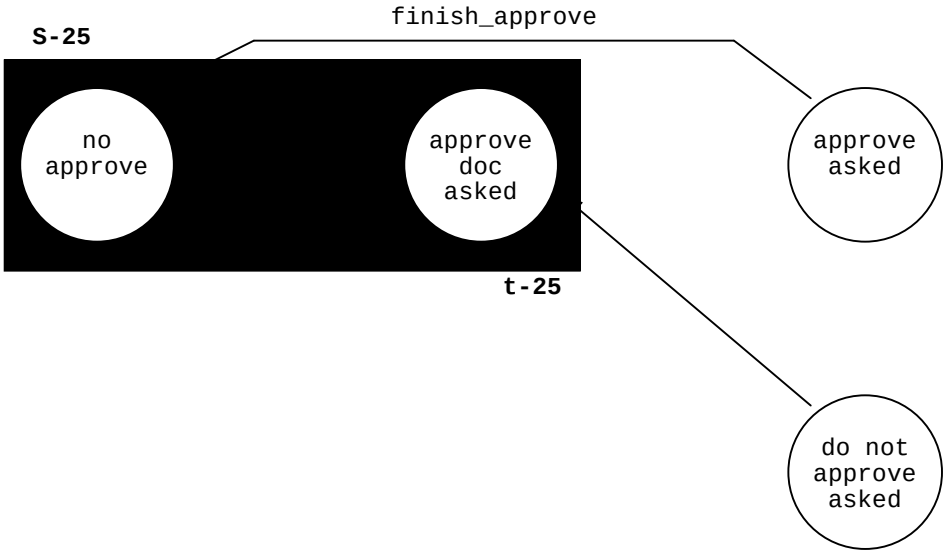

Figure 48 int_approve_doc's subprocesses and traps w.r.t. ISO_Document

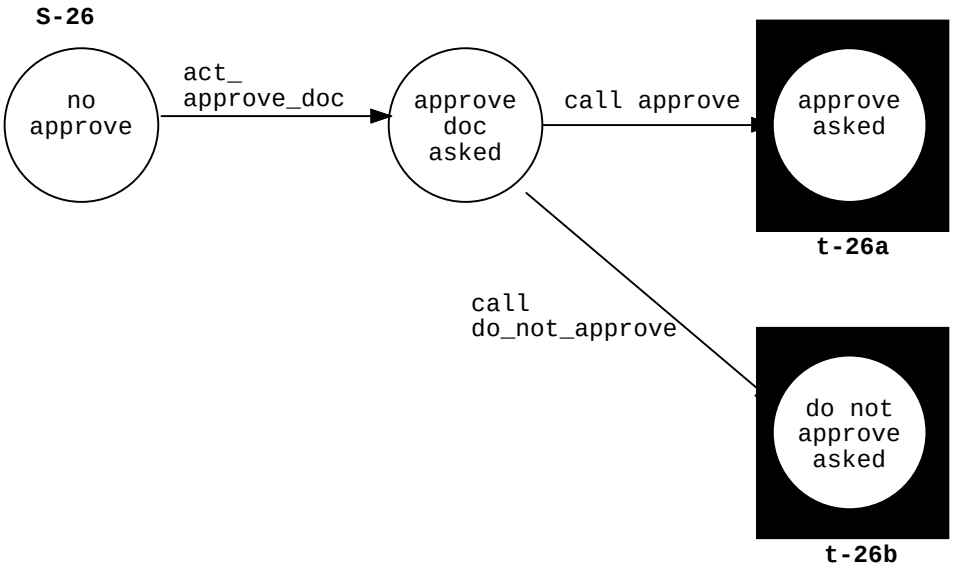

Figure 49 int_approve_doc's subprocesses and traps w.r.t. ISO_Document

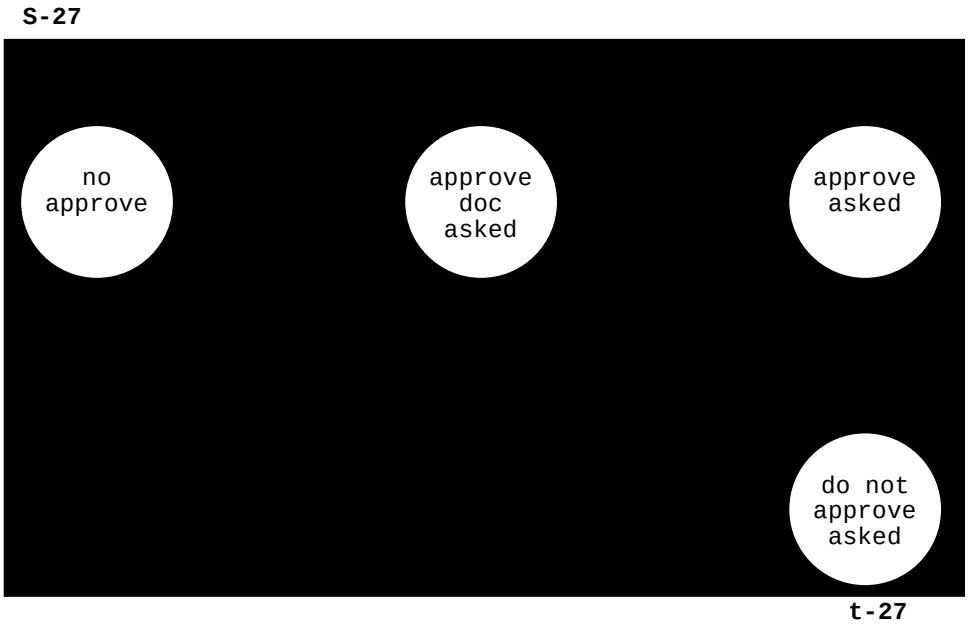

Figure 50 int_approve_doc's subprocesses and traps w.r.t. ISO_Document

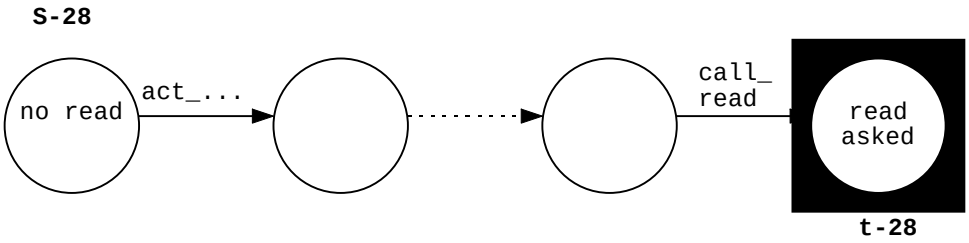

Figure 51 int_. . . subprocesses and traps w.r.t. ISO_Document

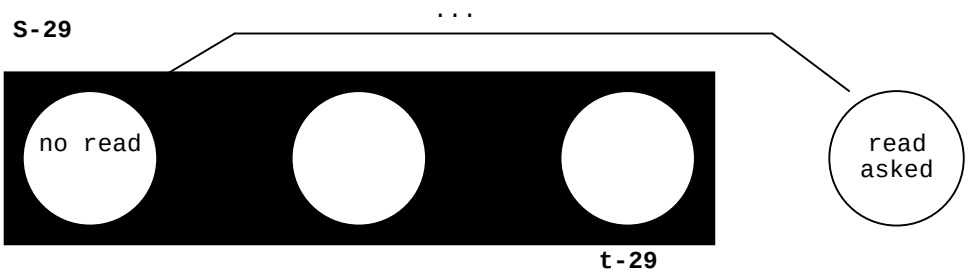

Figure 52 int_. . . subprocesses and traps w.r.t. ISO_Document

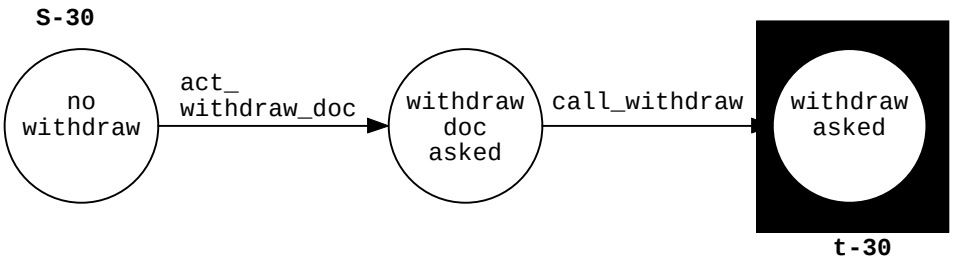

Figure 53 int_withdraw_doc's subprocesses and traps w.r.t. ISO_Document

Figure 54 int_withdraw_doc's subprocesses and traps w.r.t. ISO_Document

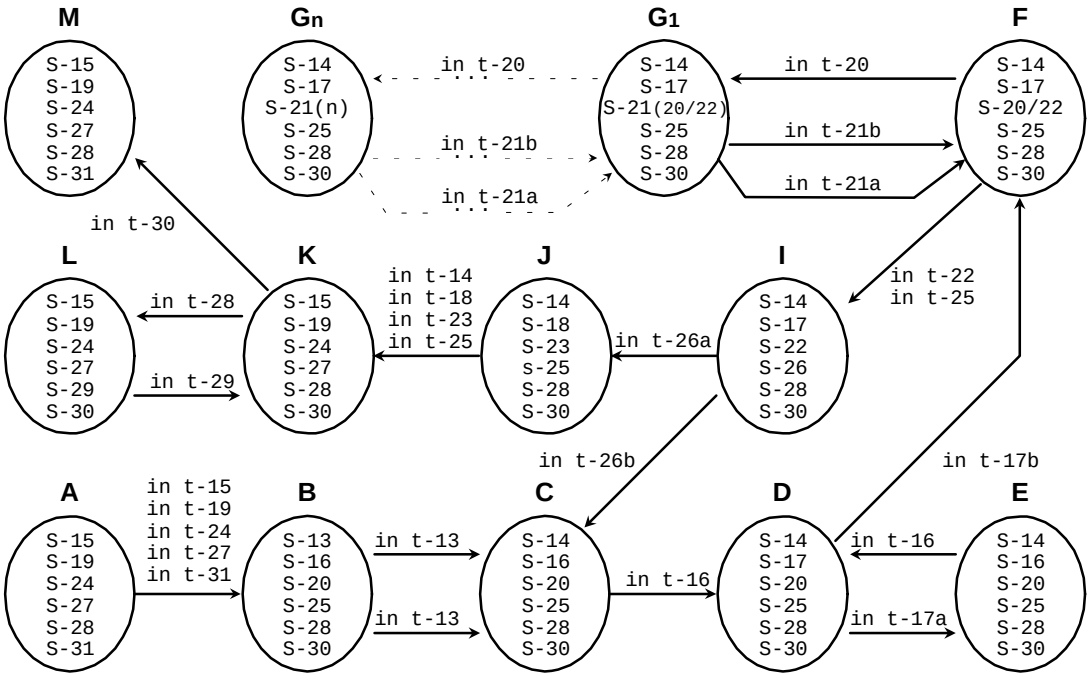

Figure 55 ISO_Document; manager of int_create_version, int_modify_doc, int_review_doc, int_approve_doc, int_. . . and int_withdraw_doc

The manager process ISO_Document is much more complicated compared to the previous discussed manager processes. For reference purposes the states of the manager are labeled with the letters A through M.

State A of the manager process represents the so-called ‘all subprocesses’ of its employee processes. The transition from A to B is a kind of initialization process of ISO_Document and confines its employee processes to the subprocesses prescribed in B. All ISO_Document employee processes are discussed in more detail below.

In B of ISO_Document the employee process int_create_version is in subprocess S-13. When the employee process enters its trap t-13 a transition is made from subprocess S-13 to S-14 (create_first or create_next). Note that during this transition the attributes fDocumentID, fVersionNumber and fStatus of ISO_Document are updated (see next chapter OFD). From C through J of ISO_Document the employee process int_create_version remains in subprocess S-14. In J of

ISO_Document and when int_create_version has entered its trap t-14 a transition is possible from subprocess S-14 to S-15 (provided the other employee processes have entered their respective traps at J). From K onwards the employee process int_create_version remains in its 'all subprocess' S-15 (that is after the document is issued).

In B and C of ISO_Document the employee process int_modify_doc is in subprocess S-16. In C and when int_modify_doc has entered its trap t-16 a transition is made from subprocess S-16 to S-17 (open_modify). In D the employee process may enter either trap t-17a or t-17b. In trap t-17a a transition is made from subprocess S-17 to S-16 (modify) and in trap t-16 (E) back to subprocess S-17. In D and when the employee process has entered trap t-17b (close_modify) it remains in this trap for some time. However it allows the manager to handle the subprocesses of other employees (F through I). By the condition of some other employee process the manager may make a transition from I to either C or J. Since int_modify_doc was locked in trap t-17b it can change its subprocess to either S-16 or S-18. In J and when int_modify_doc enters trap t-18 (issue) in subprocess S-18, it makes a transition to its 'all subprocess' S-19 and remains in that subprocess (that is after the document is issued).

The employee processes int_create_version and int_modify_doc are straight forward. The employee process int_review_doc is more complicated due to the ISO requirement that a document must be reviewed by authorized personnel. Moreover there may be a single reviewer or a review team, and whereby a review team may review the document either in sequence, in parallel or in a combination of both (depending on the quality procedures of the organization and document type). This results in that there may be $n \geq 1$ $\{n \in \mathbb{N}\}$ instances of the employee process int_review_doc in contrast with all the other employee processes, which have one instance only with respect to the instance of ISO_Document under consideration.

From B through F (F entered from D) all employee processes int_review_doc are in subprocess S-20. Now let $i, j \in \{1, ..., n\}$ with $i \neq j$. In F and when the i-th employee process int_review_doc_i enters trap t-20_i it makes a transition from subprocess S-20_i to S-21_i (review). In G_i either of following situations may happen:

a) When int_review_doc$_i$ is in trap t-21a$_i$ (review_not_ok) or t-21b$_i$ (review_ok) it makes a transition from subprocess S-21$_i$ to S-22$_i$. The manager process transits from G_1 to F.
b) The j-th employee process int_review_doc$_j$ enters its trap t-20$_j$ and makes a transition from subprocess S-20$_j$ to S-21$_j$ (review). The manager process transits from G_1 towards G_2.

In other words and seen from the manager process ISO_Document perspective the following situations may occur:

a) The manager cycles between the states F -> G_1 -> F (n times). This means that the document is reviewed in sequence by a review team (the previous rewiewer finishes its review (review_not_ok or review_ok) before the next reviewer starts reviewing the document).
b) The manager cycles (once) between the states F -> G_1 -> ... -> G_n -> ... -> G_1 -> F. This means that the document is reviewed in parallel by a review team (in G_n all reviewers are in subprocess S-21, which means that all reviewers are reviewing the document).
c) A combination of a) and b) is possible. This means that some reviewers may have reviewed the document (subprocess S-22), some reviewers are reviewing the document (subprocess S-21) and some reviewers must start to review the document (subprocess S-20). The manager cycles at least once through the states F -> G_1 -> G_m -> G_1 -> F, with $m < n$.

The manager transits from F to I when all employee processes of int_review_doc have entered their trap t-22 in subprocess S-22. However in I the employee processes remain in trap t-22. Depending on whether the employee process int_approve_doc calls approve or do_not_approve will int_review_doc make a transition to either subprocess S-23 or S-20. In J and in trap t-23 int_review_doc makes a transition from subprocess S-23 to its ‘all subprocess’ S-24 and it remains in that subprocess from K onwards.

From B through F (including the review cycle F -> G_1 -> G_m -> G_1 -> F, where in this case $m \leq n$) the employee process int_approve_doc is in subprocess S-25. In F and when int_approve_doc is in trap t-25 it makes a transition from subprocess S-25 to S-26 (all employee processes int_review_doc’s must be in trap t-22). In subprocess S-26 a call is made to either approve or do_not_approve in trap t-26a or t-26b. In both cases int_approve_doc makes a transition from subprocess S-26 to S-25 and ISO_Document

transits from I to either J or C. In J and in trap t-25 a transition from subprocess S-25 to the ‘all subprocess’ S-27 is possible.

ISO-9001 states that pertinent issues of documents are available at all locations which need these documents and that obsolete documents are promptly removed from all points of issue. This is modeled with the employee processes int_... and int_withdraw_doc. However the reason why a document is used is not modeled here, but solely that it is used by a read call.

A document of a specific version may only be read after it is issued and before it is withdrawn. Therefore from A through K the employee process int_... remains in subprocess S-28. In K and when int_... is in trap t-28 a transition occurs from subprocess S-28 to S-29 (read). As soon as int_... reaches trap t-29 in subprocess S-29 a transition occurs back to subprocess S-28 and the manager returns from L to K. Note that there may be many employee processes int_.... At present the only condition modeled is that the document may be released to a reader after it is issued and before it is withdrawn. Perhaps it is better if ISO_Document is modeled such that it keeps track of all the readers (by a borrow list that registers loans and returns of documents). A withdraw call can than be preceded by a notification to all borrowers stating that the document is no longer valid.

The employee process int_withdraw_doc remains in subprocess S-30 from B through K (including L). In K and when int_withdraw_doc enters trap t-30 a transition occurs from subprocess S-30 to S-31.

3.2.6 OFD

Finally the OFD's are presented concerning document control. Since at present no formalism exists within SOCCA for the OFD's, the process perspective (transitions and the effect on the data) is shown in the table below.

(call_modify_doc) - external modify_doc act_modify_doc	Inserts an instance of modifies relationship; modifies.status := passive
(call_withdraw_doc) - external withdraw_doc act_withdraw_doc	Inserts an instance of withdraws relationship
(call_review_doc) - external review_doc act_review_doc	Inserts an instance of reviews relationship; reviews.NoOfReviewers := n reviews.ReviewOk := 0 reviews.ReviewNotOk := 0
(call_approve_doc) - external approve_doc act_approve_doc	Inserts an instance of approves relationship
call_create_version create_version act_create_version	Nothing: effect is postponed until after either call_create_first or call_create_next
call_open_modify open_modify act_open_modify	Updates modifies.status := active
call_modify modify act_modify	Updates Document.fContent
call_close_modify modify act_modify	Updates modifies.status := passive

call_issue issue act_issue	Updates: Document.fDocumentTitle Document.fAuthor Document.fIssueDate ISO_Document.fStatus := readable
call_withdraw withdraw act_withdraw	Updates Document.fStatus := obsolete Deletes instance of withdraws
call_review review act_review	Nothing: effect is postponed until after call_review_ok or call_review_not_ok
call_review_ok review_ok act_review_ok	Updates: INC(reviews.ReviewOk) {= i} ISO_Document.fReviewedBy[i] ISO_Document.fReviewDate[i] ISO_Document.fRevComment[i]
call_review_not_ok review_not_ok act_review_not_ok	Updates: INC(reviews.ReviewNotOk) {= i} ISO_Document.fReviewedBy[i] ISO_Document.fReviewDate[i] ISO_Document.fRevComment[i]
call_approve approve act_approve	Updates: ISO_Document.fApprovedBy ISO_Document.fApprovalDate Deletes instance of approves
call_do_not_approve do_not_approve act_do_not_approve	Nothing
call_read read act_read	Nothing
call_create_first create_first act_create_first	Creates instance of ISO_Document with: Document.fDocumentID ISO_Document.fVersionNumber := 1 ISO_Document.fStatus := not_available

call_create_next create_next act_create_next	Creates new instance of ISO_Document with: Document.fDocumentID := (same as previous id number) INC(ISO_Document.fVersionNumber) ISO_Document.fStatus := not_available
call_close_review close_review act_close_review	Nothing: effect is postponed until after call_approve or call_reset_review
call_reset_review reset_review act_reset_review	Resets: ReviewOk := 0 ReviewNotOk := 0

Figure 56 OFD; document control

Note: Document and ISO_Document is used to distinguish both classes. However they refer to the same instance. The instance of ISO_Document inherits the properties of its superclass Document.

3.3 Quality Plan

3.3.1 Introduction

The ISO-9001 article relating to quality planning is quite short. It mainly states that the supplier needs to establish and maintain a documented quality system as a means for ensuring that the product conforms to the specified requirements. Establishing a documented quality system involves the following activities:

a) The preparation of documented quality system procedures and instructions in accordance with the requirements of ISO-9001.
b) The effective implementation of the documented quality system procedures and instructions.

The ISO-9000-3 guidelines state the same but is more specific concerning the development, supply and maintenance of software. It states:

a) The supplier should prepare and document a quality plan to implement quality activities for each software development on the basis of a quality system, and make it understood and observed by the organization concerned.
b) The quality plan should be updated along with the progress of the development and items concerned with each phase should be completely defined when starting the phase.

Besides the above, ISO-9000-3 gives some guidelines to which items should be covered by the quality plan. It ends its article with the following important statement:

> The document that describes the quality plan may be an independent document (entitled Quality Plan), or a part of another document, or composed of several documents including the development plan.

For the purpose of modeling quality planning, the ISO requirements can de divided into the following categories:

a) Quality plan: the requirements of the document;
b) Quality planning: the requirement that a quality plan must be made or updated at the start of each project phase. This category involves the use of the quality plan.

‘Document Control’ is described in an earlier chapter. The first category relates to a specialization of ‘Document Control’. The second category relates to activities which uses this specialization of ‘Document Control’. The objective

is, besides modeling quality planning, to investigate how well the object oriented approach of SOCCA can be used with specialization and by using this specialization with some other classes.

In this chapter the quality plan will be modeled (specialization), and in the next chapter quality planning will be modeled (using the quality plan).

Only the figures which differ from the figures presented in the chapter on ‘Document Control’ are included in this and the following subchapters, otherwise reference is made to the earlier figures.

3.3.2 Class diagrams

Before modeling a quality plan, some assumptions will be made which differentiates a quality plan from a general ISO-document. These assumptions are:

a) A company may have a document describing the quality system in general. However for a project the quality plan is specific for the project (for instance it describes the quality objectives of the project). During the execution of the project the quality plan may need to be updated. This implies that an earlier version of the project's quality plan must be withdrawn (if it exists) when the updated quality plan is issued. In the document control model, as explained in the previous chapter, an earlier document version may coexist besides a newer version until it is explicitly withdrawn.

b) A quality plan may contain zero or more other documents. For instance a test plan may be a separate document, but it forms part of a quality plan. Because these sub-documents are part of the quality plan, they belong to the same class (class Quality_Plan).

The class diagram in figure 57 now contains the additional

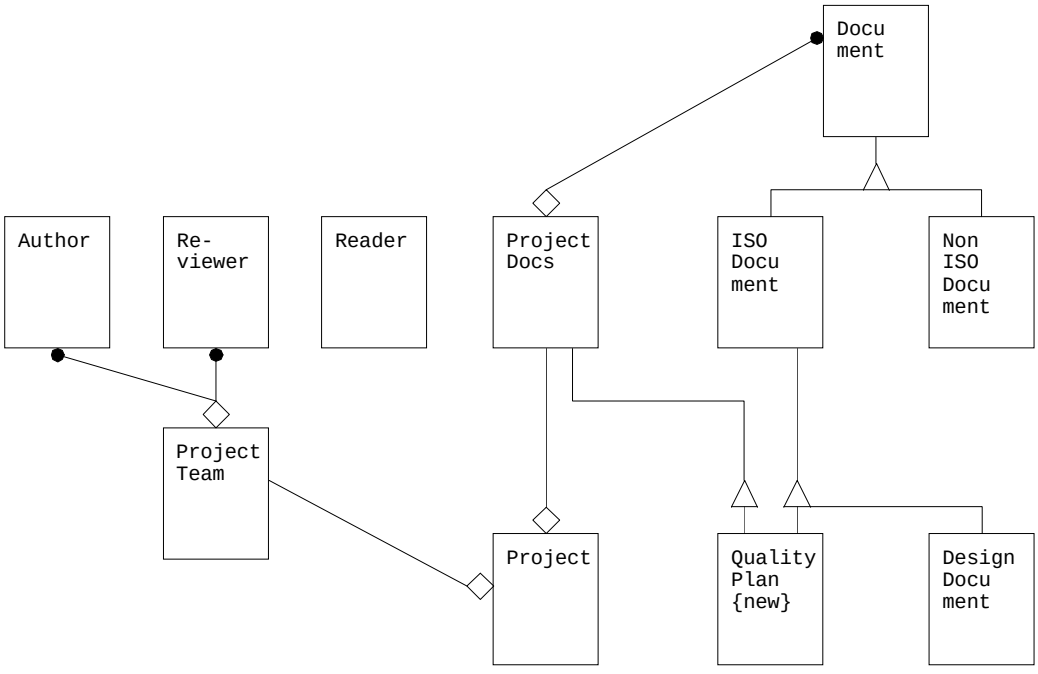

Figure 57 Class diagram: document classes and is-a and part-of relationships

class Quality_plan. (In this and other figures the word ‘{new}’ indicates what has been added, compared to the figures in the previous chapter). By multiple inheritance it inherits the attributes and operations from the classes ISO_Document and Project_Docs. By inheritance from Project_Docs the class Quality_plan can now refer to other documents that belong to the quality plan (for instance a test plan document), but have been written by other authors.

Figure 58 contains the document attributes and operations. All attributes and operations of the earlier discussed classes remain unchanged. In case an existing operation from an inherited class needs to be modified, than it will be modified in the class Quality_plan. An exception is the operation modify_subdoc, which is added to the class Author. Although this operation is added to the existing class Author, it does not change its behavior with respect to ISO_Documents. Otherwise it would have been better to add this operation in a new subclass of Author. The purpose of this operation is explained later when the STD’s of the external and internal behaviors are discussed.

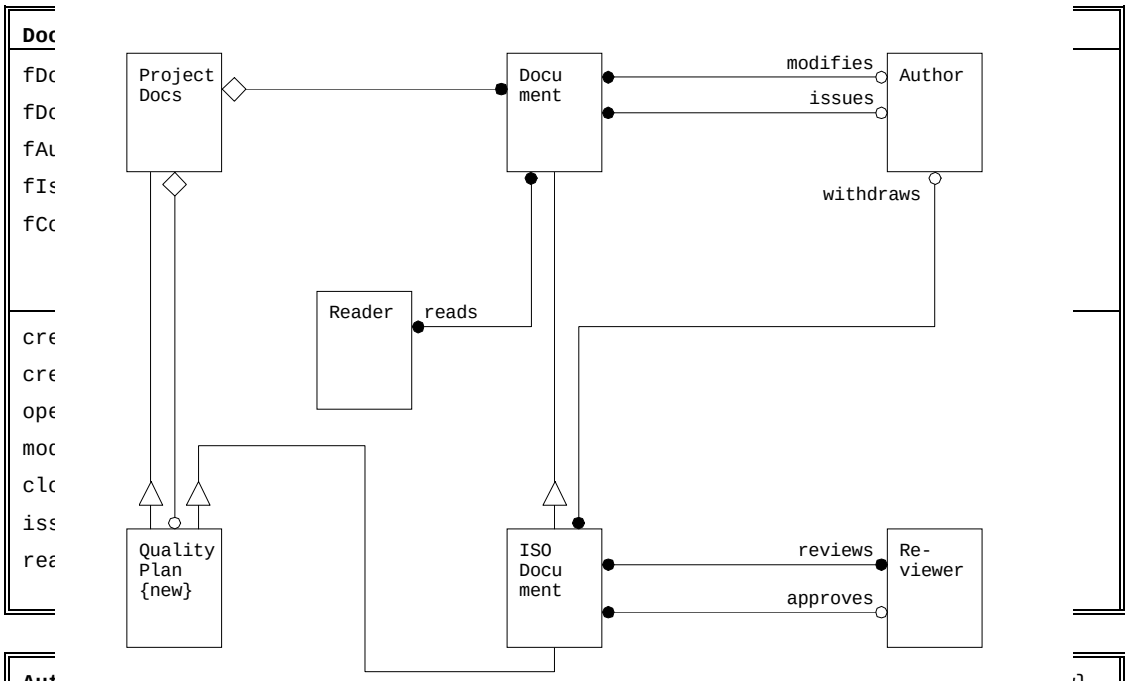

Figure 59 Class diagram: classes and general relationships

Figure 58 Class diagram: document attributes and operations

Figure 59 shows the classes and general relationships. The class Quality_plan is added and which has the ‘is-a’ relationship with the classes Project_Docs and ISO_Document. The class Project_Docs remains a container class of many Documents and its subclasses, but is at the same time a container class of only one Quality_Plan. Using multiple inheritance, sub-documents of a quality plan can be created and modified that are of the same class as Quality_Plan. The master quality plan document acts as a container class for these sub-documents.

Figure 60 needs further explanation. First the figure is viewed excluding the Project_Docs class, and with the focus on the Quality_Plan class. For reference purposes the same numbers are used with the uses relations as in the previous chapter. Due to multiple inheritance the class Quality_Plan inherits its export operations from ISO_Document and Project_Docs and thus it also inherits their uses relations 3, 4, 5, 6, 7 and 8. This enables the creation of sub-

documents for the quality plan. These sub-documents are of the same class as the master quality plan document and which also serves as a container class for its sub-documents. Note that because the sub-documents are of the same class as their container class that these sub-documents may in turn be a container class for their sub-documents.

The figure is now viewed including the class Project_Docs. This class is used as a container class to create all project documents, that is all project's Non_ISO_Documents and ISO_Documents as explained in the previous chapter, and the Quality_Plan (but only one quality plan document as shown in figure 59). Thus with this class it is possible to create the master quality plan document.

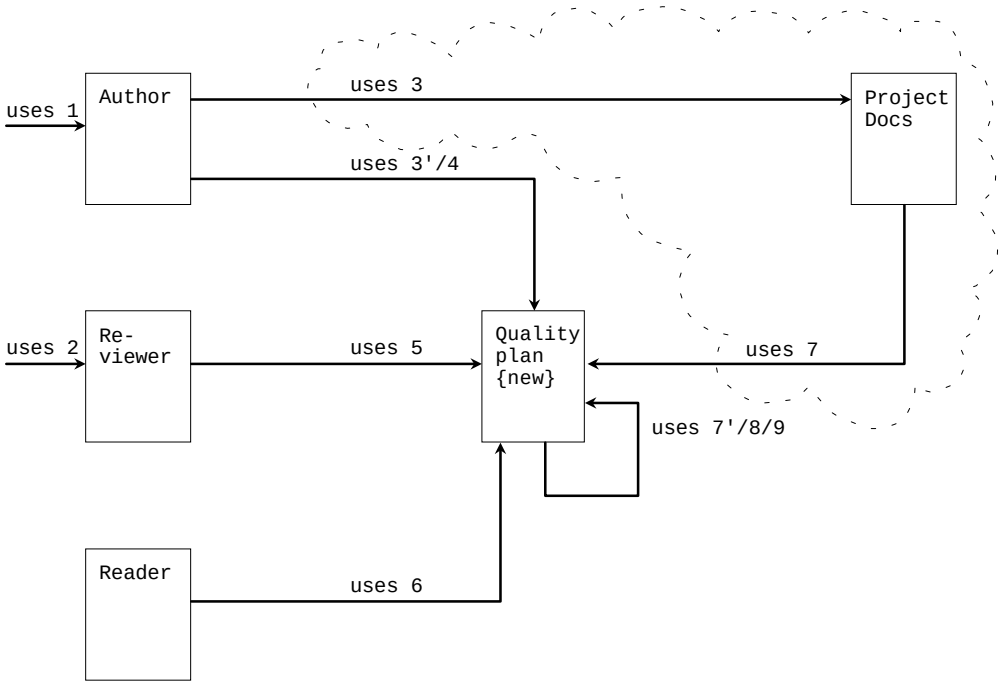

Figure 60 Import/export diagram

```
uses 1
  modify_doc
  withdraw_doc
  modify_subdoc {new}
uses 2
  review_doc
  approve_doc
uses 3/3´{new}
  create_version
uses 4
  open_modify
  modify
  close_modify
  issue 1)
  withdraw
uses 5
  review
  review_ok
  review_not_ok
  approve
  do_not_approve
uses 6
  read
uses 7/7´{new}
  create_first
  create_next 1)
uses 8
  close_review
  reset_review
uses 9{new}
  make_copy
  delete_copy

1) modified operation
```

Figure 61 Import list

Finally the condition must be met that only one version of a quality plan may exist. This is achieved with the additional

export operations make_copy and delete_copy. When the quality plan is modified it makes a copy of itself (with the exception of the first version). Thus until the issue of the updated version a copy of itself exists that can be read by others. When the quality plan is issued the copy is deleted and the new version becomes available.

To avoid name conflicts in the export operations (for instance between the export operations of Project_Docs and Quality_Plan with the uses 3 relations), these inherited, but conflicting uses relations of the class Quality_Plan are indicated with a ‘´’. The uses 3 relation is used to create a version of a document, and in case of the quality plan to create a version of the master quality plan document. The uses 3´ relation is used to create a version of a quality plan’s sub-document. The class Author needs to know from which class it should call the operation create_version. This is achieved with the additional operation modify_subdoc in the class Author, which calls the create_version operation from the Quality_Plan class. This is further explained in the section that discusses the STD’s of the external and internal behaviors.

3.3.3 STD of the external behavior

The STD's of the external behaviors are:

- Author;
- Reviewer;
- Project_Docs;
- Quality_Plan and Quality_Plan as Project_Docs.

The STD's of the external behaviors are shown in the subsequent figures. The STD of the external behavior of Quality_Plan is different compared to the STD of the external behavior of ISO_Document as discussed in the previous chapter. The STD of Author contains the additional operation modify_subdoc.

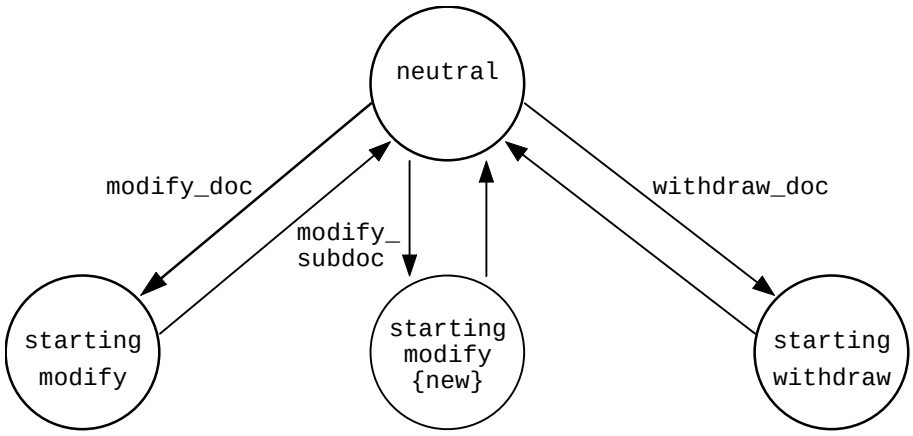

Figure 62 Author; STD of the external behavior

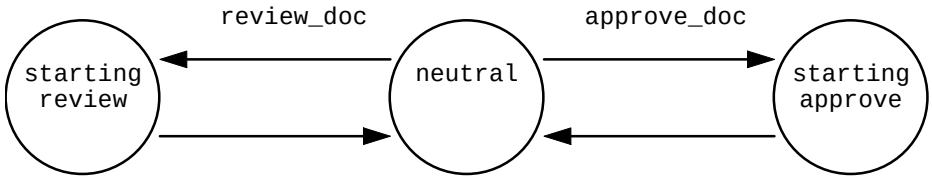

Figure 63 Reviewer; STD of the external behavior

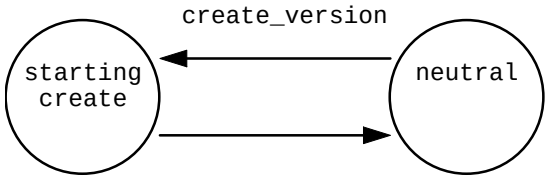

Figure 64 Project_Docs; STD of the external behavior

Since the above figures are almost the same as discussed in the previous chapter, they do not need any further explanation. Note that concerning quality plans the class Project_Docs is only used to create a version of the master quality plan document. All quality plan sub-documents (for instance a test plan) are created with a call create_version´ to Quality_Plan.

The STD of the external behavior of Quality_Plan is different compared to the STD of the external behavior of ISO_Document. Due to the inheritance of the class Quality_Plan from its superclasses ISO_Document and Project_Docs the STD's of its external behavior can be divided into the following categories:

- the STD of its external behavior regarding document modification;
- the STD of its external behavior as inherited from Project_Docs and regarding its own sub-documents.

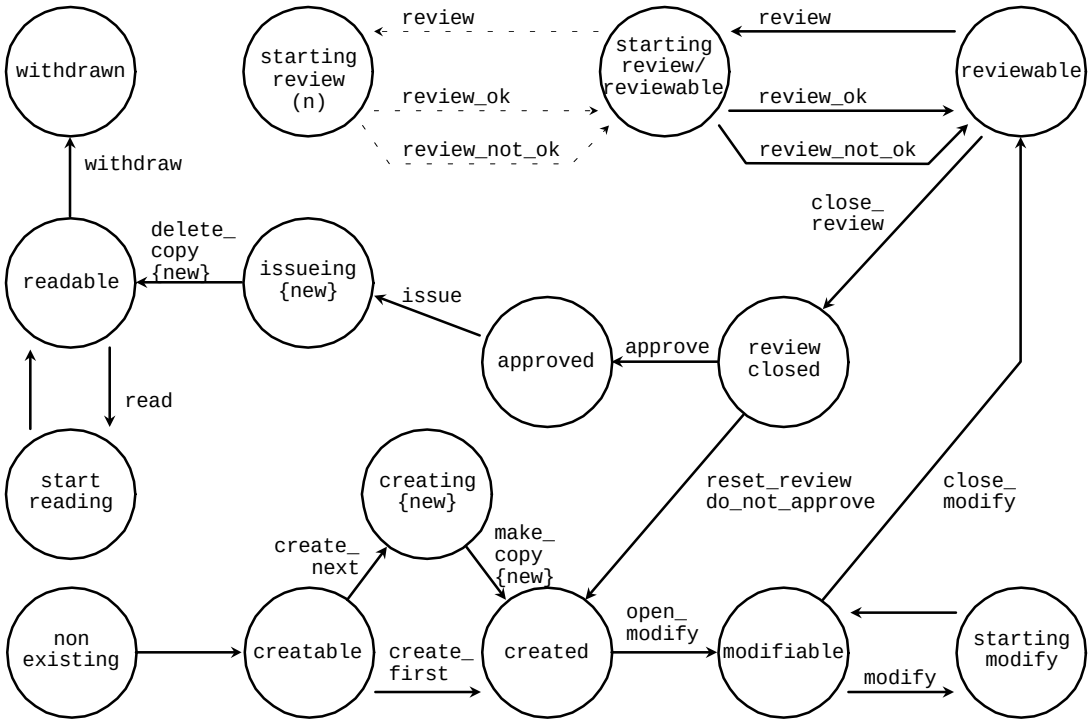

Figure 65 Quality_Plan; STD of the external behavior (w.r.t. document modification)

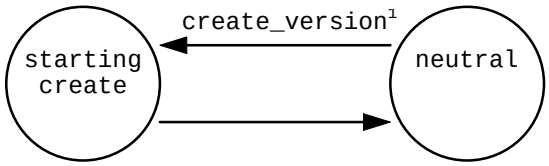

Figure 66 Quality_Plan as Project_Docs; STD of the external behavior (w.r.t. sub-documents) {new}

Both categories are drawn separately for pragmatic reasons, since otherwise the STD of the external behavior would become too clustered. The STD of the external behavior of Project_Docs as Quality_Plan should have been drawn on top of the STD of the external behavior of Quality_Plan. The state neutral of Project_Docs as Quality_Plan coincides with all states of Quality_Plan with the exception of the states non-existing and withdrawn. The master quality plan (instance of the class Quality_Plan) thus also acts as a container class for all its sub-documents. This is possible provided this instance is not in the states non-existing or withdrawn.

In the STD of the external behavior w.r.t. document modification of Quality_Plan two other changes are

implemented compared to the STD of the external behavior of ISO_Document. These changes are:

- the operations create_next and make_copy;
- the operations issue and delete_copy.

With these operations the ISO-9001 requirement can be met that only one valid version of a quality plan exists. When a document needs to be modified (version number > 1), than the instance of the class Quality_Plan makes a copy of itself. During the modification process the original copy is still available to all readers. When the modified document is issued (thus becomes accessible for readers), it deletes the copy so it is no longer accessible. The inherited operations create_next and issue are slightly modified (polymorphism).

The operation create_next is actually no longer required and instead a direct call could have been made to make_copy. However since it is called from other classes this would impose modifying the other classes as well. Instead the operation create_next is modified so it redirects the call to make_copy. Other than this the operation create_next does not do anything.
Almost the same applies to the operation issue. It has the same function as in ISO_Document, but it ends its operation with a call to delete_copy.
Thus all changes have been localized in the class Quality_Plan.

3.3.4 STD of the internal behavior

The succeeding figures show the STD's of the following internal behaviors:

- modify_doc;
- modify_subdoc;
- review_doc;
- approve_doc;
- withdraw_doc;
- ... (w.r.t. call read, which is explained in the previous chapter);
- create_version;
- create_version´.

Only some of the STD's of the internal behaviors w.r.t. the quality plan documents are different compared to the STD's of the internal behaviors w.r.t. the ISO documents.

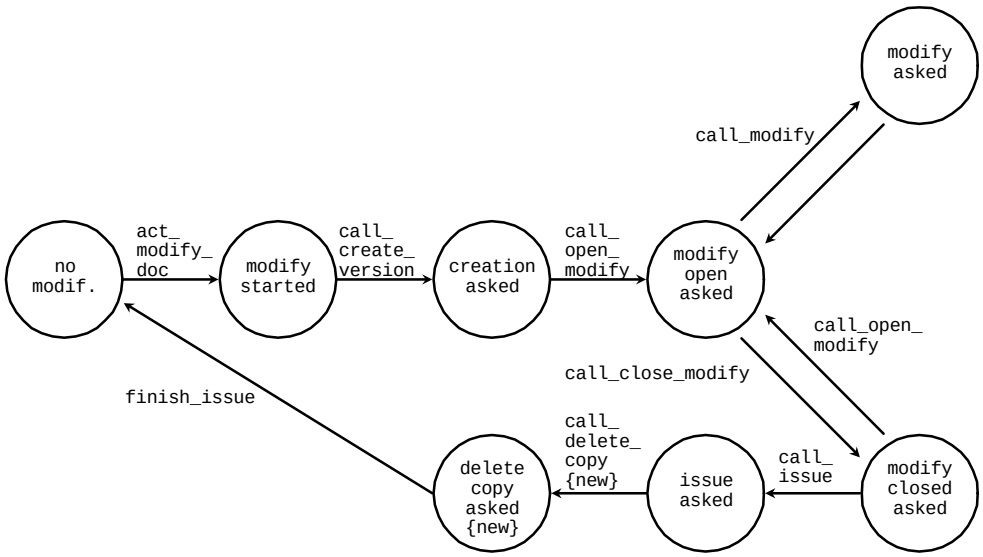

Figure 67 int_modify_doc; STD of the internal behavior

Compared to the previous chapter the call delete_copy is added. Note that the call create_version is the export operation of Project_Docs.

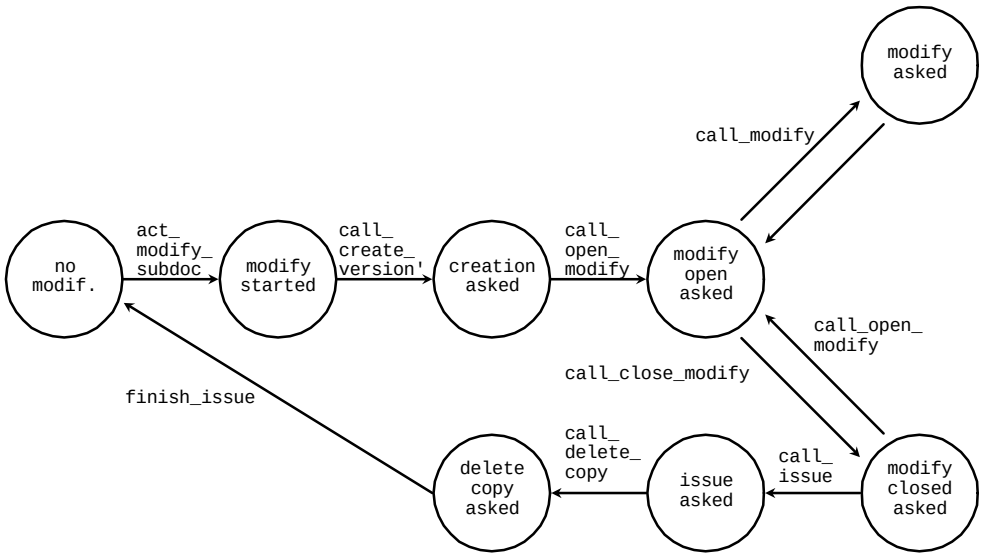

Figure 68 int_modify_subdoc; STD of the internal behavior {new}

The STD of the internal behavior of int_modify_subdoc is almost identical to int_modify_doc. However this internal behavior is activated by modify_subdoc instead of modify_doc. Furthermore the call create_version´ is the export operation of Quality_Plan.

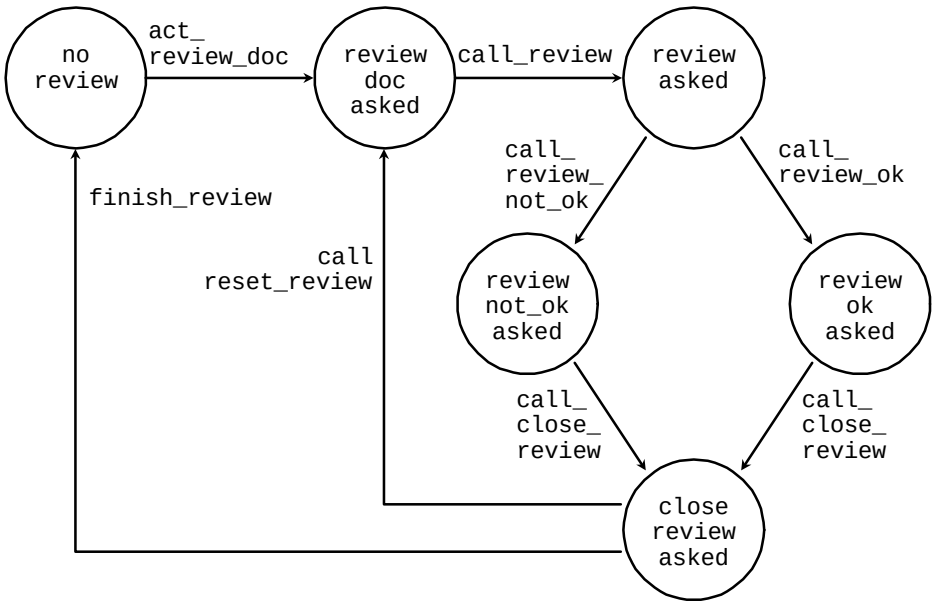

Figure 69 int_review_doc; STD of the internal behavior

There are no changes made to int_review_doc compared to the previous chapter.

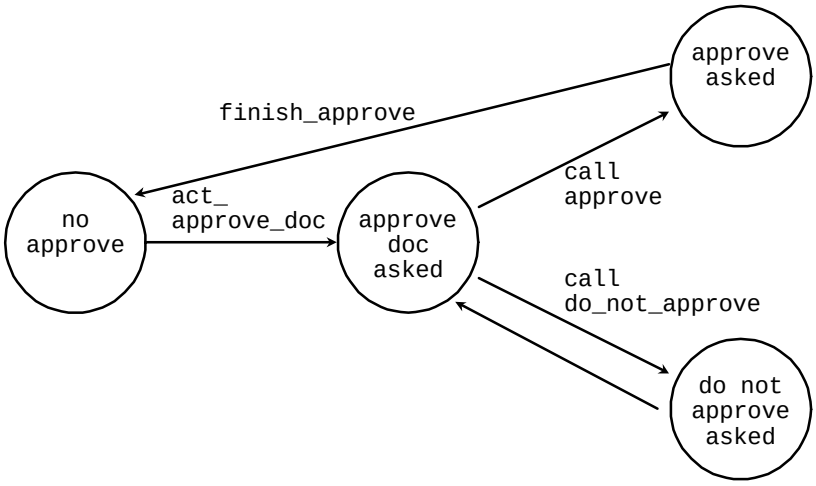

Figure 70 int_approve_doc; STD of the internal behavior

There are no changes made to int_approve_doc compared to the previous chapter.

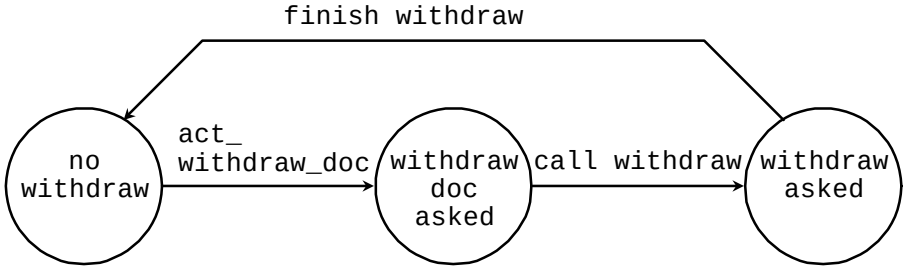

Figure 71 int_withdraw_doc; STD of the internal behavior

There are no changes made to int_withdraw_doc compared to the previous chapter.

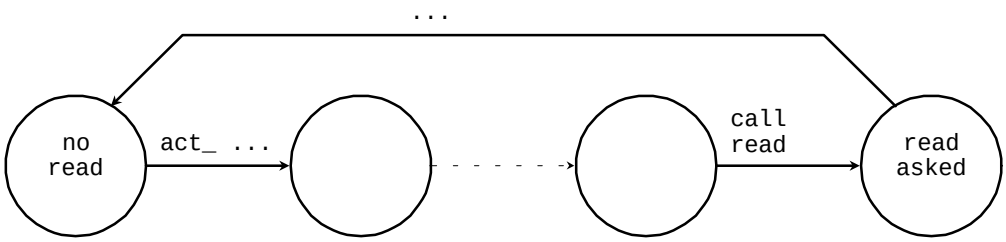

Figure 72 int_. . .; STD of the internal behavior

There are no changes made to int_... compared to the previous chapter.

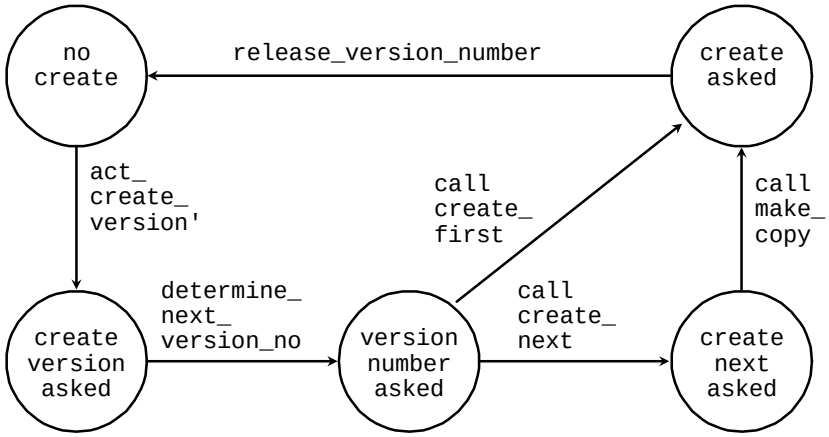

Figure 74 int_create_version´; STD of the internal behavior {new}

Compared to the previous chapter the call make_copy is added. Its purpose is to make a copy of itself (the quality plan document) before the quality plan is modified. The copy remains available to all readers until the new version of the quality plan is issued. This complies with the requirement that only one version of a quality plan may exist. The operation make_copy is called from within the operation create_next (polymorphism).

The next figure is almost identical to this figure. The difference is that create_version is activated by Project_Docs, and create_version´ is activated by Quality_Plan. The internal behavior int_create_version prepares the master quality plan, so it can be modified. The internal behavior int_create_version´ prepares a sub-document of the quality plan, so it can be modified. Note that the master quality plan also serves as a container for all its sub-documents. Other than that the behaviors of these operations are identical.

3.3.5 Subprocesses and traps

In this chapter the coordination of the parallel processes will be modeled. The previously modeled STD's of the external behaviors become the manager processes of the also previously modeled internal behaviors. The internal behaviors contain the subprocesses and traps concerning their manager process.

Similar to document control, which has been modeled in the previous chapter, the subprocesses and traps are indicated with S-# and t-#. Wherever the manager processes or the subprocesses and traps are identical with the models discussed in previous chapter 'Document Control', the same numbers will be used. New or changed subprocesses and traps use numbers ≥ 100.

The following manager processes are identified for quality plan documents:

- Author;
- Reviewer;
- Project_Docs;
- Quality_Plan and Quality_Plan as Project_Docs.

The manager process Reviewer and its subprocesses and traps remain unchanged compared to the models presented in chapter 'Document Control' and will therefore not be modeled again in this chapter.

Minor changes apply to the manager processes Author and Project_Docs or to their subprocesses and traps. More significant changes apply to the manager process Quality_Plan and some of its subprocesses and traps, compared to ISO_Document. These processes will be presented in the remaining part of this chapter.

Author

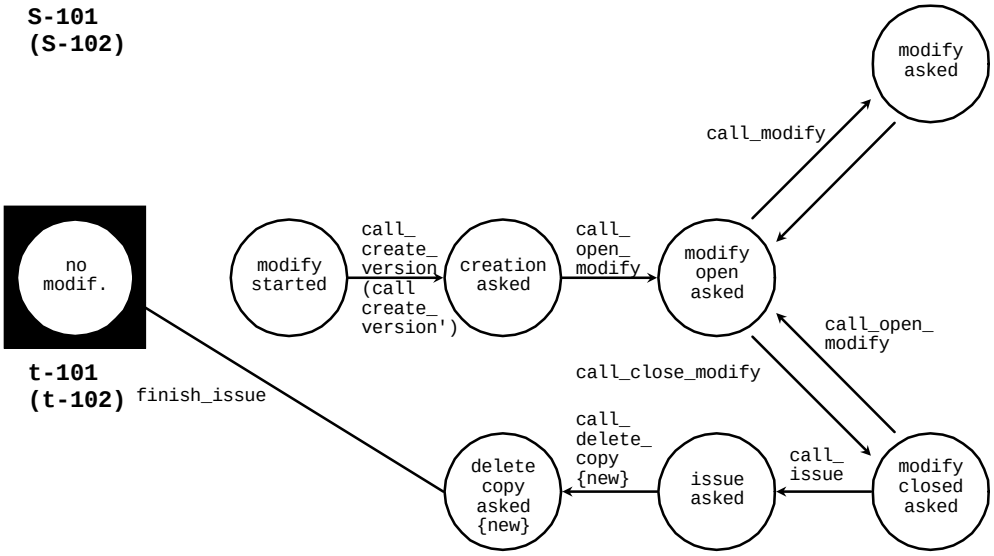

Figure 75 int_modify_(sub)doc's subprocesses and traps w.r.t. Author

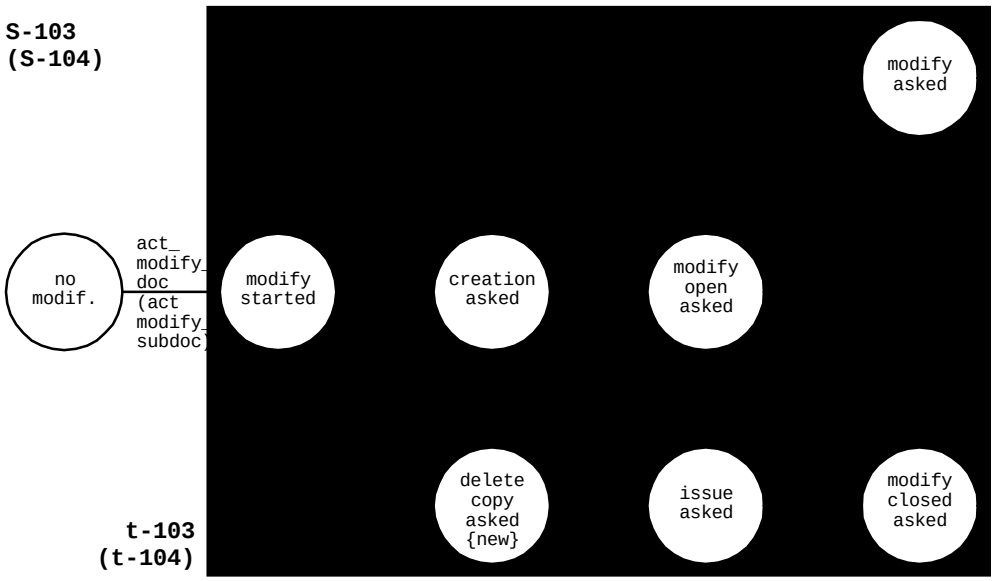

Figure 76 int_modify_(sub)doc's subprocesses and traps w.r.t. Author

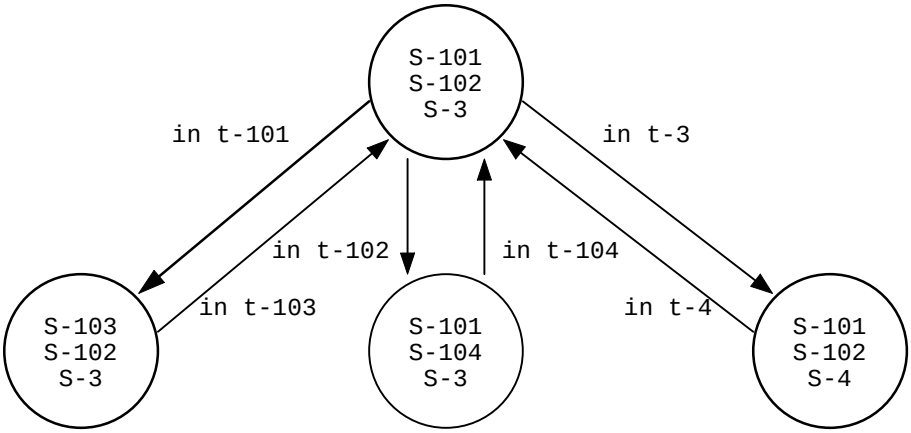

Figure 77 Author; manager of int_modify_(sub)doc and int_withdraw_doc

The employee processes of Author are:

- int_modify_doc;
- int_modify_subdoc;
- int_withdraw_doc.

Because the subprocesses and traps of int_modify_doc and int_modify_subdoc are almost identical, these subprocesses and traps are drawn in the same figures. The only differences are:

- int_modify_doc is activated by act_modify_doc and int_modify_subdoc is activated by act_modify_subdoc;
- int_modify_doc contains the operation call create_version (a call to Project_Docs) and int_modify_subdoc contains the operation call create_version´ (a call to Quality_Plan).

The subprocesses and traps of int_modify_subdoc are enclosed between brackets, as well as the operations that are different compared to the operations of int_modify_doc.

The subprocesses and traps of int_withdraw_doc are the same as modeled in the previous chapter ‘Document Control’.

The state {S-101, S-102, S-3} is the neutral state of author and as soon as the employee processes enter the traps t-101 or t-102 or t-3 a transition is possible to either state {S-103, S-102, S-3} or {S-101, S-104, S-3} or {S-101, S-102, S-4} (modify_doc or modify_subdoc or withdraw_doc). As soon as the employee processes enter the traps t-103 or t-104 or t-4 the manager returns to its neutral state.

Project_Docs

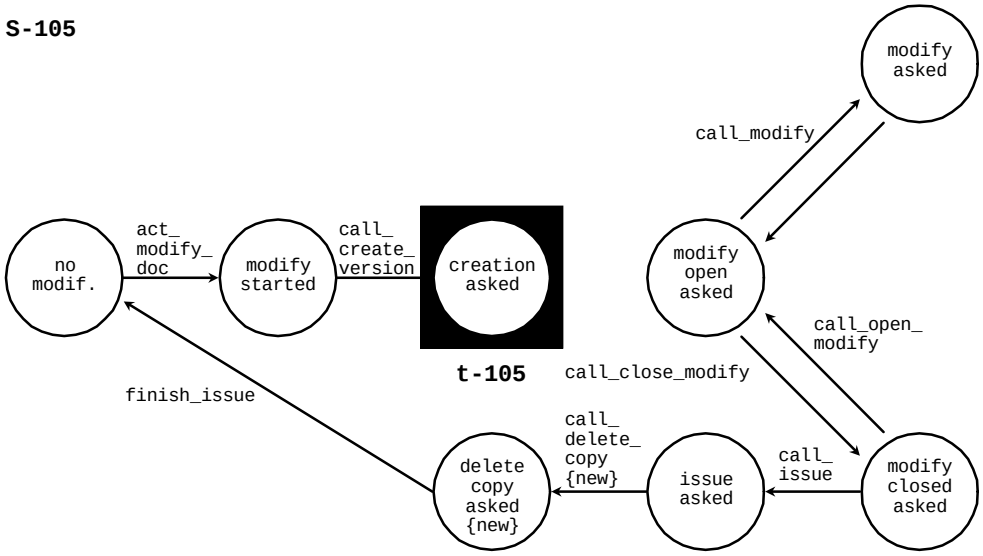

Figure 78 int_modify_doc's subprocesses and traps w.r.t. Project_Docs

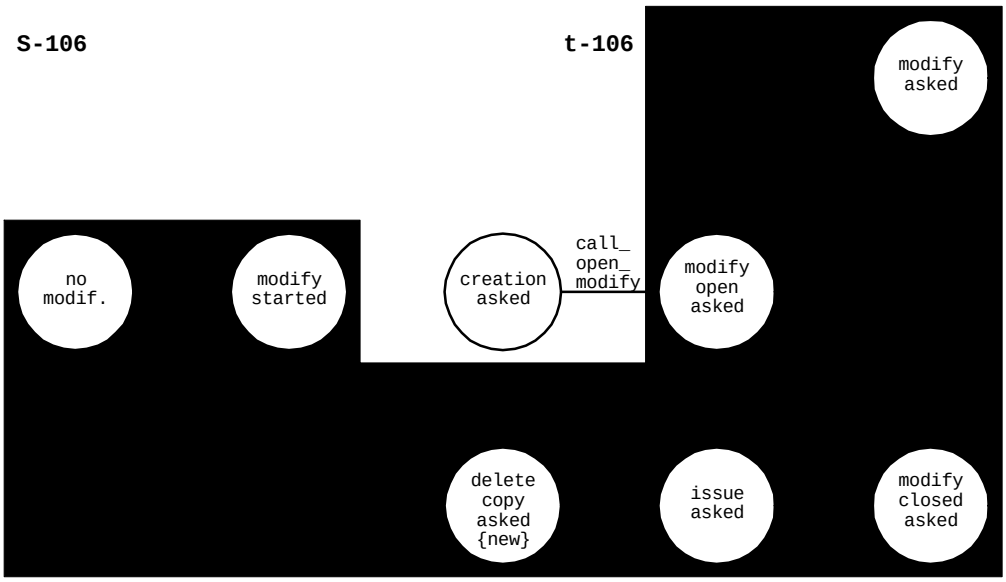

Figure 79 int_modify_doc's subprocesses and traps w.r.t. Project_Docs

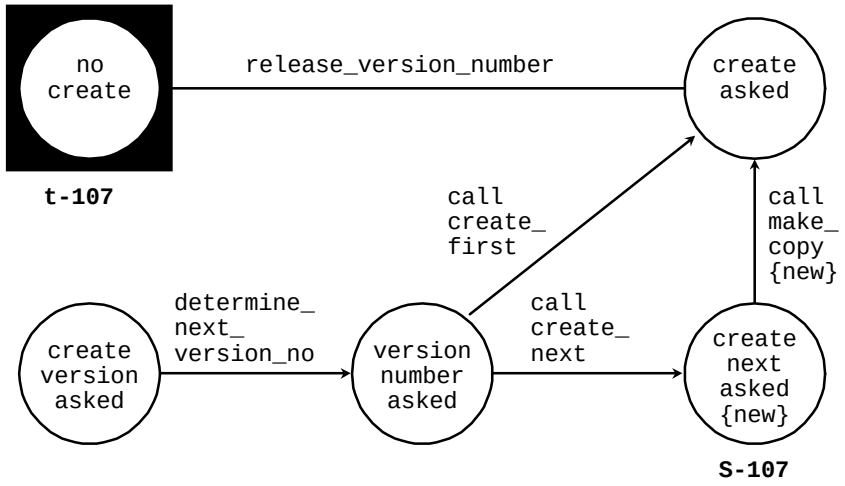

Figure 80 int_create_version's subprocesses and traps w.r.t. Project_Docs

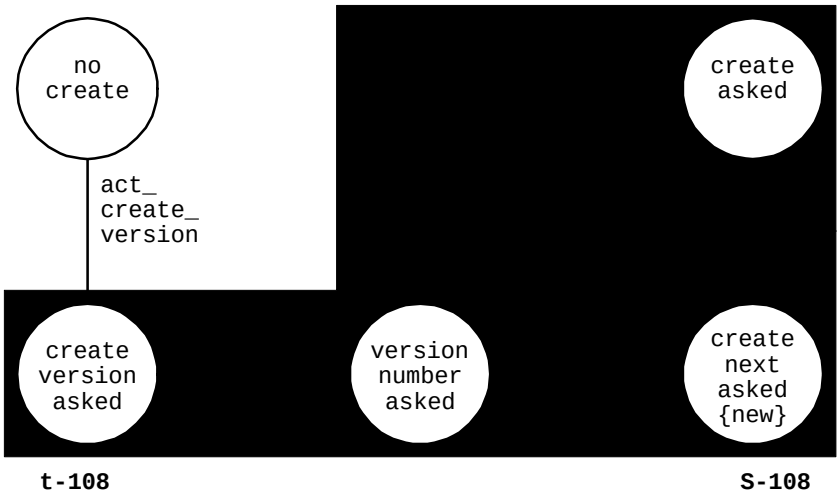

Figure 81 int_create_version's subprocesses and traps w.r.t. Project_Docs

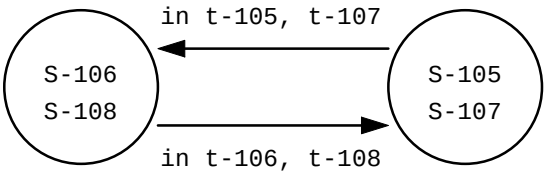

Figure 82 Project_Docs; manager int_modify_doc and int_create_version

The employee processes of Project_Docs are:

- int_modify_doc;
- int_create_version.

The manager Project_Docs is essentially the same as modeled in the previous chapter ‘Document control’. The employee processes however are slightly different, although these differences do not effect the nature of the subprocesses and traps with respect to this manager.

The employee process int_modify_doc has the additional state delete copy asked and a transition to this state. This additional state however has no effect on the subprocesses S-9 and S-10 and traps t-9 and t-10 presented in chapter ‘Document control’. For formal reasons these subprocesses and traps are redrawn and renumbered to the respective subprocesses S-105 and S-106 and traps t-105 and t-106.

The same applies to the employee process int_create_version. Although this employee process contains the additional state create next asked and a transition to this state, its subprocesses S-107 and S-108 and traps t-107 and t-108 are comparable with the respective subprocesses S-11 and S-12 and traps t-11 and t-12.

The resemblance of this manager process, its subprocesses and traps with the model presented in the previous chapter was expected, since no changes have been made to the operations of Project_Docs.

Quality_Plan

Quality_Plan is the manager process of the employee processes:

- int_create_version;
- int_modify_doc;
- int_review_doc; (see subprocesses S-20, S-21, S-22, S-23, S-24 and traps t-20, t-21a, t-21b, t-22, t-23, t-24)
- int_approve_doc; (see subprocesses S-25, S-26, S-27 and traps t-25, t-26a, t-26b, t-27)
- int_...; (see subprocesses S-28, S-29 and traps t-28, t-29)
- int_withdraw_doc. (see subprocesses S-30, S-31 and traps t-30, t-31)

Because the subprocesses and traps of the last four employee processes are identical to the subprocesses and traps presented in chapter ‘Document Control’, the numbers of the subprocesses and traps refer to that chapter.

Project_Docs as Quality_plan is the manager process of the employee processes:

- int_create_version´;
- int_modify_subdoc.

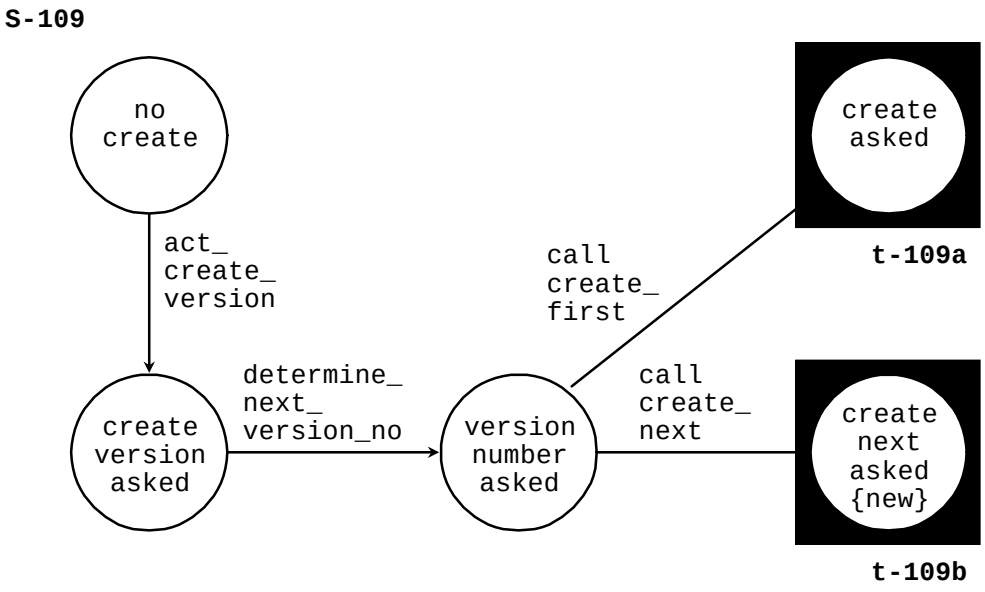

Figure 83 int_create_version's subprocesses and traps w.r.t. Quality_Plan

Figure 84 int_create_version's subprocesses and traps w.r.t. Quality_Plan

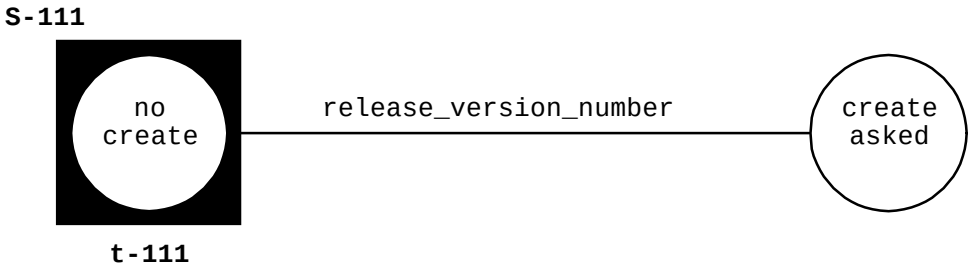

Figure 85 int_create_version's subprocesses and traps w.r.t. Quality_Plan

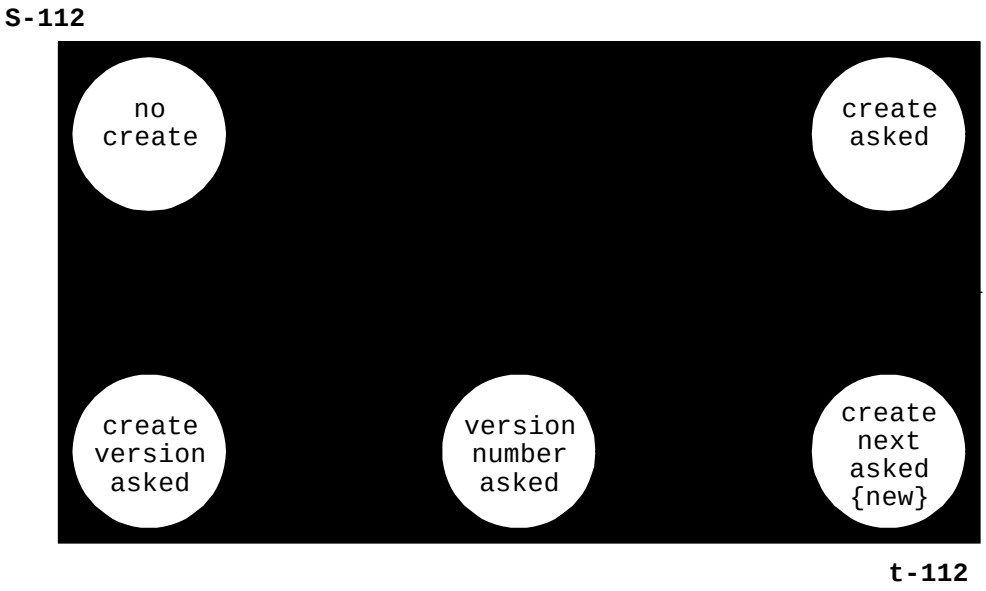

Figure 86 int_create_version's subprocesses and traps w.r.t. Quality_Plan

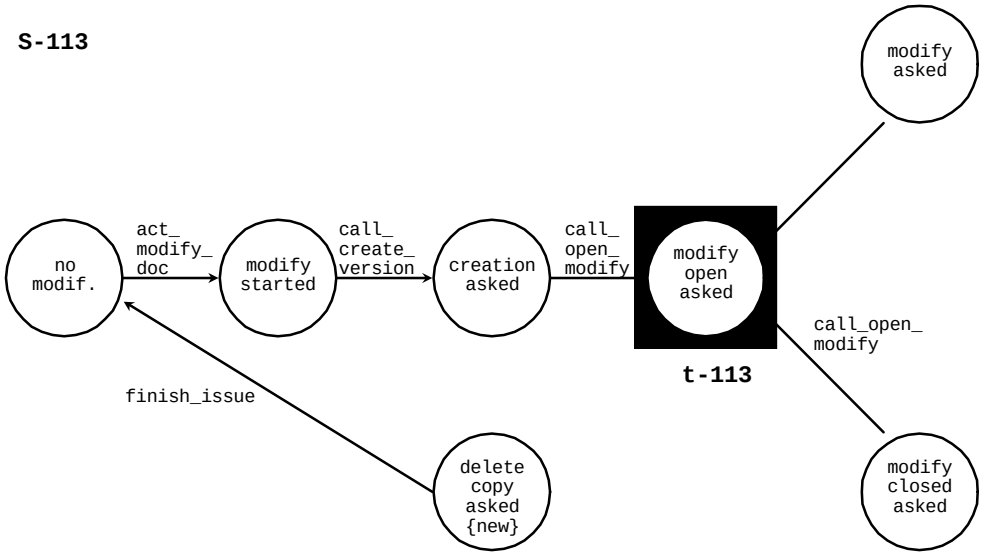

Figure 87 int_modify_doc's subprocesses and traps w.r.t. Quality_Plan

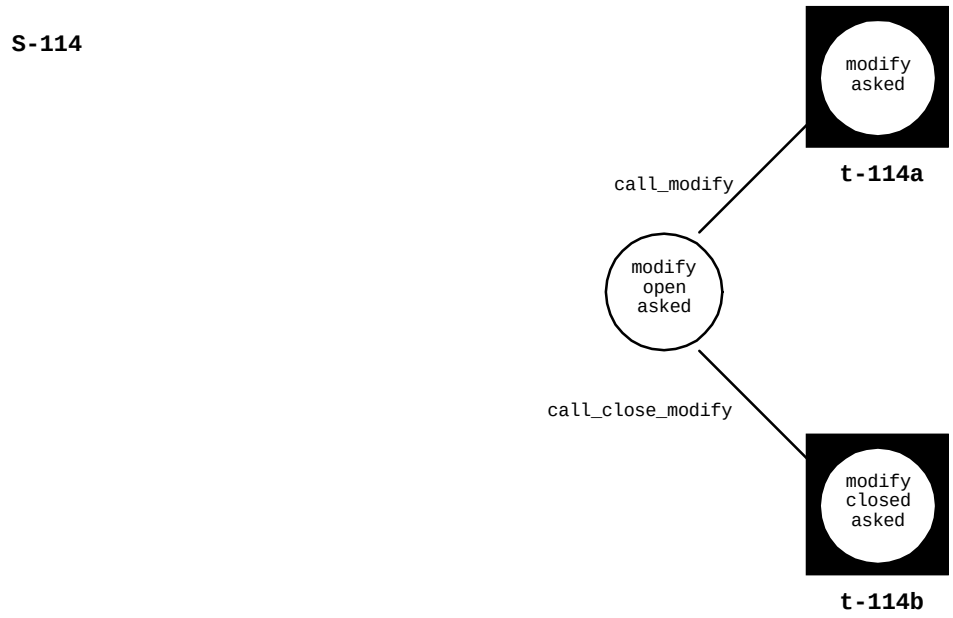

Figure 88 int_modify_doc's subprocesses and traps w.r.t. Quality_Plan

Figure 89 int_modify_doc's subprocesses and traps w.r.t. Quality_Plan

Figure 90 int_modify_doc's subprocesses and traps w.r.t. Quality_Plan

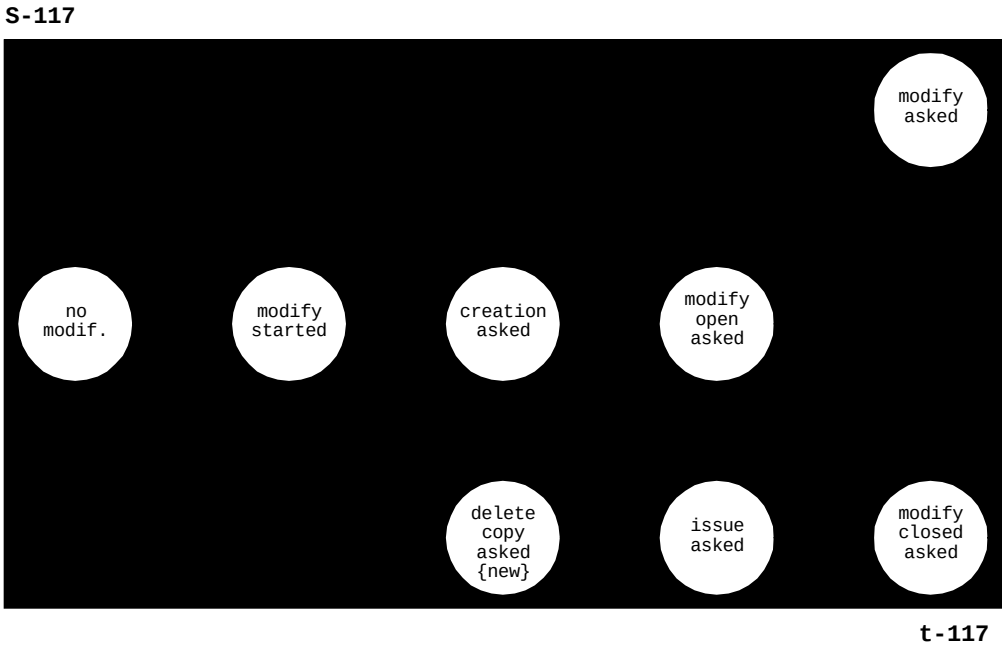

Figure 91 int_modify_doc's subprocesses and traps w.r.t. Quality_Plan

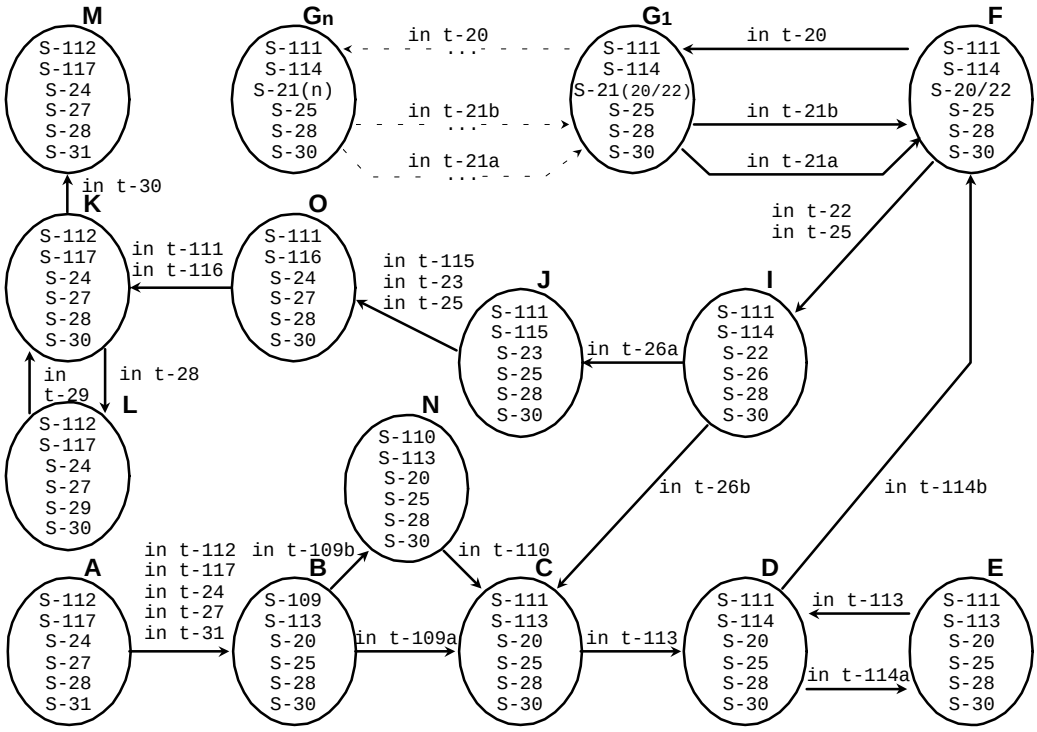

Figure 92 Quality_Plan; Manager of int_create_version, int_modify_doc, int_review_doc, int_approve_doc, int_. . . and int_withdraw_doc

For reference purposes the states of the manager are labeled A through O. Compared to the manager ISO_Document the manager Quality_Plan has the two additional states N and O.

State A of the manager process represents the so-called ‘all-subprocesses’ of its employee processes. The transition from A to B is a kind of initialization process of Quality_plan and confines its employee processes to the subprocesses prescribed in B. The only employee processes that are different compared to the employee processes of ISO_Document are int_create_version and int_modify_doc. These two employee processes will be discussed in more detail below, the other employee processes have already been discussed in the previous chapter.

The employee process int_create_version has the following subprocesses and traps w.r.t. Quality_Plan: S-109, S-110, S-111, S-112 and t-109a, t-109b, t-110, t-111, t-112. It has the additional state create next asked and transition call make_copy compared to its states and transitions modeled in the previous chapter. All its subprocesses and traps, except the subprocess S-110 and traps t-109b and t-110, are more or less comparable to the subprocesses and traps presented in the previous chapter.

In B of Quality_Plan the employee process int_create_version is in subprocess S-109. When the employee process enters trap t-109a a transition is made from subprocess S-109 to S-111 (create_first), or when it enters trap t-109b a transition is made from subprocess S-109 to S-110 (create_next). Note that this latter transition is caused by the modified operation create_next, which in turn calls the operation make_copy. When the trap t-110 is entered a transition is made from subprocess S-110 to S-111. Note that the call make_copy is made from the same object the manager belongs to. During the transitions from subprocess S-109 to S-111 (create_first) or S-110 to S-111 (make_copy) the attributes fDocumentID, fVersionNumber and fStatus are updated. In case of the transition from subprocess S-110 to S-111 a copy of the original document is made and a pointer is maintained to that copy before the attributes are updated (see next chapter OFD). From C through O of Quality_Plan the employee process int_create_version remains in subprocess S-111. In O of Quality_plan and when int_create_version has entered its trap t-111 a transition is possible from subprocess S-111 to S-112 (provided the other employee processes have entered their respective traps). From K onwards the employee process

int_create_version remains in its ‘all subprocess’ S-112 (that is after the document is issued and the copy of the original document is deleted).

The employee process int_modify_doc has the following subprocesses and traps w.r.t. Quality_Plan: S-113, S-114, S-115, S-116, S-117 and t-113, t-114a, t-114b, t-115, t-116, t-117. It has the additional state delete copy asked and transition call delete_copy compared to its states and transitions modeled in the previous chapter. All its subprocesses and traps, except the subprocess S-116 and trap t-116, are more or less comparable to the subprocesses and traps presented in the previous chapter.

In B and C of Quality_Plan the employee process int_modify_doc is in subprocess S-113. In C and when int_modify_doc has entered its trap t-113 a transition is made from subprocess S-113 to S-114 (open_modify). In D the employee process may enter either trap t-114a or t-114b. In trap t-114a a transition is made from subprocess S-114 to S-113 (modify) and in trap t-113 (E) back to subprocess S-114. In D and when the employee process has entered trap t-114b (close_modify) it remains in this trap for some time. However it allows the manager to handle the subprocesses of other employees (F through I). By the condition of some other employee process the manager may make a transition from I to either C or J. Since int_modify_doc was locked in trap t-114b it can change its subprocess to either S-113 or S-115. In J and when int_modify_doc enters trap t-115 (issue) in subprocess S-115 it makes a transition to S-116. In O and when int_modify_doc enters trap t-116 in subprocess S-116 (delete_copy) it makes a transition to its ‘all subprocess’ S-117 and remains in that subprocess.

So far the manager process Quality_Plan has been discussed. The manager process Quality_Plan as Project_Docs is by inheritance also part of Quality_Plan and is now presented.

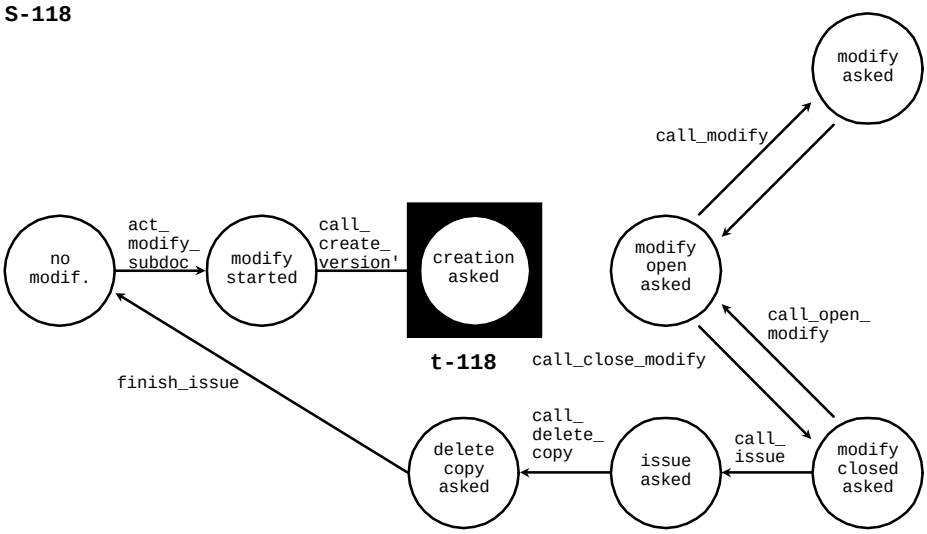

Figure 93 int_modify_subdoc's subprocesses and traps w.r.t. Quality_Plan as Project_Docs

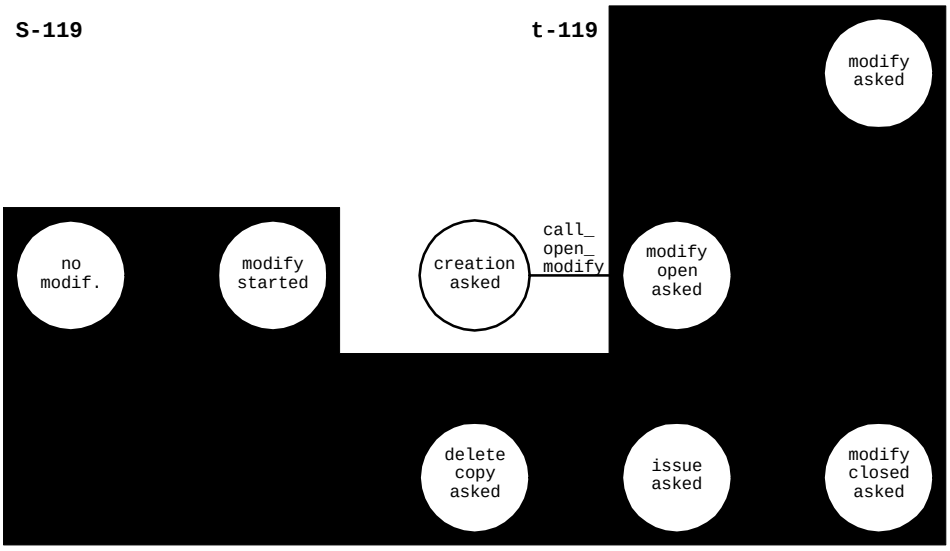

Figure 94 int_modify_subdoc's subprocesses and traps w.r.t. Quality_Plan as Project_Docs

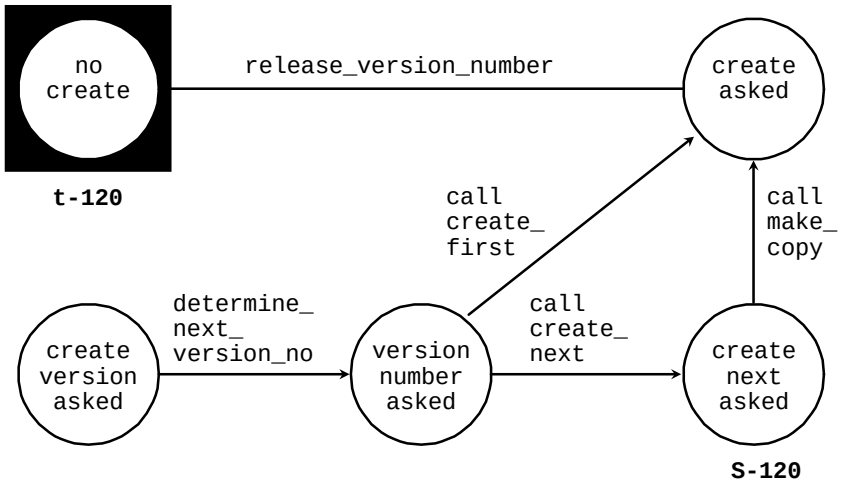

Figure 95 int_create_version´'s subprocesses and traps w.r.t. Quality_Plan as Project_Docs

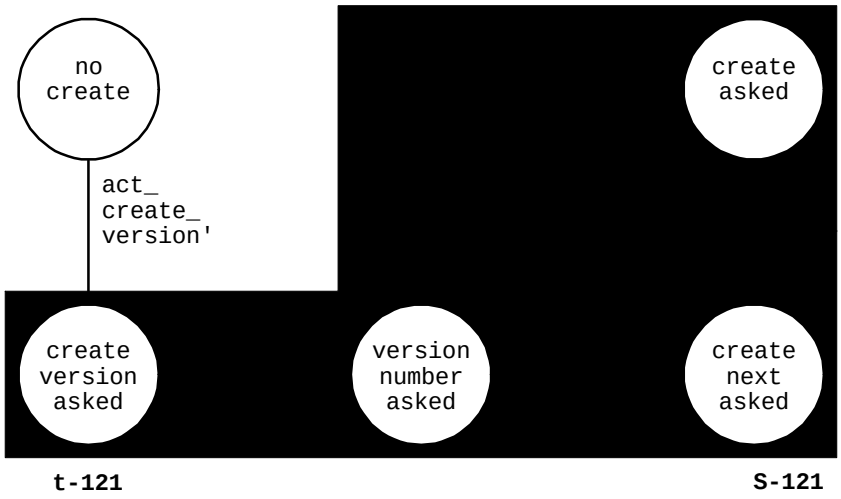

Figure 96 int_create_version´'s subprocesses and traps w.r.t. Quality_Plan as Project_Docs

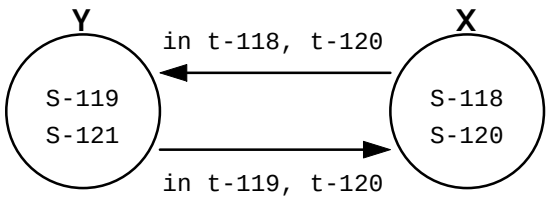

Figure 97 Quality_Plan as Project_Docs; manager of int_modify_subdoc and int_create_version´

The employee processes of Quality_Plan as Project_Docs and its subprocesses and traps are comparable with the employee processes of Project_Docs and its subprocesses and traps. The minor differences are shown in the table below:

Quality_Plan as Project_Docs	**Project_Docs**
int_modify_subdoc	int_modify_doc
int_create_version´	int_create_version
act_modify_subdoc	act_modify_doc
act_create_version´	act_create_version
call create_version´	call create_version
Subprocesses:	Subprocesses:
S-118, S-119, S-120, S-121	S-105, S-106, S-107, S-108
Traps:	Traps:
t-118, t-119, t-120, t-121	t-105, t-106, t-107, t-108

Project_Docs creates an instance of the master quality plan, whereas Quality_Plan as Project_Docs creates an instance of a sub-document of a quality plan. Both instances however belong to the same class Quality_Plan.

The states of Quality_Plan as Project_Docs are labeled X and Y. The states X and Y correspond with all the states of Quality_Plan, except no transition from X to Y is possible in the states A and M. This means for instance that B and X is the state {S-109, S-113, S-20, S-25, S-28, S-30, S-118, S-120}. In other words in all states of Quality_Plan, except A and M, the master quality plan (instance of Quality_Plan) can be extended with sub-documents. Each sub-document is another instance of Quality_Plan. This also means that sub-documents may be extended with other sub-documents (again where each sub-document is another instance of Quality_Plan). Therefore it is not necessary to model the sub-documents any further, since they are implicitly modeled by Quality_Plan.

3.3.6 OFD

Finally the OFD's are presented concerning quality plans. Since at present no formalism exists within SOCCA for the OFD's, the process perspective (transitions and the effect on the data) is shown in the table below.

(call_modify_doc) - external modify_doc act_modify_doc	Inserts an instance of modifies relationship; modifies.status := passive
(call_modify_subdoc) - external {new} modify_subdoc act_modify_subdoc	Inserts an instance of modifies relationship; modifies.status := passive
(call_withdraw_doc) - external withdraw_doc act_withdraw_doc	Inserts an instance of withdraws relationship
(call_review_doc) - external review_doc act_review_doc	Inserts an instance of reviews relationship; reviews.NoOfReviewers := n reviews.ReviewOk := 0 reviews.ReviewNotOk := 0
(call_approve_doc) - external approve_doc act_approve_doc	Inserts an instance of approves relationship
call_create_version create_version act_create_version	Nothing: effect is postponed until after either call_create_first or call_create_next
call_open_modify open_modify act_open_modify	Updates modifies.status := active
call_modify modify act_modify	Updates Document.fContent

call_close_modify modify act_modify	Updates modifies.status := passive
call_issue issue act_issue	Updates: Document.fDocumentTitle Document.fAuthor Document.fIssueDate ISO_Document.fStatus := readable
call_delete_copy {new} delete_copy act_delete_copy	If fVersionNumber > 1 then deletes instance corresponding to earlier version
call_withdraw withdraw act_withdraw	Updates Document.fStatus := obsolete Deletes instance of withdraws
call_review review act_review	Nothing: effect is postponed until after call_review_ok or call_review_not_ok
call_review_ok review_ok act_review_ok	Updates: INC(reviews.ReviewOk) {= i} ISO_Document.fReviewedBy[i] ISO_Document.fReviewDate[i] ISO_Document.fRevComment[i]
call_review_not_ok review_not_ok act_review_not_ok	Updates: INC(reviews.ReviewNotOk) {= i} ISO_Document.fReviewedBy[i] ISO_Document.fReviewDate[i] ISO_Document.fRevComment[i]
call_approve approve act_approve	Updates: ISO_Document.fApprovedBy ISO_Document.fApprovalDate Deletes instance of approves
call_do_not_approve do_not_approve act_do_not_approve	Nothing
call_read read act_read	Nothing

call_create_first create_first act_create_first	Creates instance of Quality_Plan with: Document.fDocumentID ISO_Document.fVersionNumber := 1 ISO_Document.fStatus := not_available
call_create_next {changed} create_next act_create_next	Nothing: effect is postponed until after call make_copy
call_make_copy {new} make_copy act_make_copy	Makes copy of current instance of Quality_Plan and maintains pointer to this copy. Next it updates the attributes of current instance of Quality_Plan, where: Document.fDocumentID := (unchanged) INC(ISO_Document.fVersionNumber) ISO_Document.fStatus := not_available (note that copy now is available to readers)
call_close_review close_review act_close_review	Nothing: effect is postponed until after call_approve or call_reset_review
call_reset_review reset_review act_reset_review	Resets: ReviewOk := 0 ReviewNotOk := 0

Figure 98 OFD; Quality Plan

Note: Document, ISO_Document and Quality_Plan is used to distinguish between these classes. However they refer to the same instance. The instance of Quality_Plan inherits its properties from the superclasses ISO_Document and Project_Docs.

3.4 Quality Planning

In this chapter only a partial example will be given on how a quality plan can be used. The only ISO requirement is that a quality plan must be made at the start of a project and updated before or at the beginning of each project phase. This requirement applies to all types of life cycle models used, for instance a waterfall model or prototyping.

This example case will be restricted to the situation where a test plan is created. This test plan is a sub-document of the master quality plan. The same role classes will be used as in the ISWP-6 example, which are Project_Manager, QA_Engineer and Design_Engineer. The design engineer writes the test plan and for reference purposes he uses (reads) the master quality plan document. The QA engineer reviews the test plan and the project manager approves the document. In a much later stage of the project a test team will use the test plan. The project manager, QA engineer and design engineer have many other tasks besides the above mentioned tasks, however these tasks will not be modeled in this example.

The role classes Author, Reviewer and Reader will be assigned to the above mentioned functions as follows:

```
Project_Manager : Reviewer (uses approve_doc)
QA_Engineer     : Reviewer (uses review_doc)
Design_Engineer :   Author   (uses   modify_subdoc   and
   withdraw_doc)
                  Reader
```

Although Project_Manager, QA_Engineer and Design_Engineer are new classes, in this example case they do not contain any attributes or export operations, but only use the export operations from the earlier designed classes. Therefore the class diagrams are not shown again. The import/export diagram (uses relations) and the import list are shown in the following two figures.

Because this example case is restricted to the creation (modification) of a sub-document, the class Project_Docs and the uses 3 and uses 7 relations are not drawn. Also the export operation modify_doc is not included. Note that, as explained in the previous chapter, these uses relations are only required to modify the master quality plan.

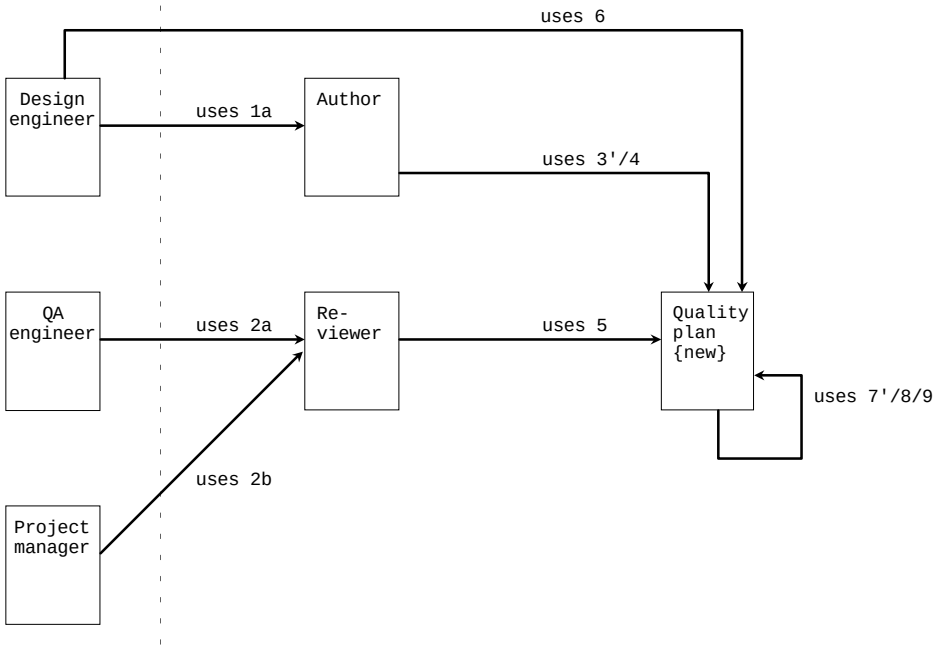

Figure 99 Import/export diagram

As can be seen in the import/export diagram, all uses relations to the right of the dotted line are identical to the uses relations explained in the previous chapter (except that the uses 3 and uses 7 relations are left out). To the left of this dotted line are the new classes Design_Engineer, QA_Engineer and Project_Manager. The uses 2 relation from the previous chapter is now split into a uses 2a and uses 2b relation, to illustrate which export operations are used by the QA_Engineer and the Project_Manager. For the same reason the uses 1 relation is modified into a uses 1a relation, since the Design_Engineer does not call all export operations from the class Author.

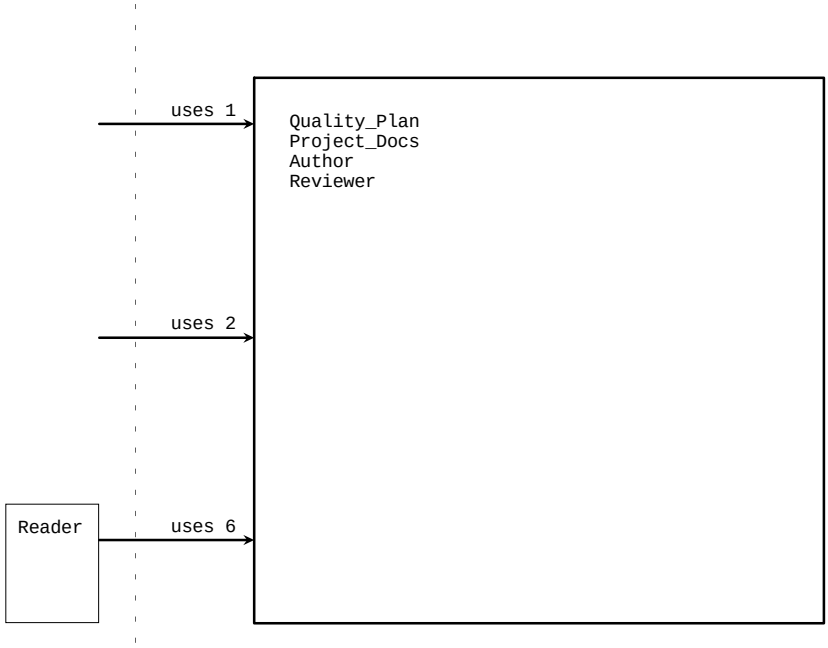

Figure 101 Reduced import/export diagram

```
uses 1a
  withdraw_doc
  modify_subdoc
uses 2a
  review_doc
uses 2b
  approve_doc
uses 3´
  create_version
uses 4
  open_modify
  modify
  close_modify
  issue
  withdraw
uses 5
  review
  review_ok
  review_not_ok
  approve
  do_not_approve
uses 6
  read
uses 7´
  create_first
  create_next
uses 8
  close_review
  reset_review
uses 9
  make_copy
  delete_copy
```

Figure 100 Import list

```
uses 1                          uses 6
  modify_doc                      read
  withdraw_doc
  modify_subdoc
uses 2
  review_doc
  approve_doc
```

Figure 102 Reduced import list

As can be seen by the example case, only a few uses relations (uses 1a, 2a, 2b and 6) are relevant for the new classes (see figures 99 and 100). In general the only possible uses relations concerning quality planning that may be used by the ‘outside world’ are the uses relations 1, 2 and 6. Therefore the import/export diagram and import list can be simplified (reduced) in such a way that all non relevant information is hidden. The ‘reduced import/export diagram’ and ‘reduced import list’ in figures 101 and 102 show all export operations concerning quality planning that may be used by the ‘outside world’. All other uses relations are confined within the big square and can be hidden from the ‘outside world’.

In the example case no export operations have been specified for Design_Engineer, QA_Engineer and Project_Manager. Therefore it is not necessary to continue with the STD’s of the external and internal behaviors. However for future research it seems interesting to extend the behaviors of Design_Engineer, QA_Engineer and Project_Manager and continue on the concepts of the ‘reduced uses relations’.

4 Summary and Conclusions

The topic presented in this master thesis is the investigation of how to combine the ISO-9000 quality standards with software process modeling using the specification formalism SOCCA. To realize this aim the master thesis is divided into two parts. The first part (chapter 2) discussed the ISO-9000 quality standards. The second part (chapter 3) modeled some of the ISO requirements with the specification formalism SOCCA.

The focus of the ISO-9000 quality standards are on business processes with the aim to deliver products and/or services that meet customer's expectations and company's objectives. The ISO requirements are covered in the standards ISO-9001, ISO-9002 and ISO-9003. The most extensive standard is ISO-9001, which covers the quality requirements for the product's total life cycle from design up to and including servicing. The other two standards are a subset of the ISO-9001 quality requirements. Only these three ISO standards contain the quality requirements. The other ISO standards (9000 and 9004 series) contain guidelines for the selection, use and application of the quality requirements. Chapter 2.4 covers ISO-9000-3, 'Quality guidelines for the development, supply and maintenance of software'. This standard was chosen, because it gives a better understanding of how to use the ISO-9001 requirements specific for the IT (information technology) sector.

In the second part of this master thesis two out of the twenty ISO-9001 articles have been modeled, which are 'Document Control' and 'Quality Planning'. In this master thesis specific attention has been paid to the ISO-9001 requirements regarding the life cycle of documents. As a consequence the articles about 'Document Control' and 'Quality Planning' in chapter 2.4 have been modified, since it was necessary to make a clear distinction between the ISO requirements and the ISO guidelines.

The ISWP-6 and ISWP-7 examples give a problem description that (among other things) describe the modification and review of a design (note that a design can be viewed as a document). The document control model presented in this master thesis shows similarities with the model presented in the paper 'SOCCA: Specifications of Coordinated and

Cooperative Activities', which models part of the ISWP-6 example. The major differences, as a result of the ISO requirements, between these two models are:

- In the document control model role classes have been used, such as author, reviewer and reader. These role classes can easily be assigned (when the current model is further extended) to various function classes, for instance a design engineer, QA engineer and project manager.
- The review of documents has been extended considerably. Documents can now be reviewed by many people, either in parallel, in sequence or in a combination of both. Based on the results of the review cycle, a document is either approved or rejected (in which case it needs further modifications). Also the names of the reviewers and approver, the dates and review comments have been added to the model.
- The class reader and the operation withdraw_doc have been added, to comply with the ISO requirement that obsolete documents are removed from all points of issue.

Quality planning was an interesting extension of document control. Document control described the requirements for ISO documents. A quality plan is also an ISO document, but with the additional requirements that only one version of a quality plan may exist and a quality plan may contain zero or more sub-documents. These additional requirements resulted in a further specialization of the class ISO_Document using multiple inheritance and polymorphism. As a consequence this effected the uses relations, some external and internal behaviors, and the corresponding subprocesses, traps and manager processes. Also some processes needed to be added or changed in the OFD.

The above discussion shows that the ISO quality requirements can indeed be modeled with the software process formalism SOCCA. Both the data perspective (EER), behavior and communication perspective (STD + PARADIGM) and the process perspective (OFD) have been modeled, although further research is needed to visualize the process perspective.

The quality planning model shows that SOCCA also allows the use of object oriented features such as inheritance and polymorphism. However due to minor changes in some of the internal behaviors (that is the addition of an extra state and transition), quite some subprocesses, traps and their manager processes needed to be redrawn for formal reasons,

even though most subprocesses and traps were in essence identical to the subprocesses, traps and their manager processes presented in chapter ‘Document Control’.

The manager Quality_Plan and Quality_Plan as Project_Docs allows the creation of sub-documents, where a sub-document is a new instance of the class Quality_Plan. The creation of such new instances is not apparently clear from the presented drawings. For instance it cannot be seen whether the manager Quality_Plan is the master quality plan document or whether it is a sub-document of the quality plan. Perhaps the creation and deletion of instances should be graphically modeled with the OFD’s. This is a topic for further research.

Other suggested topics for future research are:

- Modeling of the other ISO requirements. It should be noted that the ISO-9000 standards have been re-issued during the investigation presented in this master thesis. This means that it is likely that the articles presented in chapter 2.4 needs to be adjusted. Also I have been told that the consistency between the various ISO standards has improved.
- The reader class complies with the ISO requirement in that it will only obtain valid documents. However it is very well possible that after a reader has obtained a valid document, that the document concerned becomes invalid due to the issue of a new version. Quite often in quality procedures within a company a distinction is made between two types of readers, that is:
  a) readers who obtain a copy of a document, but do not need to be informed in case the document concerned becomes invalid (the case presented in this master thesis);
  b) readers who have a ‘controlled copy’ of a document, which means that they are informed as soon as the document concerned is withdrawn or a new version is issued.

  Future research may accommodate case b). This involves the registration of all readers who obtain a ‘controlled copy’ of a document and sending a notification to those readers as soon as the document concerned is either withdrawn or a new version is issued. A problem arises however when these readers move to another position and the registration system does not keep track of such moves (think of a library where a book is borrowed to someone and that person moves to another address without informing the library).
- In the last chapter the concept of ‘reduced uses relations’ was introduced. It hides certain details from the ‘outside world’. For future research it seems interesting to expand

the example presented in that chapter and investigate the effect of information hiding on the behavior and communication perspective and the process perspective.

5 References

1. Engels G., Groenewegen L.: *SOCCA: Specifications of co-ordinated and co-operative activities.* University of Leiden, Department of computer science, November 1993.

2. Groenewegen L.: *Simulatie, Modelling parallel phenomena in Paradigm*. University of Leiden, Department of computer science, 1994.

3. Heimbigner D.: *Software Process Example for ISWP-7*. August 1991.

4. Kellner M.I., Feiler P.H., Finkelstein A., Katayama T., Osterweil L.J., Penedo M.H., Rombach H.D.: *ISPW-6 software process example.* IEEE Computer Society Press, 1991.

5. Kema N.V., Cosso: *ISO 9000 in de IT sector: Richtlijnen voor evaluatie en invoering van kwaliteitssystemen in de informatietechnologiesector.* Lansa Publishing, Leidschendam, 1993.

6. Penedo M.H.: *Life-cycle (sub) process demonstration scenario, 9th International Software Process Workshop (ISPW-9).* March 1994.

7. Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W.: *Object-oriented modeling and design*. Prentice Hall, 1991.

8. Verkoren E.H.: *Een kwaliteitsaudit in IT-organisaties - Doctoraal scriptie informatica*, University of Leiden, Department of computer science, August 1993.

9. Wiley J.: *Software process modelling and technology.* Research Studies Press, 1994.

10. Wulms A.: *Adaptive software process modelling with SOCCA and PARADIGM.* University of Leiden, Department of computer science, April 1994.

6 Appendix

6.1 ISO-References

General:

ISO-8402 Vocabulary, 1989

Guidelines for selection and use:

ISO-9000 Quality management and quality assurance standards - Guidelines for selection and use, 1987.
ISO-9000-2 Quality management and quality assurance standards - Part 2: Generic guidelines for the application of ISO-9001, ISO-9002, and ISO-9003, 1993.
ISO-9000-3 Quality management and quality assurance standards - Part 3; Guidelines for the application of ISO-9001 to the development, supply and maintenance of software, 1991.
ISO-9000-4 Quality management and quality assurance standards - Part 4: Guide to dependability program management, 1993.

Quality assurance standards:

ISO-9001 Quality systems - Model for quality assurance in design / development, production, installation and servicing, 1987.
ISO-9002 Quality systems - Model for quality assurance in production and installation, 1987.
ISO-9003 Quality systems - Model for the quality assurance in final inspection and test, 1987.

Guidelines to the application of quality assurance standards:

ISO-9004 Quality management and quality system elements - Guidelines, 1987.
ISO-9004-2 Quality management and quality system elements - Part 2: Guidelines for services, 1991.
ISO-9004-3 Quality management and quality system elements - Part 3: Guidelines for processed materials, 1993.
ISO-9004-4 Quality management and quality system elements - Part 4: Guidelines for quality improvement, 1993.

Guideline for auditing quality systems:

ISO-10011-1 Guidelines for auditing quality systems - Part 1: Auditing, 1990.
ISO-10011-2 Guidelines for auditing quality systems - Part 2: Qualification criteria for quality systems auditors, 1991.
ISO-10011-3 Guidelines for auditing quality systems - Part 3: Management of audit programs, 1991.

Contents

Figures

Quality Systems in the IT Sector
with the
Specification of Coordinated and Cooperative Activities (SOCCA)

Report number: IR-95-22

B.H. Stappers
Rijks Universiteit Leiden
20 August 1995