Claims-Based Working

Access to business activities

Herman Shao

01-09-2010

Claims-Based Working

Access to business activities

Student:

Name:	Herman Shao
Study:	Msc. Business Information Technology, University Twente

Organization:

Name:	TNO ICT
Supervisor:	Rieks Joosten

University:

Name	University Twente
Supervisors:	Wolter Pieters
	Maria-Eugenia Iacob

Date: 01-09-2010

Version: Final

PREFACE

On 26 October 2009, I moved from Oldenzaal to Groningen to start my final project at TNO ICT. This report is the result of my theoretical and practical work of the past few months.

When I started this project I could not have imagined how much work it would be. Formulating the research questions already took a couple of months, but spending this much time on the research questions paid off in the end. It enabled me to focus my time on the topics that were most important for this research.

This research had a theoretical as well as a practical element as I had to do theoretical research and implement the results in a number of demonstrators. Making the demonstrators gave me a lot of satisfaction as it enabled me to see how the theories would and sometimes would not work in practice. The downside of doing programming work is that it can easily take up a lot of time, giving me less time to spend on this report. I had the feeling that the time I spent on this project flew by. I can hardly imagine that already more than six months have passed by. Now the end is near. I had a great time doing this project, despite of the ups and downs during my project.

I want to thank a number of people who supported me along the way for their mental support and idea sharing.
First of all I would like to thank my supervisors from the university, Wolter Pieters and Maria-Eugenia Iacob. I would like to thank you both for your critical view on this project and steering me in the right direction when I was lost. Thank you for reviewing my report time and again to make it readable and understandable.
Also I would like to thank my supervisor from TNO, Rieks Joosten. You gave me a lot of ideas (sometimes a bit too much) for this project. It was always a great pleasure to exchange and discuss about ideas. I always enjoyed those Friday afternoon “speel uurtjes”, where we, Jeroen Broekhuijsen and sometimes Daniel Boonstra discussed ideas surrounding the *Claims* topic. Jeroen, I would like to thank you for your great support on the demonstrator as you are the expert on the Microsoft’s Claims Based Access Control mechanism.
Finally I would like to thank my friends and my family for their support. Special thanks goes to Su Wan Liong, Kien Ming Lam, and Lisa Pan for reviewing my thesis and correcting all the crooked sentences and typos.

I hope you have fun reading my thesis and hope to give you new insights into organizing business processes in a declarative way.

Groningen, 1 September 2010
Herman Shao

ABSTRACT

In today's dynamic business environment the success of an organization depends on its ability to react to various changes, like shifts in the attitudes of customers or the introduction of new laws. Changes in the business environment can have an impact on the business processes of an organization. The business processes will have to be able to cope with changes and unpredicted situations.

Business processes can be seen as a collection of activities which are ordered in a particular fashion to achieve the goal of the business process. Imperative business processes explicitly state how activities are linked to each other. Thus every possible scenario is explicitly modeled in imperative business processes. Declarative business processes on the other hand describe the scenarios which are not allowed. The activities of a business process are not explicitly linked to each other. The flow through the business process is regulated by business rules. Many declarative business processes use action rules to determine the flow. Action rules determine which activities must (not) be carried out. When changes occur in a business process resulting in an addition or removal of an activity, these action rules have to be changed accordingly.

We propose a way of organizing business processes which uses business rules which do not directly refer to activities. Instead we propose to put data constraints on the individual activities. The constraints that apply to an activity determine what input data an activity requires. In order to determine if an activity is allowed to be performed certain information is needed. The business process solutions we looked at state what kind of data needs to be available, but not where to get this data from. Some data might not be available to the parties directly involved in the business process. A mechanism is proposed that can retrieve that data.

We will call our method of organizing business processes Claims Based Working. The goal of this research is to develop a reference architecture for Claims Based Working and to validate it using a demonstrator. The framework is roughly composed of two parts. The first part is concerned with the way business processes have to be organized and the second part is concerned with gathering information which is used in activities.

In traditional workflows every activity is linked to another activity. This is modeled as a line in a workflow diagram. We want to remove those links between the activities and use constraints to determine when an activity is allowed to be performed. From a security point of view this can be seen as a form or access control. Access to an activity is granted or denied based on the constraints which apply to that activity.

Activities have to be performed by entities which have to be authorized somehow to do so. There are a number of mechanisms which to authorize entities to activities.. We looked at a number of access control mechanisms and formulated an Access Control Meta-model that we use in Claims Based Working.

In our reference framework we combined the Access Control Meta-model with our view on declarative business processes to form Business Activity Access Control (BAAC). In order to make a decision whether or not an activity is allowed to start the constraints concerning the activity are needed and also information about the terms that occur in the constraints.

We use the Claims gathering mechanism of Claims Based Access Control to gather information which is necessary to make an access control decision. By combining the concepts of gathering decision making information using Claims with BAAC Claims Based Working is formed.

To validate our reference architecture a demonstrator for Claims Based Working was made. The basic components of the demonstrator are:

- a component which manages the business processes (Decorum),

- a component which evaluates business rules,
- a component which is able to generate a list of business rules for each request,
- and a component which is able to deliver information which is required to evaluate the business rules.

Two scenarios were used to validate Claims Based Working. The first scenario shows that it is possible to implicitly create a flow through a process by using constraints based on data requirements. The second scenario shows that using Claims certain decision making information can be gathered that otherwise was not available.

Our research has shown that it is possible to create declarative business processes by using data based constraints. The added value of using these constraints is that activities can be added or replaced without changing the constraints that apply to other activities.

The added value of Claim Based Working is that is provides an alternative view on organizing business processes. Claims Based Working enables business processes to be more dynamic. Business processes are easier to adapt, because the constraints for each business activity does not have explicit relations with other activities. Business activities can be added or removed without disrupting the constraints of the other activities.

The declarativeness of Claims Based Working gives the users of the business process more freedom to work according their own insight. The constraints of the activities set the boundaries in which the users can operate.

Using Claims has a number of benefits. The first is that the user of the system can partly decide where the Claims are gathered from. A second benefit is that the gathered Claims are authenticated. The system can assume that the Claims are truthful. A third benefit is that information can be gathered with is unknown by the system and by the user. If the type of Claim is known and the Identity Provider is able to deliver that type of Claim, information can be gathered which was otherwise unavailable to the user and the system.

There are a number of limitations to the research done in this thesis. The scenarios used to validate Claims Based Working were “made up” and possibility to simple compared to existing business processes. Additional validation can be done using existing business processes with a relatively high number of activities.

Another limitation concerns the constraint model that is used. In the current model of CBW only access constraints of business activities are specified. Additional research is needed to explore the usability of other constraints, like constraints on the post conditions of an activity and possibly constraints during the execution of an activity.

Using Claims information can be gathered that can be used to make decision whether or not an activity is allowed to start. There are some limitations with the used mechanism. Claims cannot be used to gather all the decision making information. Some information can be provided by the user and other information is generated by the process itself. The information coming from these sources cannot be gathered using claims, therefore it is an important additional mechanism to gather information, but it is not the only mechanism that has to be used to gather information.

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION AND OBJECTIVE

In today's dynamic business environment the success of an organization depends on its ability to react to various changes, like shifts in the attitudes of customers or the introduction of new laws (Poppendieck and Poppendieck 2006). Changes in the business environment can have impact on different levels in an organization. They can have impact on the operational, tactical or even strategic level, but these changes will likely have an effect on the business processes of the organization. Organizations will have to be able to adapt their business processes accordingly in order to cope with the dynamic business environment of today. The management of business processes is therefore an important issue in organizations. Another related problem that organizations have is that there is often a gap between the way organizations have operationalized their business processes and how the organization would like the business processes to be. A good example of a poorly implemented business process is the purple crocodile scenario. The purple crocodile scenario comes from a television commercial of the insurance company OHRA. The scenario plays out as follows:

The commercial takes place in a swimming pool where a little girl has lost her purple crocodile. Mother and daughter arrive at the service desk a day later where the purple crocodile stands in the corner behind the desk clerk.

The mother says to the man: *"Yesterday my daughter has left her purple crocodile."*

The desk clerk responds by saying with some disinterest: *"Ah yes, a purple crocodile."*
During which he turns around and takes a blank form from a tray and hands it over the mother telling her to fill in the form using block letters.

After a while the mother has filled in the form and gives it back to the desk clerk.
The desk clerk looks at the form with indifference and hands it back saying: *"You forgot to fill in the back."*

After filling in the back of the form the somewhat agitated mother gives the form to the desk clerk.
The clerk gives the form a glance and once again hands it back to the mother saying: *"Hand it in at the recreation service desk tomorrow between nine and ten o'clock."*

A bit surprised the mother tells the desk clerk: *"But the crocodile is just behind you."*
The clerk responds with: *"Yes, I know it's just behind me."*
But he still will not give it back.

From the scenario described above it can clearly be concluded that something went terribly wrong during the implementation of the process. The rigidity of the process resulted in the fact that the little girl did not get her purple crocodile back. This scenario is somewhat exaggerated, but many business processes are implemented in a rigid way that is not able to handle unforeseen scenarios.

In an attempt to prevent the situation like the one described above, we propose an alternative way of implementing and managing business processes that is able to provide a better alignment between the high level business policies and the operationalized business processes. This alternative way should let people decide for themselves what the best way is to fulfill tasks. The concepts that are relevant to our research are declarative business processes, business rules, access control and claims. This thesis brings them together to form a way of organizing business processes which we call Claims-Based Working.

1.1.1 BUSINESS PROCESSES

There are quite some definitions of a business process. Definitions which are highly quoted are those of Hammer and Davenport (Hammer and Champy 1993). They define business processes as:

A business process is a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer.

Another highly quoted definition of a business process is that of (Davenport 1993):

A business process is a structured, measured set of activities designed to produce a specific output for a particular customer or market.

Both these definitions state that a business process consists of a number of activities which are carried out to create something of value to someone. We define an activity as:

An activity consists of number of events which take place at one place by one entity without transition of information between entities.

The activities in a process need to be organized in a way that the produced output is of value to someone. These activities can be organized using a number of different standards and languages, which can be classified into imperative business process languages and declarative business languages.

IMPERATIVE BUSINESS PROCESS LANGUAGES

Imperative business processes languages strictly prescribe the sequence of business activities in a business process, thus not leaving any room for change. Imperative languages start from the inside out by explicitly specifying every possible scenario (Pesic and Aalst 2006). Typically imperative business processes are modeled using workflows. The diagram below depicts the imperative nature of the workflow, which is modeled using the purple crocodile scenario that was introduced in section 1.1. All the possible flows and sequences are modeled, which leaves no room for scenarios that need to deviate outside of this model. In the case of the purple crocodile the lost item was just behind the desk clerk, but this is a scenario which is not modeled in the workflow. This resulted in the fact that the little girl did not get her purple crocodile back.

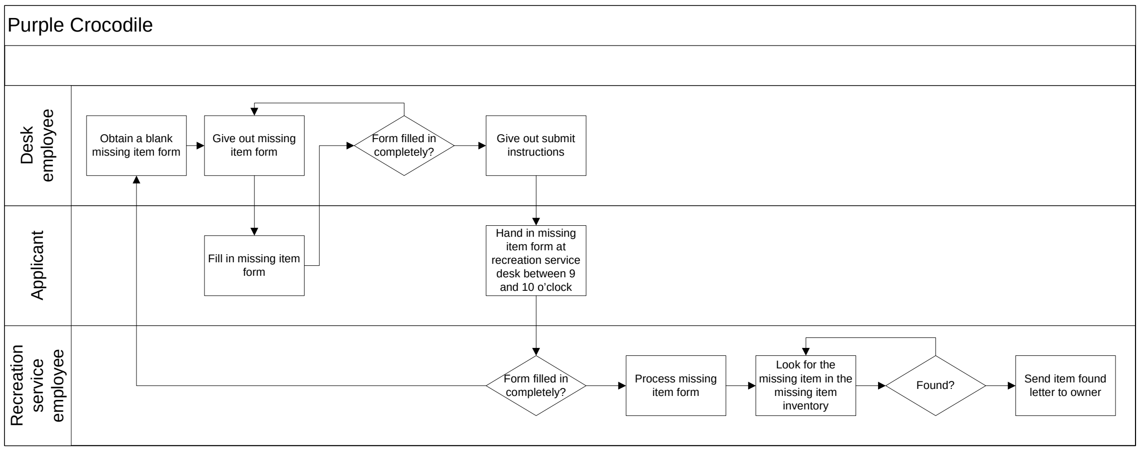

Purple Crocodile
Desk employee
Applicant
Recreation service employee
Obtain a blank missing item form
Give out missing item form
Form filled in completely?
Give out submit instructions
Fill in missing item form
Hand in missing item form at recreation service desk between 9 and 10 o'clock
Form filled in completely?
Process missing item form
Look for the missing item in the missing item inventory
Found?
Send item found letter to owner

Figure 1 Imperative business process

DECLARATIVE BUSINESS PROCESS LANGUAGES

In the paper, "A Declarative Approach for Flexible Business Process Management" (Pesic and Aalst 2006), a new view on business processes is presented. They propose a declarative approach which will make business processes more flexible, by specifying what needs to be done and not the order in which it should be done. Business processes specified in a declarative way start from the outside in. Instead of explicitly stating the flow of activities, the flow is implicitly created using constraints. The constraints set the boundaries in which the business activities can operate. By having boundaries instead of explicit paths, business activities can be executed at any time as long as the constraints are not violated. Constraints are represented by business rules.

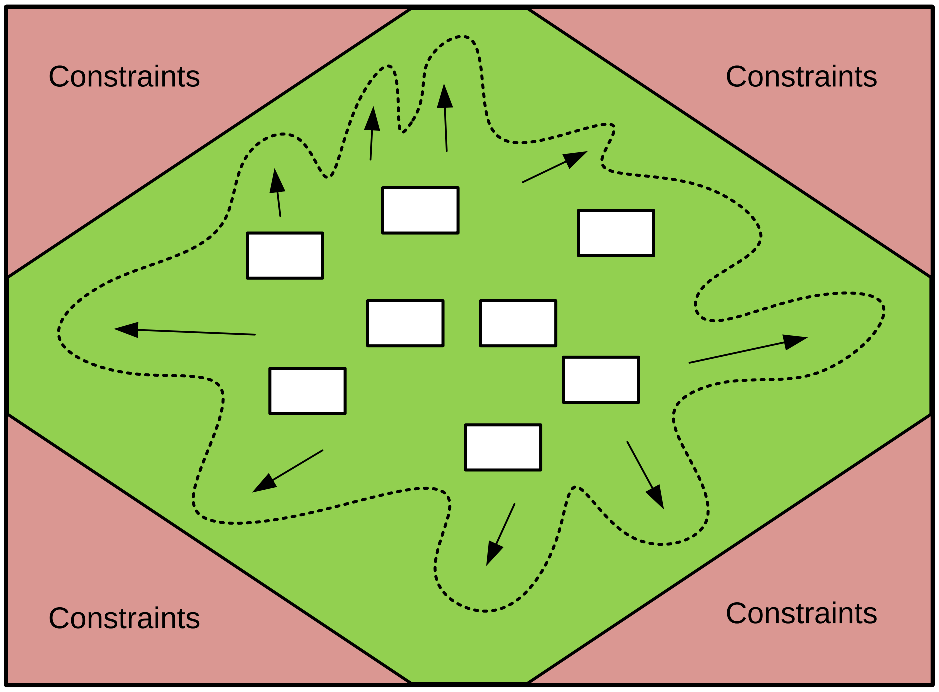


Figure 2 Declarative business process Pesic 2008 (Pesic 2008)

1.1.2 BUSINESS RULES

Business rules provide guidance for decision making. Ronald Ross (Ross 2005) defines a business rule as:

A business rule is a directive intended to influence or guide business behaviour

A business rule is composed of *Terms* and *Facts*. *Terms* are business concepts that have precisely limited meanings. *Facts* are the relations between *Terms*. A business rule uses *Facts* to create guides for business activities.

When a business rule is evaluated certain resources are checked whether or not they fall within a set of parameters. To illustrate this we can consider an ordering process. *Terms* that are related to this process are “customer” and “order”. An example of a *Fact* is: “a customer places an order”. A business rule for an ordering process can be: “A high risk customer may not place rush orders”. A high risk customer is a type of customer.

In this example a customer is classified as a high risk customer if the customer has unpaid bills of three months or older. Rush orders are orders that skip certain steps in the ordering process, like checking the solvency of the customer. When a normal customer places a rush order the rule is checked, and because the rule is not violated, nothing out of the ordinary happens. But when a high risk customer wants to place a rush order, the rule is violated and the order cannot be placed.

The rule in the given example can concern an order intake business activity. Implicitly the actions follow the result of the evaluation of the rule. An order will be placed if the rule is not violated and an order is not allowed to be placed if the rule is violated. The decision resulting from the evaluation of the business rules that pertain to a business activity looks very similar to access control. Depending on the decision made by evaluating the business rules, access is either granted (the activity can start) or denied (the activity is not allowed to start).

1.1.3 ACCESS CONTROL

The purpose of access control is to limit the actions or operations that a legitimate user of a computer system can perform (Sandhu and Samarati 1994). The actions or operations in the case of business processes are the business activities. Only legitimate users should be able to start a business process. The difference with normal access control is that access to a business activity does not just depend on a legitimate user, but also other conditions. Whether a business activity can be carried out or not depends on the business rules which apply to the business activity.

In order to be able to evaluate the business rules information is needed about the terms in the business rules. Consider the following rule: “A high risk customer may not place a rush order.” Information is needed about the customer, in order to determine if that customer is a high risk customer. Information is also needed about the order to determine whether or not it is a rush order. Certain resources will have to be accessed to retrieve this information. These resources can be locally available, like organization’s own databases or from the order form itself. The resources can also be located outside the local domain. Somehow the required information needs to be obtained from these external resources. A possible solution is provided in the access control literature, which is Claims-Based Access Control.

CLAIMS-BASED ACCESS CONTROL

Claims-based Access Control (CBAC) provides a mechanism to gather information which is necessary to make an access control decision. In CBAC terminology, the entity that makes the access control decision is called the *Relying Party* and the entity that requests access is called *Subject*. The *Relying Party* makes access control decisions based on information about the *Subject*. This information is comes from sources other than the *Relying Party* and are represented by *Claims*, that can be defined as (Bertocci 2008):

A statement about an entity made by another entity.

This definition tells us something about who makes the claims. Another definition of claims tells us something about content of claims (Cameron 2005):

A claim is an assertion of the truth of something, typically one which is disputed or in doubt.

Claims as used in this document have the meaning of both these definition, thus forming the definition:

A claim is an assertion of the truth of something, typically one which is disputed or in doubt, about an entity made by another entity.

The *Relying Party* does not make claims itself, but uses a third entity called the *Identity Provider* to make *Claims* about the *Subject*. Based on these *Claims* a *Relying Party* can make access control decisions with concerning the *Subject*. In short, the *Relying Party* uses claims, which are issued by the *Identity Provider*, to make some decision about the *Subject*.

An example bout buying a beer at a bar can be used to explain how CBAC works in general. In the Netherlands, you have to be at least 16 years old to buy a low alcoholic beverage like a beer. Imagine that Susie (*subject*) wants to buy a beer. She walks up to the bartender and asks the bartender: "Can I buy a beer?" The bartender (*Relying Party*) has to decide whether or not Susie is authorized to buy a beer, meaning whether she is older than 16. The bartender cannot really decide whether or not Susie is older than 16 years and he surely cannot trust Susie on her word. The bartender tells Susie that he needs something that, with some certainty, can prove that she is at least 16 years old.

Susie needs to get something that will prove that she is authorized to buy a beer. Luckily she passed her driving exam the day before and the CBR (*Identity Provider*) gave her a certificate which she used to get her driving license. In the Netherlands you have to be at least 18 years old to take a driving exam. Susie shows her driving license to the bartender. The bartender looks at the driving license and it looks real. The bartender (*Relying Party*) trusts that the CBR (*Identity Provider*) only lets people who are 18 years or older to take the exam. The bartender has now determined that Susie is at least 18 years old, so she definitely is old enough to buy a beer. After the bartender received this proof, he can serve the beer to Susie.

1.1.4 CLAIMS-BASED WORKING

In a traditional workflow the activities are chained together. But when is a business activity allowed to execute? In traditional workflows, activity *B* is simply triggered by activity *A* and when an activity has finished, it triggers activity *C*.

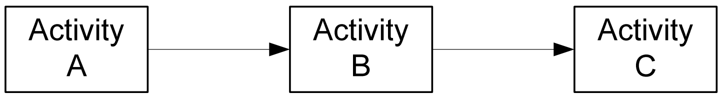


Figure 3 Simple workflow diagram

Another viewpoint is that activity *B* has the precondition that activity *A* has to be finished before activity B is allowed to start. From an access control point of view, access to activity *B* is only granted when activity *A* has finished. Activity B is apparently is dependent on the results of activity A.

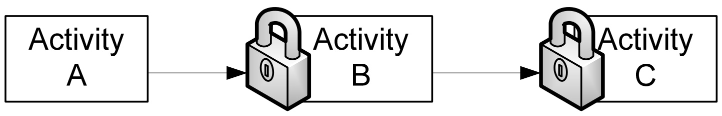


Figure 4 Access control applied to activities

Instead of defining constraints based on the activities themselves, they can also be defined based on resources. The output of activity *A* might put a certain resource in such a state that activity *B* is allowed to start. In order to do so, the resources mentioned in the constraints must be in a state as defined by these constraints. In this case, activity *A* happens to be the activity that puts those resources in such a state. If all the constraints are based on resources instead of activities, no specific order of activities will be implied. Any business activity can be executed at any time, as long as the business activity does not violate the constraints which apply to that business activity. The constraints are represented as business rules.

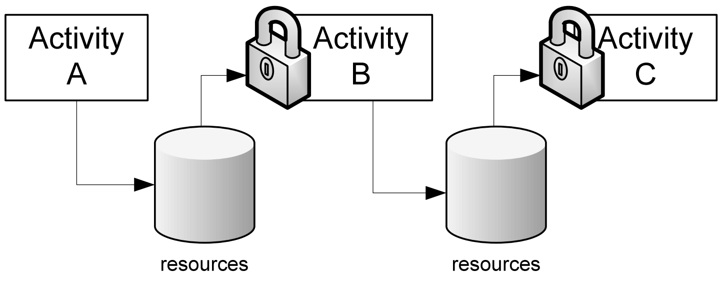


Figure 5 preconditions based on resources

In order to evaluate the business rules, information is needed about the terms in the business rules. The information that is needed might be concerned about more than one term (entity). Whereas CBAC provides claims about a single entity (the subject), CBW needs to be able to get claims about multiple terms (entities). The information that is needed to be able to make an evaluation can possibly come from different sources. The resources that need to be accessed, to form a decision on whether an activity is allowed to execute, can differ from the resources that are needed during the execution of the activity.

CBW combines a number of concepts from different areas, like business processes, business rules, access control, and claims. These concepts need to be combined to form a reference architecture for Claims Based Working. In the next section our research questions will be presented.

1.2 RESEARCH QUESTIONS

1.2.1 GOAL

The goal of this research thesis is to develop a reference architecture for Claims-Based Working which integrates business rules with CBAC and to validate it with a demonstrator.

1.2.2 MAIN RESEARCH QUESTION

What does a reference architecture for Claims-Based Working look like?

1.2.3 SUB QUESTIONS

To answer the main research question, a number of sub-questions have to be answered.

- *How can business processes be specified in such a way that access control principles can be applied to business activities?*
  Triggering business activities can be seen as access control, but research is needed to find out how this idea is supported in the literature and if it is technically possible. This sub research question has two parts to it. On the one side are the access control principles and on the other side are the business processes.
  - *What does an access control meta-model look like?*
    To define a generic access control model a literature study has to be done. The access control meta-model has to be applicable to business processes.
  - *How can the previously defined AC meta-model be incorporated in declarative business processes?*
    Access control to business processes is useful if the business processes are modeled in a declarative fashion. There are several declarative methods to model business processes. The method best suited for CBW has to be selected.
- *How can the concepts of CBAC be integrated with BAAC to form a reference architecture for Claims-Based Working?*
  Information is needed to be able to evaluate the access control constraints of BAAC. CBAC seems to be a suitable candidate for this. It is a mechanism which allows information to be gathered from resources inside and outside the domain of an organization.
- *How to implement / validate CBW?*
  The reference framework for CBW will be put into practice using a demonstrator. This is the final part of the goal set in this research. The value of a demonstrator is that it is a tool to start discussions about the way business processes are organized. The results of these discussions can be used to do additional research in the area of Claims Based Working.

1.3 PROJECT SCOPE

This research assignment is carried out at TNO ICT. The mission statement of TNO is to apply scientific knowledge with the aim of strengthening the innovative power of industry and government (TNO 2010). TNO notices that organizations often have trouble operationalizing their business policies. They recognized the gap and have proposed the idea of Claims Based Working, which is based on a number of theories regarding business process modeling.

A demonstrator is valuable for TNO and its partners, because it shows how CBW works in practice. It is valuable for this research, because it allows the ideas and concepts surrounding CBW to be evaluated for their practical use. The demonstrator will make the concept of CBW tangible by using a test scenario.

1.4 RESEARCH APPROACH

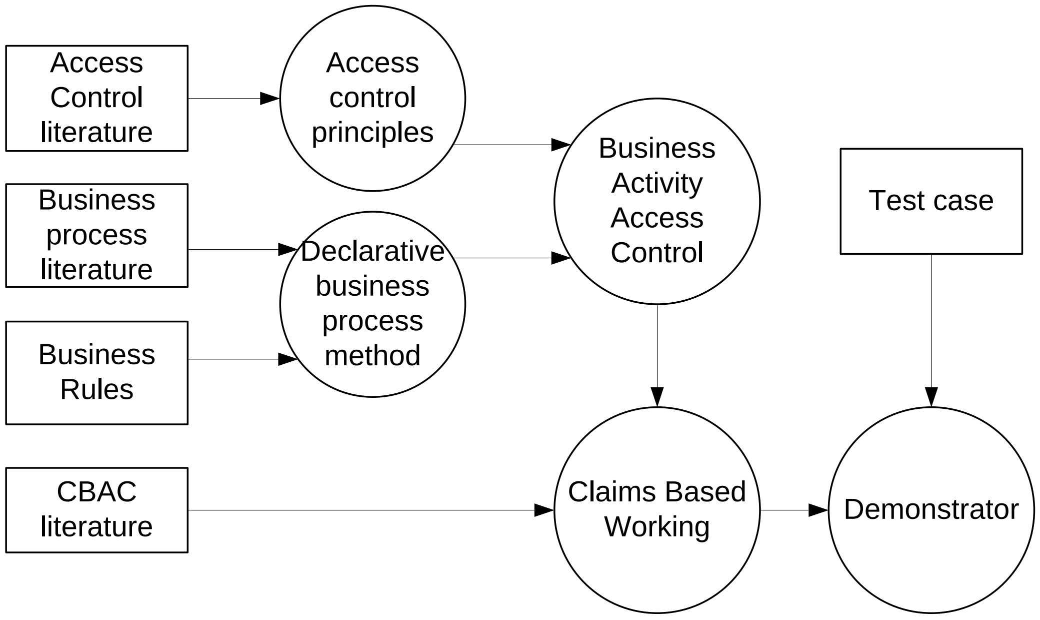

Access Control literature
Access control principles
Business process literature
Declarative business process method
Business Rules
Business Activity Access Control
Test case
CBAC literature
Claims Based Working
Demonstrator

Figure 6 Research approach

When business processes are modeled using a declarative approach, the permission to execute a business activity can be seen as a kind of access control. A literature study will be done to find out what generic access control meta-model exists. The literature describes a number of methods to model business processes. We will use a method to model business processes using declarative business rules that is suited for CBW.

Declarative business processes and access control principles will be integrated into a meta-model which we will call business activity access control. This model will be combined with the concepts of Claims Based Access Control to form a reference-architecture for Claims Based working.

During the research the demonstrator will be made using a number of modules. These modules will be combined to form the final CBW demonstrator. Two scenarios will be used to validate Claims Based Working.

1.5 STRUCTURE OF REPORT

This research thesis is organized as follows. In chapter 2 the related work to this research topic will be discussed. The literature study for this research will contain topics about access control (§2.1), business processes (§2.2), and business rules (§2.3). Chapter 3 bundles the results of the literature study to form the concepts of CBW, like the use of declarative business processes, access control in business activities (BAAC), and claims. A reference framework will be developed which incorporates these concepts. The demonstrator is documented in chapter 4. In chapter 5 we will validate CBW using two scenarios. The conclusions, contributions, limitations, and future research are presented in chapter 6.

2 RELATED WORK

In this chapter we are going to introduce a number of relevant topics which are going to be used to describe the reference framework for Claims Based Working. First the topic of access control will be introduced. After which an overview will be given about business processes. Finally we will introduce the research which is done in the area of business rules. The relevance of the different topics will become clearer in the next chapter, which will describe the reference framework we have come up with.

2.1 ACCESS CONTROL

Access control is something that we encounter many times on a daily basis. For example a typical student like myself uses public transportation in order to get to class. When we students take the bus we must show our public transportation card (OV-kaart) to the bus driver or more recently swipe the card over a check-in device. At the university we have to type in our user name and password to use the university's computer system. Each time some sort of access control is enforced. There are many access control mechanism. In this section we will discuss a number of these mechanisms.

The purpose of access control is to limit the actions or operations that a legitimate user of a computer system can perform (Sandhu and Samarati 1994). This definition talks about limiting the actions of users and the users of a computer system also needs to be legitimate.

Most users of computer systems are not allowed to perform every action that is possible in the computer system. The actions a user can perform on the system need to be limited. Users are only allowed to read and manipulate a small portion of the system for which they are authorized. Limitations are put on accounts which in turn are linked to users. When a user performs an action in the systems the action is performed through the account of that user.

Before a user can use an account the user's legitimacy has to be proven through authentication. A user is legitimate if with certain degree of certainty it can be proven that the user is the same user as is represented by the account. The picture below shows the relation of access control with other security services.

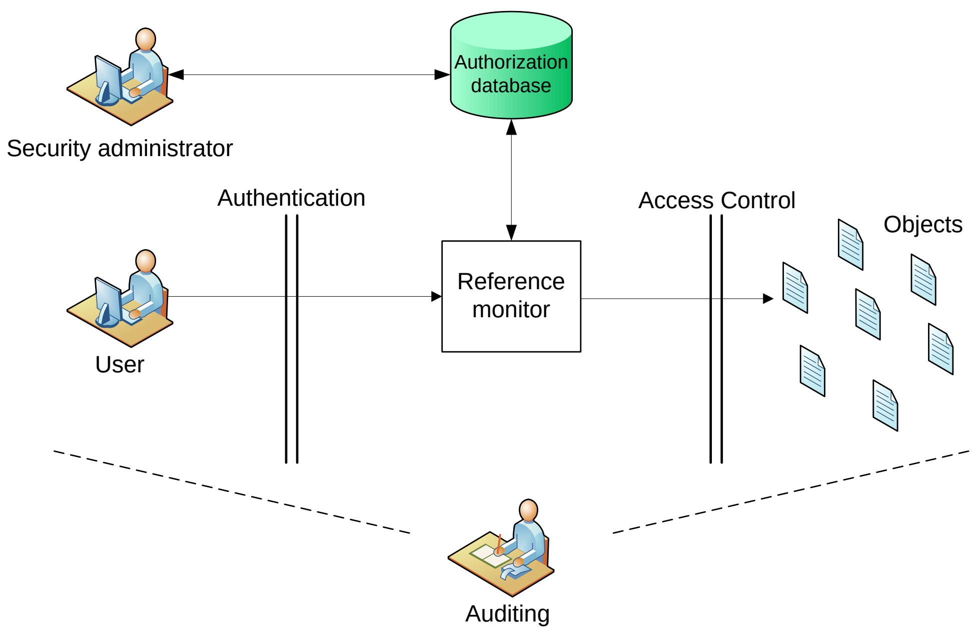


Figure 7 Overview of security services (adapted from Sandhu 1994)

Access control can be enforced using a number of different access control mechanisms. The access control mechanisms can either use white lists and/or black lists approach to provide access control.

The white list approach mandates who is allowed to do something to some object. For example a white list can state that the user account of John has read privileges for document X. This means that John is allowed to read document X. If John only has read privileges for document X then he is not allowed to do anything else to it, like modifying and deleting document X.

Black lists on the other hand state what is not allowed. For example a black list can state that the user account of John does not have read privileges for document X. This means that John can do anything to document X except reading it. When to use white lists or black lists or a combination of the two depends on the kind of access control that needs to be enforced. An access control mechanism can implement either the white lists or black lists or a combination of both.

2.1.1 ACCESS CONTROL MECHANISMS

There is a variety of different access control mechanism out there, of which the best known are Discretionary Access Control (DAC), Mandatory Access Control (MAC) and Role Based Access Control (RBAC). In recent years a number of advanced access control mechanisms have emerged. Of those the Claims Based Access Control mechanism is a relevant mechanism for this research as it allows for information gathering at the original source.

DISCRETIONARY ACCESS CONTROL

According to (NIST 1993) the definition of Discretionary Access Control is:

"A means of restricting access to objects based on the identity and need-to-know of the user, process and/or groups to which they belong. The access control is discretionary in the sense that a subject with certain access permissions is capable of passing that permission (perhaps indirectly) on to any other subjects."

If we rewrite this definition we get the following definition:

Discretionary Access Control is a means of restricting access to resources based on the identity and need-to-know of entities.

Using DAC an entity can access a resource based on the access right for that entity.

An entity has to send an access request to a reference monitor in order for the reference monitor to grant access to the entity. The reference monitor is the enforcer of access control. The reference monitor first has to know who wants access; otherwise access can only be granted to no one or everybody. Secondly the reference monitor needs to know for which resource access is requested, otherwise the reference monitor is unable to determine for what resource to provide access and access could only be granted for all resources at once. Thirdly the reference monitor needs to know what kind of access is required. Access to a resource would not be very useful if no actions were possible. For example if a user request access to a document, that user does not just wants access, but also wants to do something, like read the document or edit the document. An access control request therefore consists of the triple <e, o, p>. The *e* being the entity that requests the access, *o* being the object or resource the access is requested for and *p* being the privileges (what *e* wants to be able to do to *o*).

Discretionary Access Control can be represented by an access matrix as shown in the figure below.

	File 1	File 2	File 3	File 4	Account 1	Account 2
John	R W		R W		Inquiry Credit	
Alice	R	R W	W	R	Inquiry Debit	Inquiry Credit
Bob	R W	R	R W			Inquiry Debit

Figure 8 An access matrix (adapted from Sandhu 1994)

The columns of the matrix represent the objects. The rows of the matrix represent the entities and each cell contains the privileges which the specific entity has for the specific object. The meaning of privileges in each cell can be interpreted in two ways. An access matrix can either be a white list or a black list.

There are a number of ways to implement an access matrix. The best known are Access Control Lists (ACL), capability lists and authorization relations. Each of them has their positive and negative sides.

Access Control Lists represent the columns of the access matrix. Each object has an ACL and it contains the privileges each entity has. With ACL's it is easy to find the privileges of entities on a specific object. It is more difficult to find the privileges that a specific user has on each object. To find all the privileges of a specific user, the ACLs of every object has to be queried for whether or not the entity is on the list and if so which privileges the entity has.

The rows of an access matrix can be represented as a Capability List. Every entity has a list and a Capability List contains all the objects for which the entity has privileges. A downside to this method is that it is harder to find the privileges of all users of a specific object.

MANDATORY ACCESS CONTROL

Mandatory security policies enforce access control on the basis of regulations mandated by a central authority (Samarati and de Capitani 2001). The most common form of Mandatory Access Control (MAC) is the multilevel security policy, based on the classification of the subjects and the objects. The classification consists of two components, namely the security level and the category. An example of a set of security levels is: *Top Secret* (TS) > *Secret* (S) > *Confidential* (C) > *Unclassified* (U). The set of categories is an unordered set (e.g. *Army*, *Nuclear*, *Project X*). Each object in the system is assigned a security level and zero or more categories and each subject is assigned a set of categories and the security level for those categories.

Access is granted to an object if the security classification of the subject is the same or higher than the security classification of the object. For example if an object is assigned a security level of *Secret* and belongs to the categories *Army* and *Project X*, then the subject requesting the object has to have a security classification of *Secret* or higher for the categories *Army* and *Project X*.

Mandatory access control policies can be implemented in two ways, which are secrecy based and integrity based. Secrecy based mandatory policies were first formulated by Bell and LaPadula (Bell and LaPadula 1973). They provide confidentiality of the objects being protected. There are two principles that have to be satisfied to ensure object confidentiality. The first principle is that it is not allowed to read from objects which have a higher security level than the subject that is reading it. The second principle is that a subject is only allowed to write to objects which have the same or a higher security level. Confidentially of the objects can only be guaranteed if these two principles are satisfied. Secrecy based mandatory policies do not ensure integrity, for it is still possible for subjects to write to higher security level objects. A subject might be able to implant a Trojan horse this way.

Using the principles of Bell and LaPadula, Biba (Biba 1977) proposed a policy which ensures data integrity. As with the secrecy based policy, subjects and object get classifications, for example: *Critical* (C), *Important* (I), and *Regular* (U). The read and write privileges are different than those from the secrecy based policy. Subjects are not allowed to write to a higher classification level and are not allowed to read from lower classification levels. This prevents information stored in a lower level object from flowing into higher level objects. Unauthorized modification of data is prevented by not allowing lower level subject to write to higher level objects.

ROLE BASED ACCESS CONTROL

Role Based Access Control is a form of access control where permissions and / or denials are not directly assigned to the individual entities, but to roles. Each entity is then assigned to one or more of these roles. Instead of assigning permissions to each and every entity in the system, system administrators can assign roles to the entities. It is easier to assign entities to roles than to set the individual permissions for the entities (NIST 1993). The NIST study also indicates that permissions assigned to roles change relatively slowly. The roles as defined in RBAC can often be directly mapped to roles inside an organization. Users can be assigned permissions based on the roles they fulfill in the organization.

In the paper by (Sandhu and Coyne 1996) four conceptual models are proposed to explore the various dimensions of RBAC. RBAC0 is the base model and set the minimum requirements for RBAC. RBAC1 and RBAC

2 extend the base model with role hierarchies (RBAC1) and constraints (RBAC2). RBAC3 consolidates RBAC1 and RBAC2.

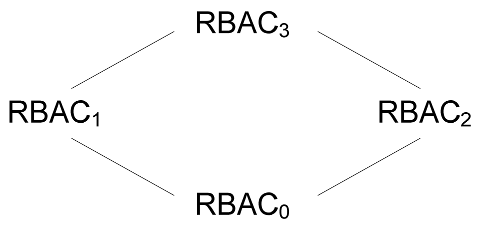


Figure 9 Family of Role Based Access Control models

RBAC$_0$ – BASE MODEL

The base model of RBAC consists of four concepts: users (*U*), roles (*R*), permissions (*P*), and sessions (*S*). *Users* are the entities for which access to objects must be allowed or denied. An example of such an entity is an employee of a company. A *Role* is a job function within the organization that describes the authority and responsibility conferred on a user assigned to the role. A *Permission* is an approval of a particular mode of access to one or more objects in the system. In the access control literature permissions are also referred to as privileges. Privileges and permissions differ in that privileges can be positive as well as negative, whereas permissions refer to positive access modes (Sandhu and Coyne 1996). Negative access modes are seen as constraints rather than negative permissions. Negative access modes are therefore modeled in the RBAC$_2$ model. The last concept of the base model is the concept of *Sessions*. A session maps one user to a subset of roles that the user has. When a user request access, a session is used to check the permission the user has. A user can use a number of sessions to access the system with different permissions.

The relationship between *Roles* and *Permissions* is a many-to-many relation. Each role can have many permissions and a permission can be assigned to multiple roles. A *User* can have multiple *Roles* and a *Role* can be assigned to many *Users*. Each *Session* belongs to exactly one *User* and each session contains one or more *Roles*. These roles need to be roles which are assigned to the user of the session.

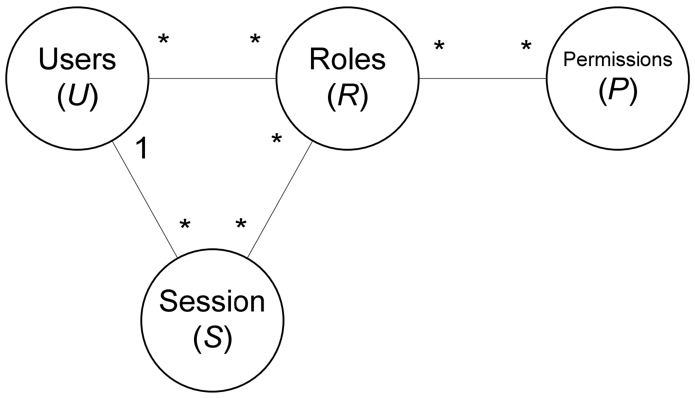


Figure 10 RBAC$_0$

RBAC$_1$ – ROLE HIERACHIES

The RBAC$_1$ model introduces hierarchy to roles. By introducing hierarchy in the roles the natural role structure of a company can be modeled. A *Role* belong to zero or more higher level *Roles* and can also contain zero or more lower level *Roles*.

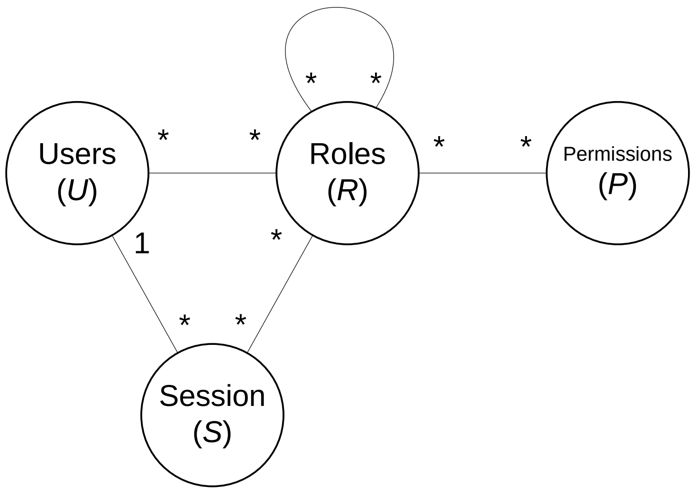


Figure 11 RBAC$_1$

For example the *Role* of project supervisor can contain the lower level roles of programmer and designer.

RBAC$_2$ – CONSTRAINTS MODEL

The RBAC$_2$ model for the most part is the same as the RBAC$_0$ model. The only difference is that *Constaints* are introduced. Constraints can have impact on *Users*, *Roles*, *Sessions*, *Permissions* or any combination of users, roles, permissions and sessions. Using *Constraints* a number of restrictions can be put on the RBAC model. There are too many possible constraints to name them all, but we will give a few examples. It possible to have constraints that ensures mutually exclusive roles. This constraint ensures that *Roles* that need to be carried out by different users cannot be assigned to the same user. Constraints can also be put on the number of users that can fulfill a certain role (e.g. there can only be one user that has the CEO role).

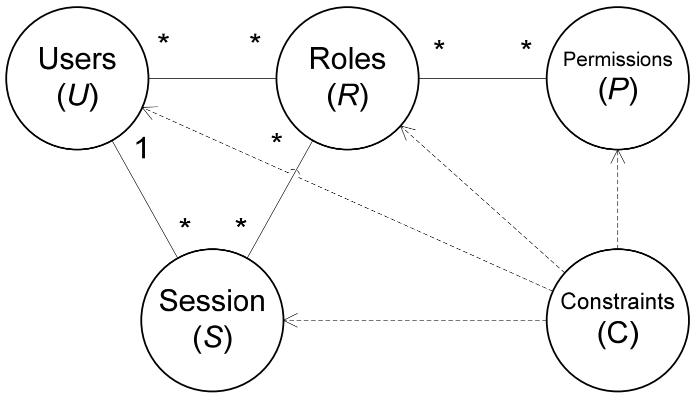


Figure 12 RBAC$_2$

RBAC$_3$ – CONSOLIDATED MODEL

RBAC$_3$ is the consolidated model of RBAC$_1$ and RBAC$_2$.

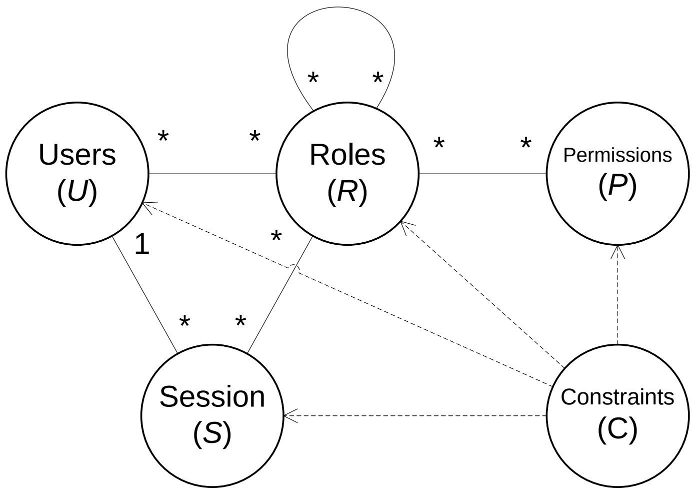


Figure 13 RBAC$_3$

The main advantage of using the RBAC systems is that users are assigned to roles, which makes it easier to manage the permissions of the different users of the system. A possible disadvantage of the RBAC system occurs when specific users need a number of custom permission. In this case a new role needs to be made to suit the requirements of that user. When having a lot of these exceptions the number of custom roles can be very high and the individual differences between the roles can become very small, which might cause for some confusion with the system administrators. The next access control mechanisms allow for a higher rate of exceptions and customization.

CLAIMS BASED ACCESS CONTROL

Claims Based Access Control is a mean to provide access control using claims. As mentioned earlier, we define a claim as:

An assertion of the truth of something, typically one which is disputed or in doubt, about an entity made by another entity.

A claim can be anything that describes an aspect of a subject. A few examples of a claim are: “Susie is older than 18”, “Susie has a driving license”, “John is in the group ‘system administrators’ within the Fabrikam domain” These examples are all assertions about an entity made by another entity. Whether or not these assertions can be used to grant or deny access depends on the trustworthiness of the entity that made the assertions.

The definition of trust according to (Oasis 2007) is:

Trust is the characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make a set of assertions about a set of subjects and/or scopes.

A trust relation must exist in order for entity A to rely on Entity B to make assertions about some other entity. In other words if entity *A* trusts entity *B* if *A* considers the claims issued by *B* as truthful. By having this trust entity *A* does not have to verify claims coming from entity *B*. Entity *A* still has to make sure that the claims issued by *B* are not forged by another entity. The issuer of claims can be verified using security tokens, which are constructs signed by the issuer, containing claims and possibly credential information.

Microsoft has made a model which provides access control using claims. The model is called the Identity Metasystem and can be defined as (Bertocci 2008):

The Identity Metasystem is a model that provides a technology agnostic abstraction layer for obtaining claims

The architecture of the Identity Metasystem is based on the Laws of Identity (Cameron 2005). Cameron has come up with a total of seven laws, which are summed up below:

1. User Control and Consent
Technical identity systems must only reveal information identifying a user with the user’s consent.

2. Minimal Disclosure for a Constrained Use
The solution which discloses the least amount of identifying information and best limits its use is the most stable long term solution.

3. Justifiable Parties
Digital identity systems must be designed so the disclosure of identifying information is limited to parties having a necessary and justifiable place in a given identity relationship.

4. Directed Identity
A universal identity system must support both “omni-directional” identifiers for use by public entities and “unidirectional” identifiers for use by private entities, thus facilitating discovery while preventing unnecessary release of correlation handles.

5. Pluralism of Operators and Technologies
A universal identity system must channel and enable the inter-working of multiple identity technologies run by multiple identity providers.

6. Human Integration
The universal Identity Metasystem must define the human user to be a component of the distributed system integrated through unambiguous human-machine communication mechanisms offering protection against identity attacks.

7. Consistent Experience Across Contexts
The unifying identity metasystem must guarantee its users a simple, consistent experience while enabling separation of contexts through multiple operators and technologies.

Cameron states that identity related systems must obey these laws in order for them to be accepted by the users of the system. The identity Metasystem is built to obey these laws.

The Identity Metasystem has three roles defined, which are the subject, relying party and the identity provider (Microsoft 2005).

- **Identity Providers**, are entities that issue a digital identity. Everybody can potentially be an identity provider, but not every identity provider will be accepted for this interaction. There are low level identity providers and high level identity providers (e.g. government).
- **Relying Parties**, are entities that request an identity token that contains specified claims. They are called relying party because they rely on a third party to validate the specified claims as accurate.
- **Subjects**, which are the individuals and other entities about whom claims are made.

Using the concepts mentioned above a number of patterns can be created to support claims based access control. Two of the possible patterns will be described. The first will be the canonical pattern, which is the minimum pattern to enable CBAC. The second pattern will be the claims transformer pattern. This last pattern is used to translate claims across different domains.

THE CANONICAL PATTERN: SUBJECT-IP-RP

The canonical pattern is the minimum pattern necessary to enable CBAC. There are three parties involved; a subject, a relying party, and an identity provider.

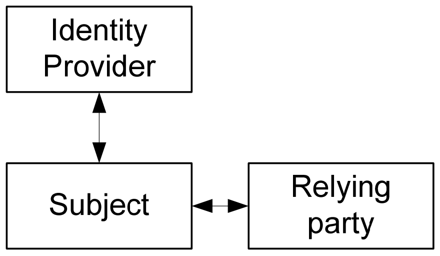


Figure 14 Subject-IP-RP pattern

The *Relying Party* is the party that has something of interest to the *Subject*. When a *Subject* wants to gain access to resources of a *Relying Party*, the *Relying Party* sends a policy to the subject containing a list of required claims and a list of *Identity Providers* that are trusted by the *Relying Party*. The *Subject* determines whether it can comply with the policy, meaning whether of not the required claims can be gathered from one of the *Identity Provider* listed in the policy. The *Subject* sends a request for claims to one of the *Identity*

Providers. After the selected *Identity Provider* has authenticated the *Subject*, the requested claims are sent back to the subject in the form of a token which is signed by the *Identity Provider*. The *Subject* uses this token to invoke the *Relying Party*. The *Relying Party* examines the token to check whether the token is generated by an *Identity Provider* that is trusted and if the required claims are in the token. If this is the case, the claims are passed through to some access control logic and if the access control logic is satisfied the claims can also be passed to the resource itself.

The *Relying Party* trusts the *Identity Provider* to correctly authenticate the *Subject* and to deliver correct claims. The *Relying Party* can make decision about the truthfulness of the *Claims* based on the trust relation between the *Relying Party* and the *Identity Provider*. If the *Relying Party* does not trust the claims coming from an *Identity Provider*, it would not be possible to make an access control decision based on these claims. If the *Relying Party* trusts the *Identity Provider*, then the *Relying Party* can assume the claims as truthful and make access control decisions based on them.

THE CLAIMS TRANSFORMER PATTERN: SUBJECT-IP-CLAIMS TRANSFORMER-RP

The claims transformer pattern is an interesting pattern, because the claims which the relying party receives do not directly come from a party which originally generated the claims. In this pattern the *Claims Transformer* is used to translate or transform the claims coming from the *Identity Provider* to claims which can be used by the *Relying Party*. The *Claims Transformer*, just like the *Identity Provider*, is a *Security Token Provider* (STS).

There can be a number of reasons to use a *Claims Transformer.* Indirect trust is one of the reasons to apply this pattern as in some situations certain parties are not trusted directly. For example when boarding an airplane the nice lady at the gate does not let you board the plane when you present her with your passport in itself. A boarding pass is required which can be obtained from the check-in desk with your passport or identification card.

Claims can also be in the wrong format. This can be for the result of language. For example if a claim is in Chinese an English speaking *Relying Party* probably will not be able to understand the claim.

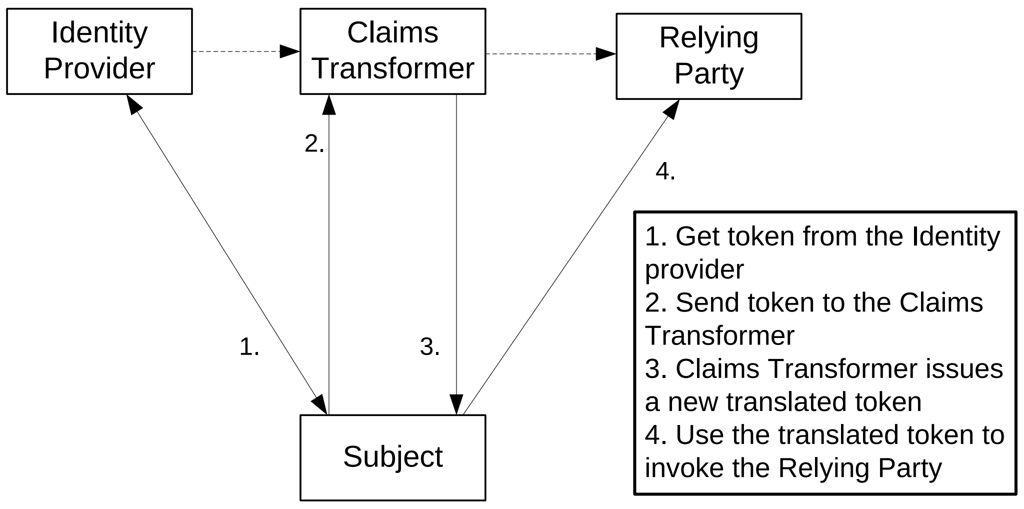

Identity Provider
Claims Transformer
Relying Party
2.
4.
1.
3.
Subject
1. Get token from the Identity provider
2. Send token to the Claims Transformer
3. Claims Transformer issues a new translated token
4. Use the translated token to invoke the Relying Party

Figure 15 Claims transformer pattern

2.1.2 CONCLUSIONS

The each of the access control mechanisms discussed above seemingly has more functions than the access control mechanism discussed above it. With DAC it is possible to enforce access control on specific Subjects. MAC extends this by adding level to the Subjects and Objects. RBAC goes even further by adding roles to Subjects and Objects. But it is not possible to use multiple access control mechanism. If for example DAC is implemented it is not possible to use levels or roles. If RBAC is implemented it is not possible to directly authorize Subject or use levels. In our framework we want to use an access control meta-model which can combine multiple access control mechanisms.

The last part access control mechanism discussed in this section is Claims Based Access Control. This mechanism is different from the other mechanisms as it focuses on gathering information which is necessary to make an access control decision. The other mechanism all assume that the information to make an access control decision is locally available.

Using *Claims* information can be gathered by the *Relying Party* from external sources. CBAC shifts the responsibility of gathering information to the *Subject*. When a *Subject* wants access to resources of a *Relying Party*, the *Relying Party* makes the *Subject* responsible for gathering the necessary information for the *Relying Party* to make access control decisions.

2.2 BUSINESS PROCESSES

2.2.1 DEFINITIONS

As stated before a business process consists of a number of activities which are carried out to create something of value to someone (Hammer and Champy 1993); (Davenport 1993). An activity is a set tasks or operations which can be carried out by one entity at one location. To create the end product of a business process, a number of business activities have to be carried out. The order in which the activities are carried out can be orchestrated in various ways. On one extreme the order of activities can be predetermined. The activities can only be carried out in the order as described. On the other extreme the order of activities can be free. The order of activities is determined by people who are carrying them out. These two extremes fall in line with theory X and theory Y (McGregor 1960). McGregor makes a number of assumption regarding humans (see table below).

Theory X	Theory Y
Humans inherently dislike working and will try to avoid it if they can.	People view work as being as natural as play and rest. Humans expend the same amount of physical and mental effort in their work as in their private lives.
Because people dislike work they have to be coerced or controlled by management and threatened so they work hard enough.	Provided people are motivated, they will be self-directing to the aims of the organization. Control and punishment are not the only mechanisms to make people work.
Average employees want to be directed.	Job satisfaction is key to engaging employees

	and ensuring their commitment.
People do not like responsibility	People learn to accept and seek responsibility. Average humans, under the proper conditions, will not only accept but even naturally seek responsibility.
Average humans are clear and unambiguous and need security at work.	People are imaginative and creative. Their ingenuity should be used to solve problems at work.

Table 1 Theory X & Y

Business processes which are arranged according to Theory X are suited for manufacturing plants where there is not much variation in the order in which the activities need to be carried out. Theory Y is better suited for knowledge workers, where the order of activities or the activities themselves are not clear. Business processes can be implemented in a number of way to be theory X oriented or theory Y oriented or a combination of both. Business processes which are implemented using traditional workflows are compliant with theory X. In workflows all the possible paths a process instance can run through are predefined. Workflows consist of chained business activities. An activity is a certain amount of work which is executed by one entity (e.g person or computer) at one place. The time it takes to execute an activity can vary depending on the complexity of the activity, the capabilities of the entity which carries out the activity and external factors. The entity that carries out the activity is responsible for the execution of the activity and the time it takes to finish the activity.

During its life time an activity can be in a number of states. The diagram below shows the four states an activity can be in (Pesic and Aalst 2006).

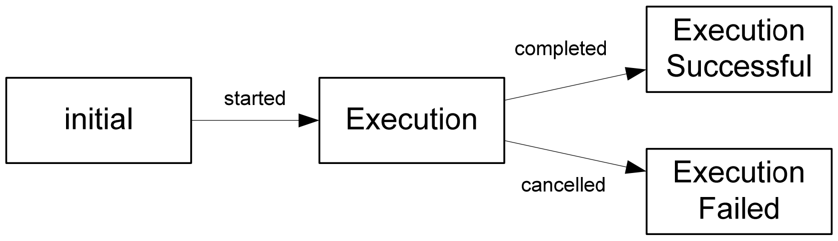


Figure 16 Activity states

At the beginning of the activity lifecycle an activity is in an initial state. An activity which is in the initial state can be executed. The time it takes to successfully complete an activity or cancel an activity can depend on a number of factors. It might even be possible that an activity will never go out of the running state. Usually this will indicate that something unforeseen has happened during the execution of the activity. An employee could have been disturbed during the execution of the activity and later forgot all about it.

2.2.2 TYPES OF IMPLEMENTATIONS

Implementations of business processes have to make trade-off between flexibility and support. On one hand there is a desire to control the process preventing incorrect execution and results. On the other hand there is a desire to make processes flexible to let the user make decisions of their own. (Aalst, Pesic et al. 2009). Imperative business modeling languages are controlling in nature by explicitly specifying the procedure. Declarative business modeling languages support flexibility by specifying what needs to be done and not explicitly how it should be done (Pesic and Aalst 2006).

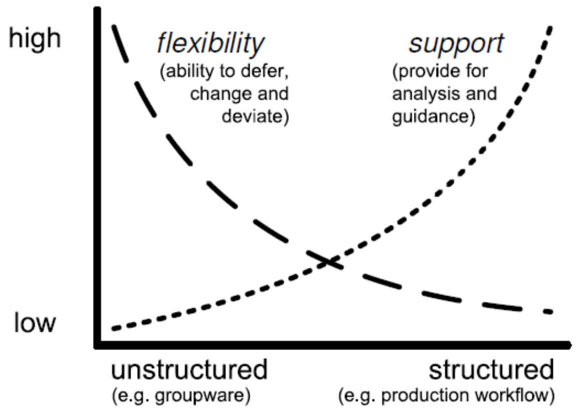


Figure 17 Tradeoff Flexibility and Support (Dongen 2005)

IMPERATIVE APPROACH

Traditional Workflow Management Systems (WFMS) are supportive in nature (right hand side of the figure above) and use procedural models to explicitly specify each step of the process. The activities in a business process have to be coordinated in a way, such that the activities are carried out at the right moment. It is not meaningful to perform the activity to process the missing item form, if the form has not been filled in completely.

Activities are linked together to form workflows. A typical workflow consists of an initiator, which sets the workflow in motion. The initiator for the purple crocodile scenario is the mother informing the desk clerk that her daughter has lost her purple crocodile.

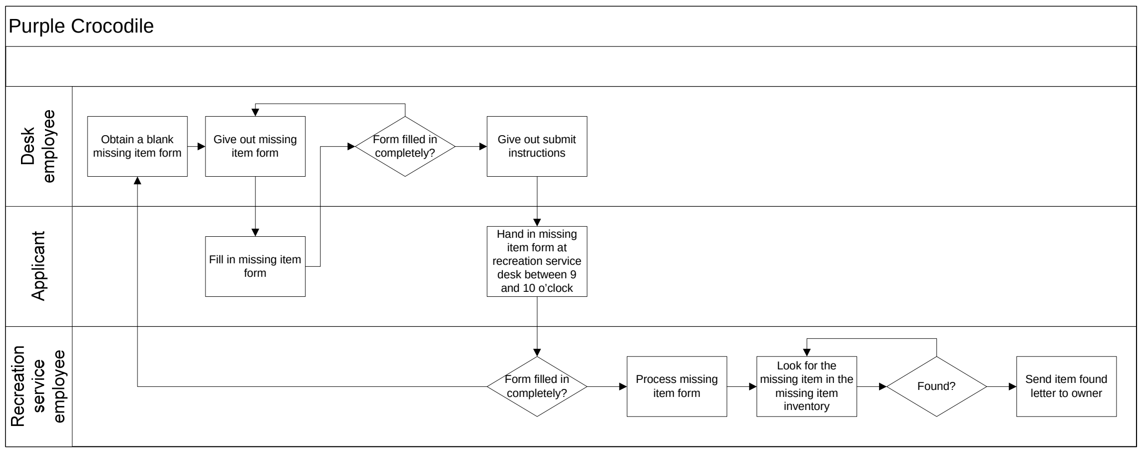

Purple Crocodile
Desk employee
Applicant
Recreation service employee
Obtain a blank missing item form
Give out missing item form
Form filled in completely?
Give out submit instructions
Fill in missing item form
Hand in missing item form at recreation service desk between 9 and 10 o'clock
Form filled in completely?
Process missing item form
Look for the missing item in the missing item inventory
Found?
Send item found letter to owner

Figure 18 Workflow

DECLARATIVE APPROACH

In contrast with the imperative approach, the declarative approach uses constraints to organize business processes.

Pesic (2008) defines three types of scenarios exists in business processes:

- Scenarios which are forbidden and thus may never occur.
- Scenarios which are allowed but should be avoided if possible.
- Scenarios which are allowed.

Of the entire set of possible scenarios traditional workflows only implement a small subset. Scenarios that fall outside of the implemented paths are not supported. The proposed declarative approach makes it possible to execute much more scenarios. All scenarios except for the forbidden scenario are allowed. This is an important difference with traditional workflows as they describe what is possible and the declarative approach describes what is forbidden. This can be compared with white lists and black lists, where imperative business processes follow the white list approach. Traditional workflows explicitly describe the possible scenarios. Declarative business processes on the other hand follows the black list approach as they only describe the scenarios which are not allowed, every other scenario which is not described as forbidden is allowed.

The order of activities in a declarative process is determined by constraints. Mandatory constraints concern the forbidden scenarios and optional constraints specify the optional scenarios. The total set of allowed scenarios is all the scenarios that do not violate the mandatory constraints. If no constraints are specified the business process is very flexible. The decision what to do is left entirely to the user. The more constraints are specified the more the business process shifts to the supportive side.

The constraints can be modeled using a number of predefined templates. The figure below shows a number of them. The *responded existence* relation means that when *A* is executed *B* also has to be executed, but not vice versa. The *co-existence* relation on the other hand means that if *A* is executed *B* also has to be executed at some time and if *B* is executed *A* also has to be executed. These two relations do not specify an order of execution. The relations *response*, *precedence*, and *succession* do specify an order of activity.

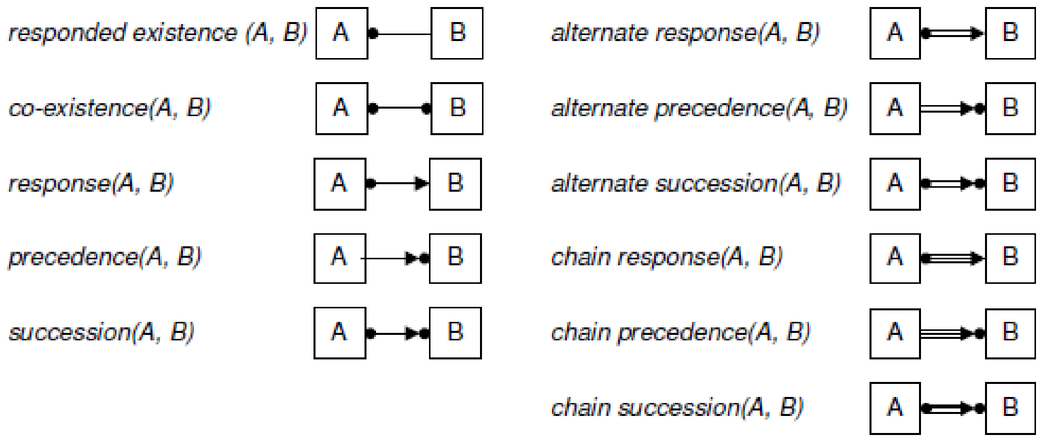


Figure 19 Notations for relational templates (Pesic 2008)

PERSPECTIVES ON BUSINESS PROCESSES

(Pesic 2008) uses three perspectives of process models to determine the order in which activities will be executed, which users can execute which activities, and which information will be available during execution. The three perspectives are: the control-flow perspective, the resource perspective, and the data perspective.

The control-flow perspective determines in which order the activities in a process can be executed. The control-flow perspective is probably the best known. This perspective defines the order in which activities can be carried out. For example activity A is carried out first, then activity B and after it has finished activity C can be carried out.

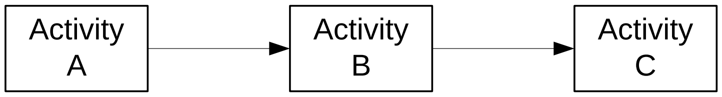


Figure 20 control-flow perspective

The resource perspective defines which entities are authorized to carry out each activity. It also shows how each entity is allocated to the activities. In the figure below there are a total of three human entities. There is one human entity with role X and two entities with role Y. Entities with role X are authorized to execute activity A and Activity B. Entities with role Y are authorized to execute activity B and C. A entity with a certain role can only execute activities that are assigned to that specific role.

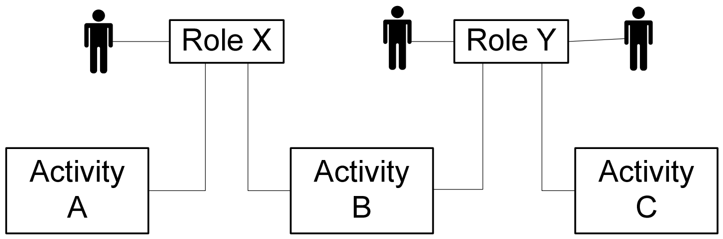


Figure 21 Resource perspective

Another perspective is the data perspective. This perspective defines which data elements are available in the process and how users can access them while executing activities. In the figure below there are three data elements X, Y, and Z. These three data elements are read and edited by the activities. Data element X and Z are edited by activity A. Activity B is allowed to read data element X and Z but not edit them. Activity B in turn edits data element X. Data element Z is not only read in activity B but also in activity C. The data elements can be seen as the input and output of the different activities. The input of the activities are the data elements which are read during the execution of that activity and the data elements which are changed by an activity during execution can be seen as output data.

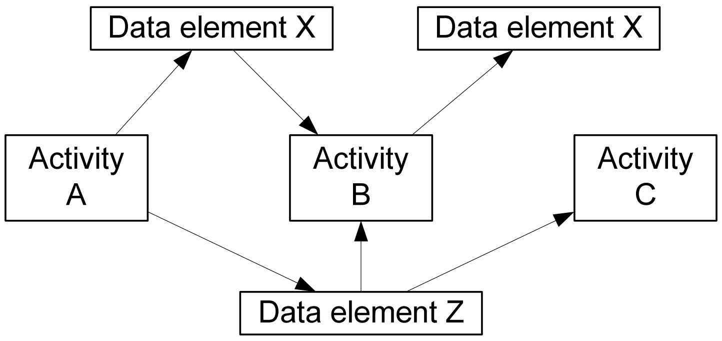


Figure 22 Data perspective

An interesting thing when looking at the data perspective is that there are methods which use the output of a business process as a starting point to determine which activities are required to facilitate in the creation of the process output.

There are methods which use the output of a business process to determine which activities are actually required to creation the process output. In a number of papers (Aalst 1999; Reijers 2003) the product/data structure is proposed as the starting point to organize the activities in business processes. In production oriented processes it is often clear what the end product will be.

For example in a car manufacturing process the actual car is the end product of the process. A car is a product which is not produced as whole at once. There are many sub products which have to be combined to become a car. A car consists of an engine, a chassis, wheels, bodywork, etc. All these parts can be represented in the bill-of-materials (BOM) of a car.

Products are also created by administrative processes encountered in banking, insurance and government. Examples of products of administrative processes are: tax declarations, traffic violations, insurance claims, and purchasing orders. For these administrative products it is also possible to construct bill-of-materials. The BOM of an insurance claims can look as follows. The insurance policy must contain customer data and insurance data. Medical data can be added optionally. The black dots indicate that the sub product is mandatory. Customer data consists of historical data and personal data. The insurance data consists of risk data and either standard rates or custom rates are applied. This choice is represented by the circle with more than one arrow going into it.

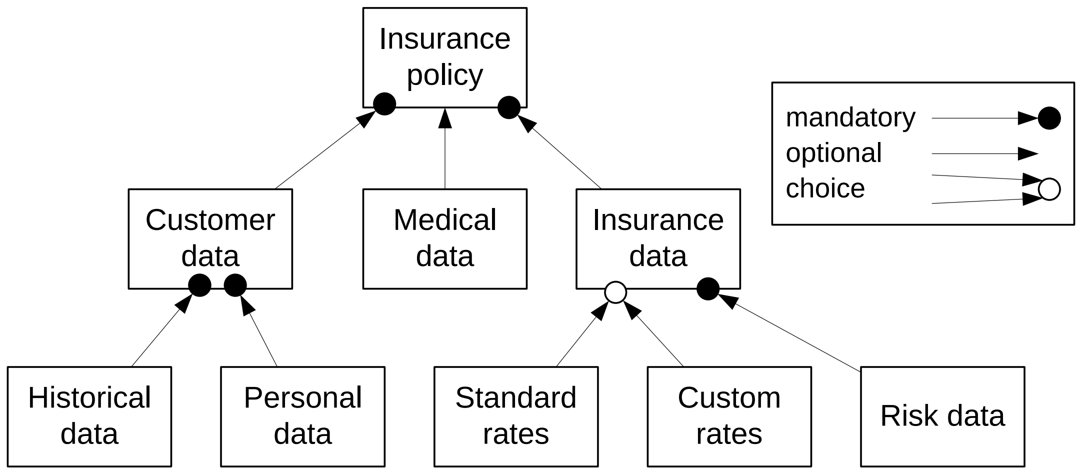


Figure 23 BOM of an insurance policy (Aalst 1999)

The method of creating workflows as described by van der Aalst (Aalst 1999) assumes that the process requirements are product-oriented. In many cases both the product-centered view and the process-centered view are useful. The distribution of entities over the activities is part of the process-centered view and is not addressed in the BOM approach.

2.2.3 CONCLUSIONS

The declarative business process approach in combination with the product/data structure is very interesting. With the product/data structure the outputs and inputs of the business activities can be defined, as each sub-product is created by at least one activity. Defining date requirements for activities allows for a declarative approach in setting up business processes. Data requirements can be defined using business rules, which will be discussed next.

2.3 BUSINESS RULES

Ronald Ross claims to be the founding father of business rules. In the mid 1980's Ross came up with the idea of business rules during a football match. During the match he was thinking about two questions which were (Ross 2010):

"What fundamental elements are needed to play the game of football, and which of those elements are more stable?"

The figure below shows the results of the analysis Ross made at the time.

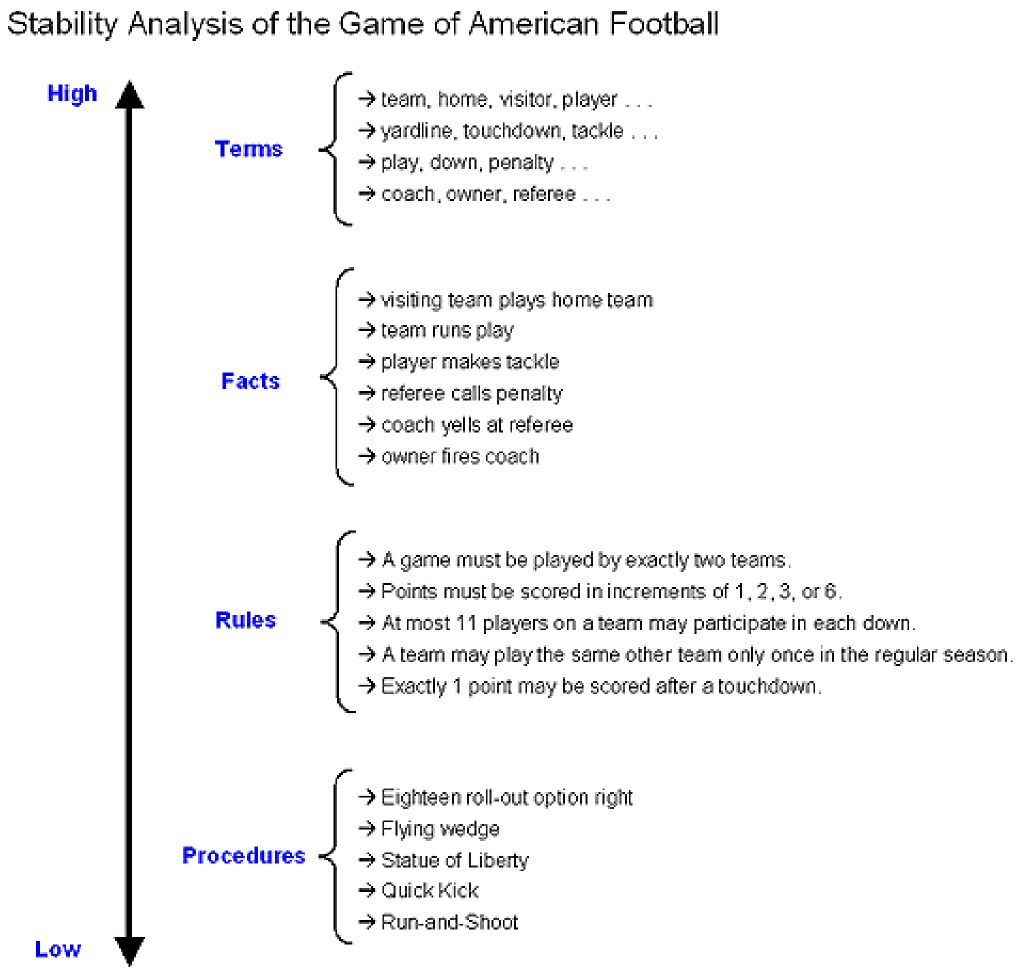

Stability Analysis of the Game of American Football
High
Low
Terms
→ team, home, visitor, player . . .
→ yardline, touchdown, tackle . . .
→ play, down, penalty . . .
→ coach, owner, referee . . .
Facts
→ visiting team plays home team
→ team runs play
→ player makes tackle
→ referee calls penalty
→ coach yells at referee
→ owner fires coach
Rules
→ A game must be played by exactly two teams.
→ Points must be scored in increments of 1, 2, 3, or 6.
→ At most 11 players on a team may participate in each down.
→ A team may play the same other team only once in the regular season.
→ Exactly 1 point may be scored after a touchdown.
Procedures
→ Eighteen roll-out option right
→ Flying wedge
→ Statue of Liberty
→ Quick Kick
→ Run-and-Shoot

Figure 24 Analysis of a football game

The fundamental concepts are represented by terms. For example players are needed to play a football match. A bunch of players is not enough; they need to be grouped into team. One team is the visiting team; the other is the home team. The teams need somewhere to play on. A football field has a certain layout, with yard lines and touchdown areas and certain other elements.

The fundamental concepts have relations with one another. These relations are called facts. For example just having a home team and visiting team is not enough to play football. They have to play against each other and players make touchdowns.

Having all the terms and facts are still not enough to play a football match. There need to be rules which provide guidance for the facts. For example there is a rule which states that there are at most 11 player of a single team on the field. This rule sets a restriction on the fact that a playing team consists of players.

Having the three concepts (terms, facts, and rules) is basically enough to play a football match, but without a strategy and procedures a football match would just be a bunch of people running around with a leather ball. The fourth concept that is needed is procedures. These are the plays a coach can order his team to run in to score touchdowns. During a match these procedures change in order to adapt to the current situation on the field.

The concepts of Ross do not just apply to football, but also apply in the business environment. Rules are needed to make sure that the business processes and procedures behave as intended.

2.3.1 STRUCTURES OF BUSINESS RULES

Looking at the literature Ross was not the only one who recognized business rules as being important for business processes.. Barbara von Halle and John Zachman among others are well known for their work regarding business rules. Von Halle proposed four categories of business rules (Halle 1997; Steinke and Nickolette 2003):

Definitions: These define entities and attributes
Facts: These are either relationships between entities or associations between an entity and its attributes.
Constraints: These are conditions about data that must always be true.
Derivations: These are usually applying logic to create a new piece of information.

The Business Rule Group categorizes business rules in a similar way. A business rule must be one of the following subtypes (BRG 2000):

Structural Assertion – a defined concept or a statement of a fact that expresses some aspect of the structure of an enterprise. This encompasses both terms and the facts assembled form these terms.
Action Assertion – a statement of a constraint or condition that limits or controls the actions of the enterprise
Derivation – a statement of knowledge that is derived from other knowledge in the business.

In the figures (25, 26, 27, 28, and 29) below the following icon mean a supertype-subtype connector

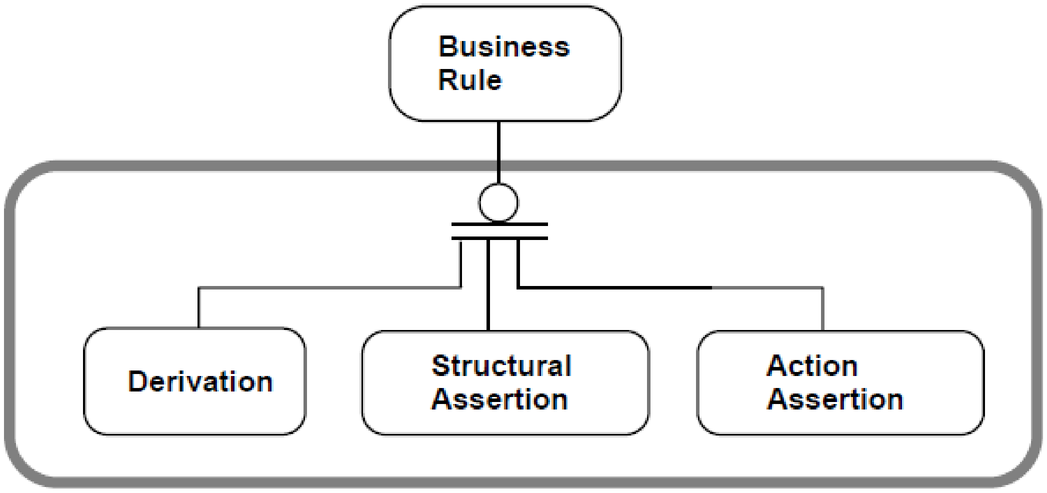


Figure 25 Business Rule Types (BRG 2000)

The *Derivation* subtype has two additional subtypes, which are *Mathematical Calculation* and *Inference*. These two subtypes are defined as:

Mathematical Calculation: A derivation that produces a *Derived Fact* according to a specified mathematical algorithm. *e.g. The derived fact Rental Amount is calculated from the Rental Rate multiplied by the Number Of Days.*

Inference: A derivation that produces a *Derived Fact* using logical induction or deduction. *e.g. The derived fact Rental Rate of a car is inferred from the Rental Rate of the car group that the car is part of.*

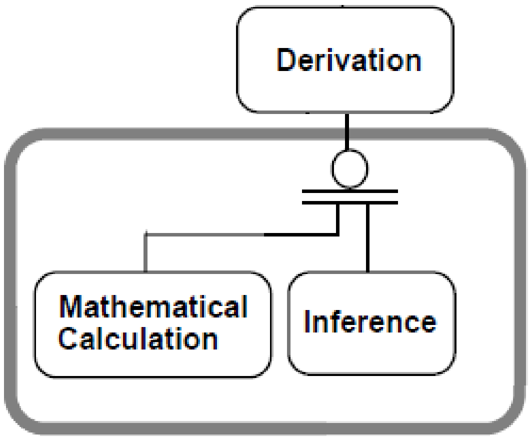


Figure 26 Derivations adapted from (BRG 2000)

The *Structural Assertion* subtype has two subtypes which are *Term* and *Fact*. These two subtypes are defined as:
Term: A word or phrase used by the business.
Fact: An associating of two or more *Terms*. It expresses a potential for association ('can be' or 'may be') rather than expressing a 'must be' association.

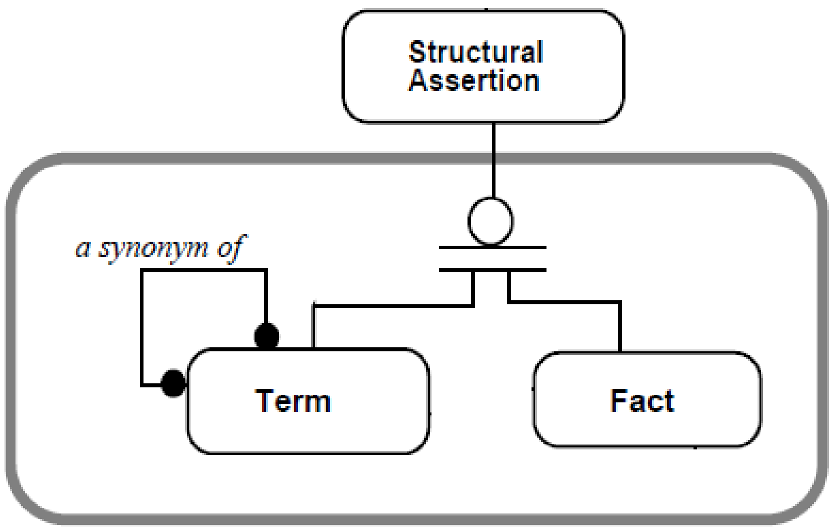


Figure 27 Terms and Facts adapted from (BRG 2000)

The *Action Assertion* subtype has to be classified as a *Condition*, an *Integrity Constraint*, or an *Authorization*. These three classes are defined as:

Condition: An assertion that if something is true, another business rule will apply. A *Condition* can be thought of as a test (if true). A *Condition* may be the basis for enforcing or testing other *Action Assertions*.
Integrity Constraint: An assertion that must always be true.
Authorization: An assertion that a specific privilege has been defined with respect to one or more *Constructs*. A *Construct* is a generalization that represents either a *Business Rule* or an *Action*.

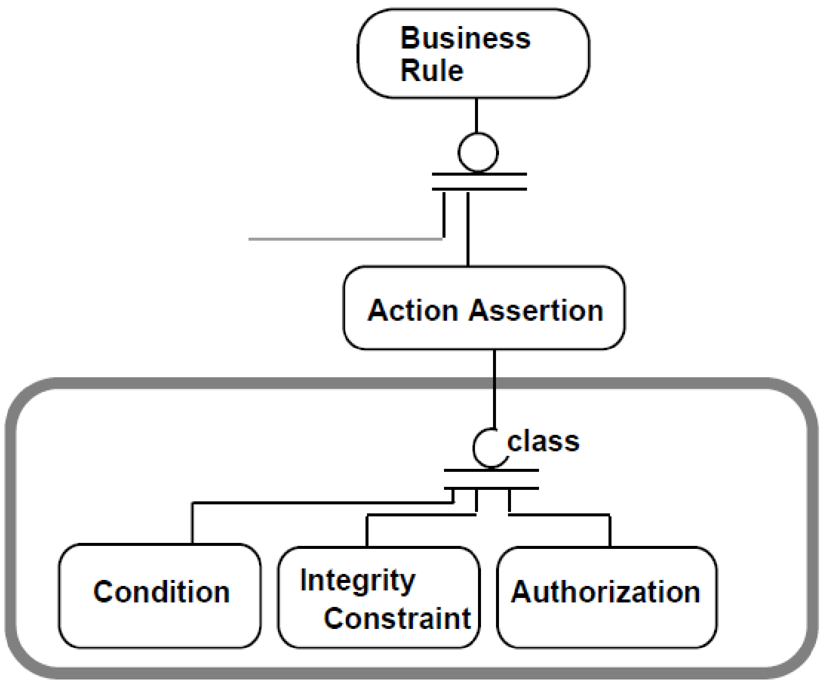


Figure 28 Action Assertions (BRG 2000)

An *Action Assertion* consists of an *anchor object* and a *correspondent object*. The *anchor object* and *correspondent object* of the following rule, “a car must have a registration number”, are “car” and “registration number” respectively. *Action Assertions* rules can be written in the form “must (not) be” or “must (not) happen”. In this form the *Action Assertion* is an *Action Controlling Assertion*. *Action Assertions* can also be rules which state what “should (not) be” or “should (not) happen”. Rules written in this form are *Action Influencing Assertions.*

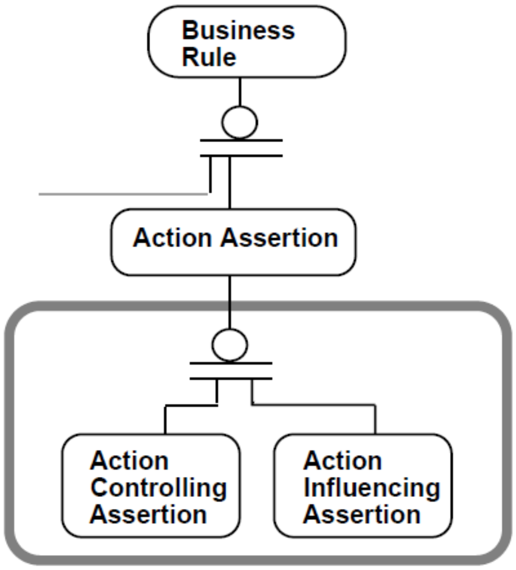


Figure 29 Action Controlling VS Action Influencing

2.3.2 CLASSIFICATIONS OF BUSINESS RULES

There are many kinds of business rules possible using the business rule model as described by the BRG. Many researchers have attempted to classify the possible rules. The results of the classifications made by the different researches have some differences between them. We will sum up a number of classifications made by different researchers.

Wagner (Wagner 2005) made the following classification of business rules:

Integrity rules – expresses constraints. *Each project must have one and only one project manager*

Derivation rules – expresses conditions that result in conclusions. *Platinum customers receive a 5% discount. John Doe is a platinum customer. As conclusion, John Doe receives a 5% discount*

Reaction rules (ECA) – specifies a trigger that activates the evaluation of the rule a condition that is evaluated, and a subsequent activity that will be carried out if the specified condition is met. *The evaluation of a reaction rule is triggered as soon as a new invoice is received. If the invoice amount is more than $1000 then a supervisor review is initiated*

Production rules (CA) – are similar to reaction rules, but do not specify a particular circumstance in which the evaluation takes place. *If there are no defects in the last 10 widgets, the entire batch is quality approved*

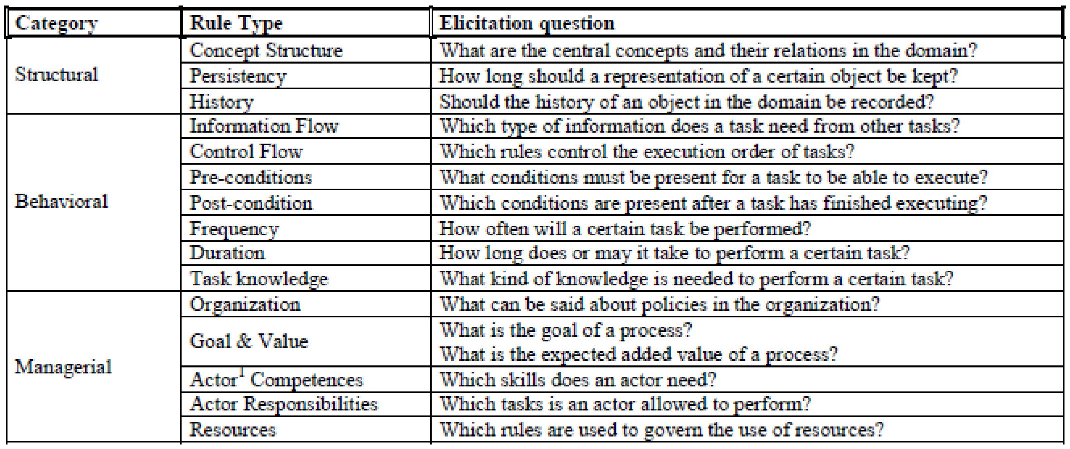
Transformation rules – restrict the state changes of objects. *An employee's age can change from 30 to 31, but not from 31 to 30.*

Another classification is that of (Iacob 2009), which shows many similarities with the classification of Wagner.

State constraints – The stock for product X must be always of at least 100 units.
Deduction rules – A client younger than 20 has the risk profile 'low'
Calculation rules – the premium for a cancellation insurance is 6% from the total travel costs.
Process constraints – a delivery must be always preceded by the payment.
Production rules – IF the stock for product X is smaller than 100 units THEN place an order for X
ECA-rules – IF the client wants to make a reservation and the client is not on the black list THEN make the reservation
Deontic rules – A declaration for an amount > € 5000,- may only be approved by a senior member of the staff.

The state constraints seem to be similar to the integrity rules of Wagner. The calculation rules seem similar to the derivation rules of Wagner. The process constraints and deontic rules are missing in the classification of Wagner, but the transformation rules of Wagner is missing from the classification of Iacob.

In an attempt to make a more complete classification, Weiden (Weiden, Hermans et al. 2008) made a classification according to the role rules fulfill in the organization. The paper classifies business rules into three categories: structural, behavioral, and managerial rule types. The structural business rules are concerned with describing the static aspects of a business. Behavioral rules define the conditions on the execution of tasks in the business. Managerial rules define higher-level constraints on the business.



Category	Rule Type	Elicitation question
Structural	Concept Structure	What are the central concepts and their relations in the domain?
	Persistency	How long should a representation of a certain object be kept?
	History	Should the history of an object in the domain be recorded?
Behavioral	Information Flow	Which type of information does a task need from other tasks?
	Control Flow	Which rules control the execution order of tasks?
	Pre-conditions	What conditions must be present for a task to be able to execute?
	Post-condition	Which conditions are present after a task has finished executing?
	Frequency	How often will a certain task be performed?
	Duration	How long does or may it take to perform a certain task?
	Task knowledge	What kind of knowledge is needed to perform a certain task?
Managerial	Organization	What can be said about policies in the organization?
	Goal & Value	What is the goal of a process? What is the expected added value of a process?
	Actor[1] Competences	Which skills does an actor need?
	Actor Responsibilities	Which tasks is an actor allowed to perform?
	Resources	Which rules are used to govern the use of resources?

Table 1: Business-Rule Classification Scheme

Example rules:

Structural:
Concept Structure – Each car type has a rental class. For example, a Ford Focus is a class two rental
Persistency – No information about a rental should be kept for longer than five years
History – All rentals made by some customer should be recorded as belonging to that same customer

Behavioral:
Information Flow – When handing over a car, always use the car that was previously assigned to this rental
Control Flow – Cars will only be allocated to customers, if the reservation for the rental is accepted
Pre-conditions – A car must be physically present in a EU-Rent branch, before it can be assigned to a rental
Post-conditions – After a car has been serviced, the car is considered to be in a legal and roadworthy condition
Frequency – Each car must be serviced every three months or 10.000 kilometers, whichever occurs first

Duration – Assessment of a car reservation must be done on the same day the reservation was received
Task knowledge – Rented cars must meet local legal requirements for mechanical condition and emissions for each country that may be visited during the rental.

Managerial:
Organization – We only rent cars in legal, roadworthy condition to our customers
Goal and Value – We want to make sure that there will be no rentals turned down because there are no cars available
Actor Competences – All EU rent employees must have a valid driving license
Actor Responsibilities – Special cars can only be assigned to a rental by senior employees
Resources – In each car group, if a branch loses cars to take it more than 10% of below its quota, it must increase the number back to within 10% of quota by transferring cars from other branches, or buying some cars

2.3.3 CONCLUSIONS

Business rules can be categorized in a number of different ways. We categorize business rule into two main categories. The first category are the action rules which contains all the business rules that are triggered by an event and/or state that certain activities need to be performed. Examples of business rules that belong to this category are *reaction rules (ECA)* and *production rules (CA)*.

The second category contains the rules that put constraints on the structure, behaviour, or information of a system or organizations. Examples of business rules which belong to this category are integrity rules, state constraints, and deontic rules.

In our declarative approach we want to avoid the use of business rules which belong to the first category and focus on rules which belong to the second category. The business rules in this category do not state when certain activities have to be performed.

3 REFERENCE FRAMEWORK CLAIMS BASED WORKING

In this chapter we will formulate a reference framework for *Claims Based Working* using the results of our literature research. The reference framework consists of two major parts. The first part is what we call *Business Activity Access Control* and the second part is the gathering of information in order to make decisions.

Business Activity Access Control consists of a number of components. Access control is one of the components of our reference framework. Using the literature about the different access control mechanisms, we will formulate an *Access Control Meta-model*. Another component of the framework is declarative business processes.

This chapter begins with a description of the *Access Control Meta-model* and declarative business processes. These are combined in the *Business Activity Access Control* model.

In order to enforce access control, information is required in order to make access control decisions. This information could be gathered by the access control enforcer itself, but another method is to use claims to gather this information. This shifts the responsibility of information gathering from the access control enforcer to the subject that wants access.

Combining the *Business Activity Access Control* model and *Claims* will result in a model which will be called *Claims Based Working*. Here we will combine the results of this chapter to form a meta-model for *Claims Based Working*. This last part of this chapter also contains an architecture which will support the meta-model. The figure below provides an overview of the components which together form Claims Based Working.

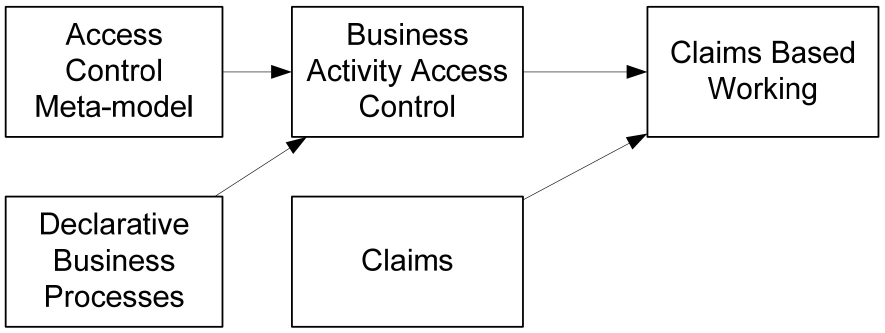

Access Control Meta-model
Business Activity Access Control
Claims Based Working
Declarative Business Processes
Claims

Figure 30 Overview Claims Based Working

3.1 ACCESS CONTROL META-MODEL

In the literature research a number of access control mechanism have been described, like *Discretionary Access Control*, *Mandatory Access Control*, and *Role Based Access Control*. These access control mechanisms all consist of three basic components. The first is the *Subject* that requests access, the second is the *Object* for which access is requested and the third is the *Feature* of the *Object* the Subject wants to perform on the object.

The things that stay the same for each mechanism are the last two components, which are the *Object* and the *Features* of an *Object*. An *Object* has a finite number of *Features*.

For example the object *document X* can have the features *read*, *write*, *delete* and the object *account Y* can have the features *read*, *credit*, and *debit*. *Features* can be shared by multiple *Objects*, like the *read* feature, but the combination of a *Feature* and an *Object* is the specific *Feature* that a specific *Object* has. Access control can be enforced on the combinations of the *Objects* and the *Features*. We will call the combination of an *Object* and a *Feature* an *Object Feature*.

The *Subject* is the entity that requests access to a specific combination of an *Object* and a *Feature*. The different mechanisms differ in the way they assign access right to a *Subject*. The *Discretionary Access Control* mechanism assigns *Subjects* directly to the relevant *Object Features* using access control lists. The *Mandatory Access Control* mechanism classifies the *Object Features* and the *Subject* also has a classification. The *Role Based Access Control* mechanism enforces access control based on *Roles*. Each *Object Feature* has a number of *Roles* which are allowed to perform that specific *Object Feature*. The *Subject* is assigned to one or more *Roles* which grant the *Subject* access to particular *ObjectFeatures*.

The picture below depicts the meta-model on the left and the access control mechanisms in their simplified form on the right. This model separates the *Subject* from the *Object Features*. The way access rights are assigned to a *Subject* depends on the access control model set for that particular *Object Feature*. In this model different *Object Features* can have different access control mechanisms. For example for an object *document X* which has the features *read* and *write*, anybody with a certain classification is allowed to read the document. The *Mandatory Access Control* mechanism can be used to accommodate this. But when only one *Subject* is allowed to write the document, MAC is less suited for this. The *Discretionary Access Control* mechanism is better suited in this situation.

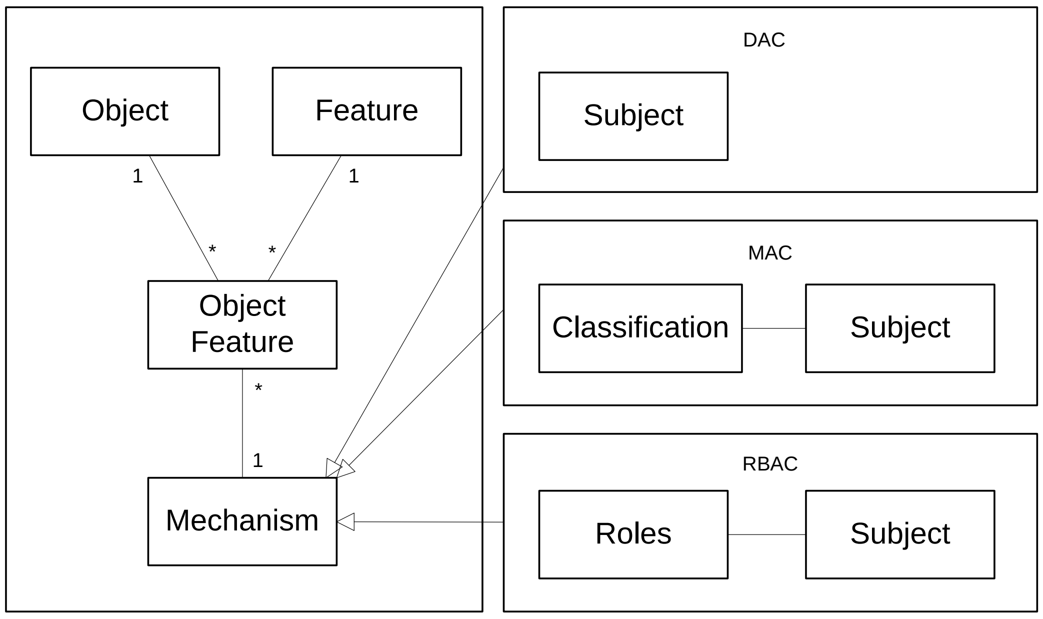


Figure 31 Access Control Meta-model

The Access Control meta-model is an important model which will be used in the *Business Activity Access Control* (BAAC) model. Another important aspect of the BAAC model is the aspect of declarative business processes. The declarative business processes which we will use in particular will be described below.

3.2 DECLARATIVE BUSINESS PROCESS

As mentioned before business processes are a collection of activities that together create output of value to the customer. Traditionally business processes are implemented in a way in which the order and sequence of activities is predefined. This is the imperative approach.

Declarative business processes organize processes in a way in which the order and sequence of business activities is not explicitly specified. With declarative business processes it is possible that the number and order of activities differs each time the business process is executed.

In the figure below we visualize two runs through a process. The thick black arrows indicate the first run through the process. In this case all the activities are executed once in order to deliver the desirable output. The second run is indicated by the dotted arrows. In this case only a subset of the possible activities is executed. The idea behind declarative business processes is that the order in which the activities are executed is not explicitly specified.

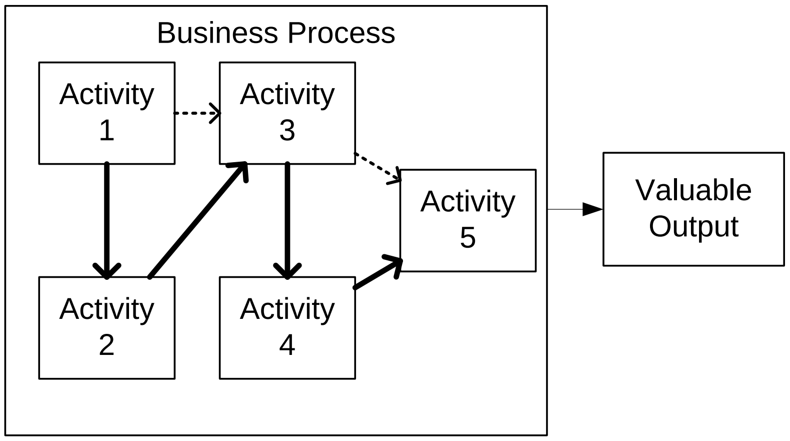


Figure 32 Business Process instances

The difference in the activity sequence between the two runs can be caused by decisions made locally, but the difference in the two runs can also be controlled using constraints. The declarative approach by Pesic (2008) uses constraints based on the events triggered by activities. An example of such a constraint can be the following: “When Activity 1 is completed, then start Activity 2”.

We have a slightly different approach to declarative business processes. The constraints we use are based on data requirements of each activity. Activities produce a certain output which contributes to the output of the business process, but an activity can also require certain input. The input might be required to be used to create or to transform the output of the activity. The output of the activity in turn can be used as input for other activities. The constraints based on input requirements implicitly specify a certain sequence of activities. With this approach it is possible that the input requirements for an activity can be delivered by the output of a number of activities.

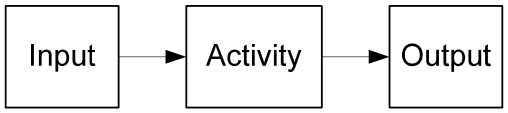


Figure 33 Business Activity input and output

If we assume that the output of a business process is the product of the business process, then the output generated by the different activities in the process can be seen as sub-products. The (sub) products can be material products or immaterial products, but information about these products must be stored in order for them to be used in other activities.

Using the BOM approach the sub-products can be mapped to form the bill of materials of a business process. Each of the sub-products must be created by one or more activities. Some of those activities need other sub-products as input in order to create its output. The sub-products required by those activities are the input constraints of those activities. These constraints need to be fulfilled in order for that activity to be able to produce its output. The constraints can be seen as the access control requirements for activities will be discussed in the next section.

3.3 BUSINESS ACTIVITY ACCESS CONTROL

Business Activity Access Control concerns access control on business activities. The activities in a business process are performed by certain subjects, but not every subject is allowed to perform every activity. Access control is therefore enforced on the business activities. The figure below shows an example of two business processes.

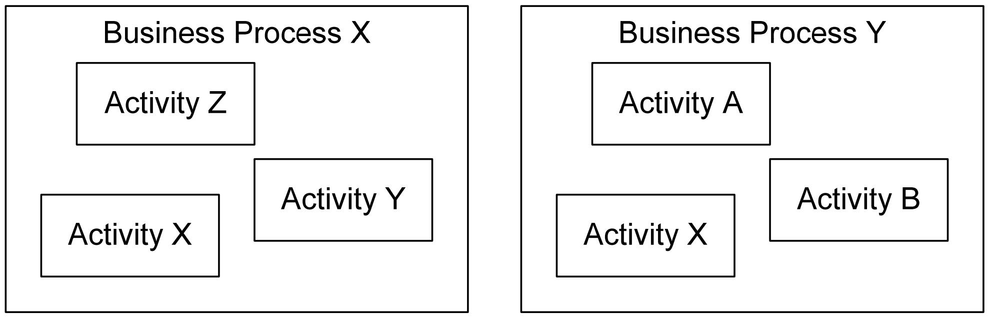


Figure 34 Example Business Processes

The business processes both contain a number of different business activities. Activity X occurs in both business processes, but the authorized subjects for activity X may differ between the two processes. A mechanism is necessary which will accommodate this. Similarities can be drawn between the way business processes are organized and our Access Control Meta-model. The figure below shows the Access Control Meta-model next to a business process.

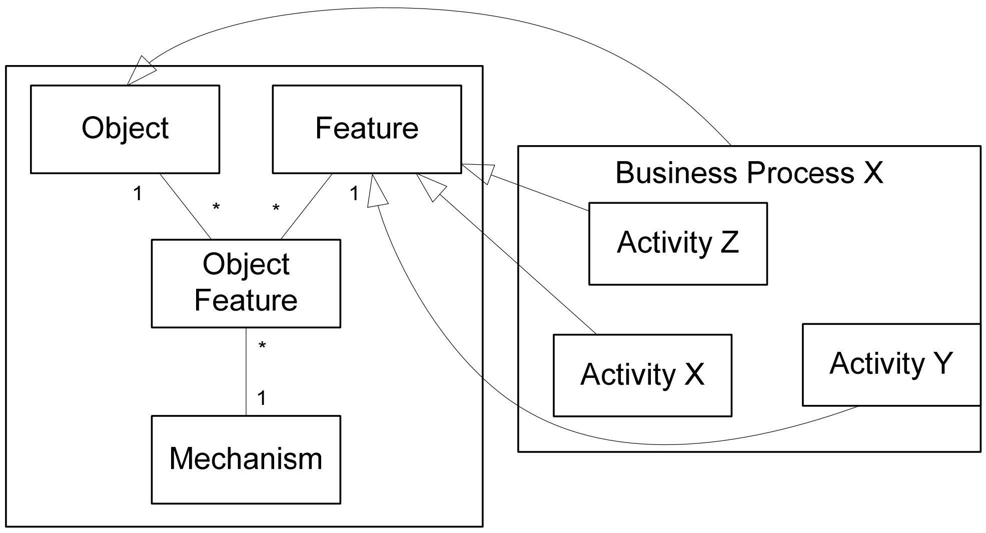


Figure 35 Access Control Meta-model and Business Processes

The *Object* in the meta-model contains a number of *Features* and for each of the *Features* in the *Object* access control can be enforced. A business process contains a number of *Activities* and each of the *Activities* must only be allowed to be performed if the constraints for that activity are met. In other words access control needs to be enforced on the *Activities*. If we apply the Access Control Meta-model on business processes we will get the following Business Activity Access Control (BAAC) model.

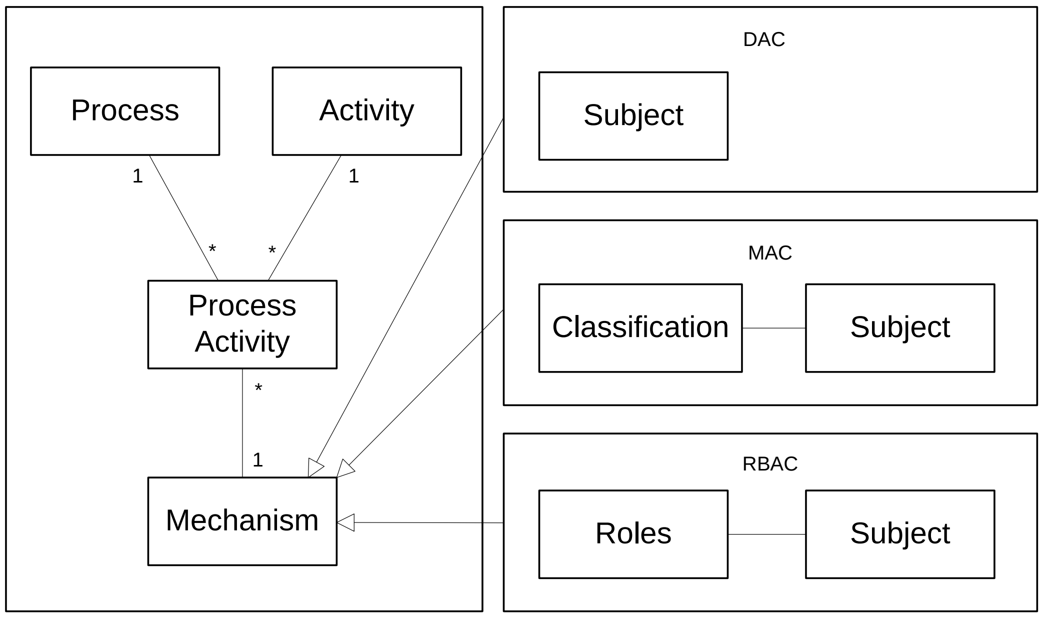


Figure 36 Business Acitivity Access Control model

Using this Access Control model we can assign subjects to each business activity using a number of different access control mechanisms. A subject is authorized to perform the activity if the subject is assigned to it. Subjects can for example be directly assigned to business activities using the *Discretionary Access Control* mechanism, but the BAAC model also enables entities to be assigned to using the role of the entities or the level of classification.

Some activities might allow everybody to perform the activity. A new access control mechanism can be introduced to accommodate this. This mechanism is called the “All-or-Nobody” access control mechanism. Either everybody is allowed to perform the activity or nobody is allowed to perform the activity. Figure 37 illustrates this access control mechanism. When everybody is allowed access the “All-or-Nobody” mechanism is in sense an empty black list. Nobody is on the black list, so everybody is allowed access. When nobody is allowed access, this mechanism is in sense an empty white list. Nobody is on the list, so nobody is allowed access.

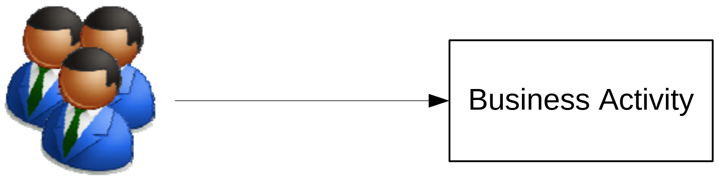


Figure 37 All-or-Nobody Access Control

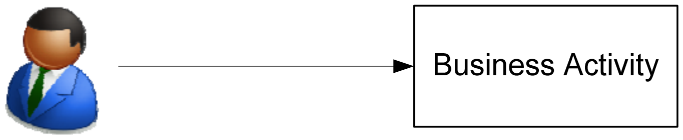


Figure 38 Discretionary Access Control

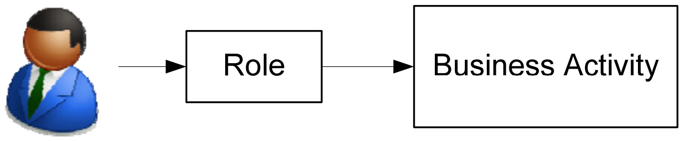


Figure 39 Role Based Access Control

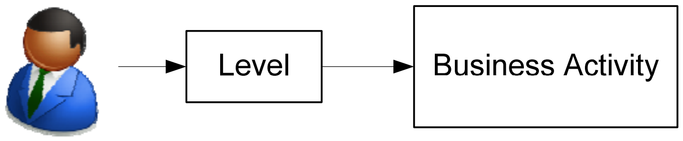


Figure 40 Mandatory Access Control

Whether or not a subject is authorized to perform an activity is not the only factor which determines if an activity is allowed to start. As discussed earlier an activity might also have input constraints. Using the Bill-of-Materials approach the input constraints for each activity can be made clear. Each activity has a certain product as output and in order to be able to produce the output certain other products may be required. The input constraints can be represented by business rules.

There are many sorts of business rules, but not all of them are suited to be used in a declarative manner. The business rules we use in our approach are business rules which are constraints on the structure, behavior, or information of an organization or system. An example of business rules which can be used in a declarative manner are the deontic rules. An example of a deontic rule is:

“A declaration for an amount > € 5000,- may only be approved by a senior member of the staff.”

The rule above can apply to a *Declaration Approval activity* and describes who is allowed to approve declaration. In this case the rule states that declaration above a certain amount can only be approved by a senior member of staff. If a junior member of staff gets a declaration with an amount above € 5000,- then the *Declaration Approval activity* is not allowed to start. If the amount of the declaration is below € 5000,- or the staff member is a senior member, then the *Declaration Approval activity* is allowed to start.

Business rules consist of terms and facts. The terms in the above business rule are the amount of a declaration and the role of a staff member. Information about these terms must come from somewhere. Either the information is gathered by the party that enforces the access control or this information is supplied by the party that wants access. We choose the latter approach, and propose Claims as means to gather this information. Using Claims the party that want access is responsible to deliver the required information to the party that enforces access control in order for it to make an access control decision.

3.4 CLAIMS

Claims Based Access Control (CBAC) is a mechanism to gather information in order to make access control decisions. Traditionally this information is gathered by the party that makes the access control decisions, but CBAC shifts this responsibility to the party that requests access. Instead of the enforcer gathering information to make a decision the requester needs to gather this information to allow the enforcer to make a decision. The mechanism uses *Claims* to gather information for decision making.

In order to evaluate the business rule presented in the previous section, two things needed to be known in order to evaluate the rule. The first thing was the amount of the declaration and the second was the role of the user. This information has to come from somewhere. For this rule the information likely is available in the company’s own databases, but things get more complicated when information is required that is not readily available. Consider the following rule: High risk customers are not allowed to place rush orders.

Whether or not a customer is a high risk customer has to be determined somehow. Customers with low creditworthiness can be defined as high risk customers, but his information has to come from somewhere. In the Netherlands there is an organisation called the BKR which registers the creditworthiness of people. This information has to be retrieved from the BKR.

A mechanism which enables to retrieve such information is Claims Based Access Control. The literature about CBAC and the Identity Metasystem (IdM) describes a mechanism to gather claims about the subject that wants access to certain resources. The literature only describe scenario’s where information is gathered about the entity that requests access, but the information that needs to be gathered to evaluate business rules may concern more than one entity. The *Claims* we want to use in Claims-Based Working must be about more than the entity that is responsible for the activity.

For example if we have the rule that high risk customers are not allowed to place orders and an employee with the role of administrative worker wants to place an order on behalf of the customer, then information is required about the employee and the customer. Information is required about the employee in order to answer the question whether or not the employee is authorized to execute the activity and information about the customer is necessary to determine whether the customer is high risk or not. As long as the claims provider is able to deliver the requested claims it should not be a problem to get information about multiple entities.

3.5 CLAIMS BASED WORKING

Claims Based Working is a way of organizing business processes where the subjects which are involved in the business process are able to decide which activities are needed and in which order these activities need to be executed to produce the desired output of the business process. Claims Based Working uses declarative business rules to set the constraints for each activity and Claims are used to gather information in order to evaluate the business rules. To achieve this we combine a number of components.

The business processes are organized in a declarative manner and each of the activities in the business process produces a sub-product which contributes something to the end product of the business process. In order to find the relationships between the different sub-products we use the Bill-of-Materials approach to get a product/data model of the business process.

Another component is the Business Activity Access Control model. Each activity is only allowed to start if the constraints for that that activity are met. The constraints are represented by business rules. These rules come from different layers in the organizations and are managed somehow to ensure that the rules which apply to an activity are actually enforced by the BAAC model.

In order to make access control decisions, information is required about relevant objects and entities. These objects and entities occur as terms in the business rules which are used to make access control decisions. This required information is gathered somehow and the mechanism behind CBAC enables this information to be gathered.

3.5.1 CLAIMS BASED WORKING META-MODEL

The model below shows how the different components relate to each other. With the results of the Bill-of-Materials approach the product of the business process and the sub-products are clear. Using the results of the BOM approach the business activities in the business process can be organized. The BOM approach also delivers a number of constraints of the required products. These constraints are managed using a certain Business Rule Management system. The Business Activity Access Control model uses claims and business rules in order to provide access control to the business activities.

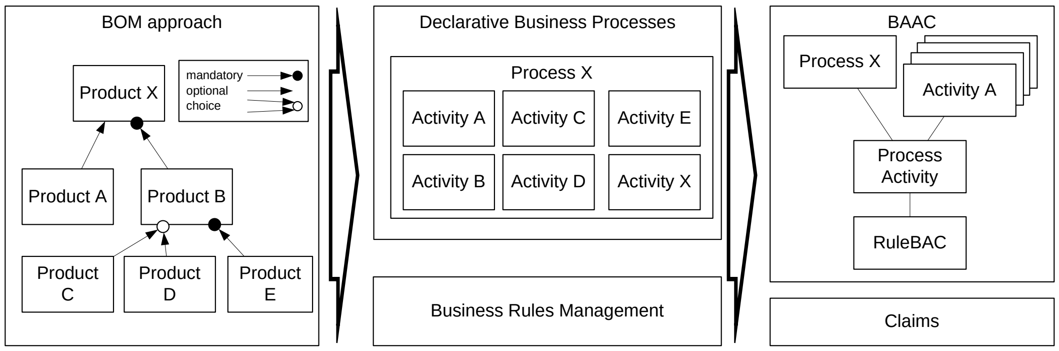


Figure 41 Claims Based Working Meta-model

To demonstrate how CBW works in practice an architecture model is devised which supports the Claims Based Working meta-model.

3.5.2 CLAIMS BASED WORKING ARCHITECTURE

To organize a business process according to CBW, an architecture is required which supports CBW. A number of components are used in the CBW architecture. The figure below shows the Claims Based Working architecture.

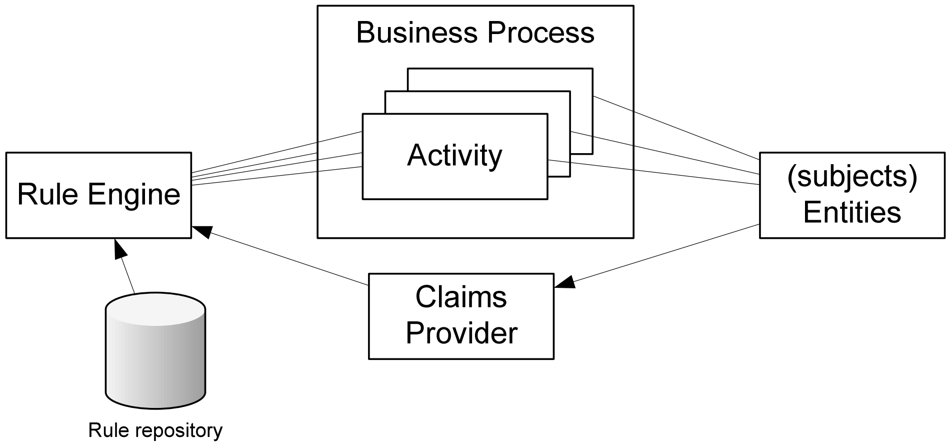


Figure 42 CBW Architecture

The *Business Pro*cess and its *Activities* are connected to a *Rule Engine* which enforces access control on the *Activities*. The *Rule Engine* makes the decision whether or not a *Business Activity* is allowed to start. To make decisions, the *Rule Engine* requires *Business Rules* which apply to the *Activities* and information about the terms which occur in the business rules.

The *Rule Repository* contains all the rules which apply to the *Activities*. Each time the *Rule Engine* requests the rules for a specific *Activity* the *Rule Repository* generates a list with all the rules which apply to that *Activity*. How the rules are generated and managed is out of the scope of this research.

The information which is necessary to evaluate the rules is gathered using *Claims*. The information about the terms in the business rules is gathered using *Claims* provided by the *Claims Provider*. The *Claims Provider* does not deliver *Claims* automatically.

The *Rule Engine* generates a list of all the *Claims* it requires in order to make a decision. This list of required *Claims* is gathered by the *Entity* which wants access to the *Business Activity*. That *Entity* gets the *Claims* from a *Claims Provider* and passes them on to the *Rule Engine*. The *Rule Engine* can then make an access control decision using the *Business Rules* and the *Claims* about the terms in the business rules.

The architecture as described above will be implemented in a number of demonstrators. The demonstrators are built using PHP and Java. The next chapter will discuss the demonstrators which were built during this research.

4 PROOF OF CONCEPT

In this section we describe the demonstrators which were made to test a number of concepts used in Claims Based Working. These concepts are the Access Control meta-model, the rule engine, the Claims gathering mechanism, and Claims Based Working. The approaches will be tested in three demonstrators.

The first demonstrator tests the Access Control Meta-model using two simple access control mechanisms, namely the All-or-Nobody Access Control mechanism and the Discretionary Access Control mechanism. This demonstrator allows us to test the Access Control meta-model and what implications the meta-model has.

The second demonstrator uses the Access Control meta-model from the first demonstrator and adds two advanced access control mechanisms, which are the Rule Based Access Control mechanism and the Claims Based Access Control mechanism. This second demonstrator is meant to see what the implications of the rule engine and the Claims gathering mechanism are. The rule engine will be used in the third demonstrator to make an access control decision and the Claims gathering mechanism will be used to gather information which will be used by the rule engine to make an access control decision.

The third demonstrator is the proof of concept of Claims Based Working. In this last demonstrator we will explain how the different components come together. The third demonstrator will be used in the next chapter for validation of Claims Based Working.

4.1 DEMONSTRATOR 1: TESTING THE ACCESS CONTROL META-MODEL

This demonstrator will implement the Access Control Meta-model. This demonstrator will use an office environment scenario to show how the Access Control Meta-model works. In this demonstrator we will use two simple access control mechanisms. The first being the All-or-Nobody Access Control mechanism and the second is the Discretionary Access Control mechanism.

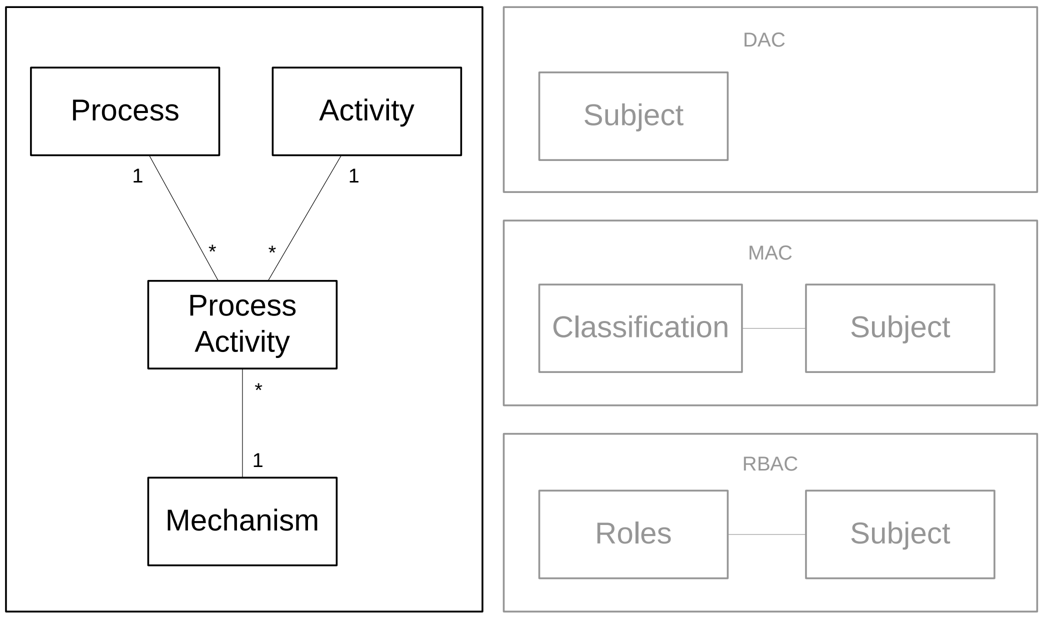


Figure 43 Overview AC Meta-model demonstrator

4.1.1 SCENARIO

The scenario is based on an office environment. In such an environment there are walls separating the different rooms. The doors provide access to rooms behind them. Doors can be opened mechanically with or without keys or electronically. Doors that have an electronic locking mechanism are usually controlled by a computer.

In such an environment there are a number of object on which access control is applied, for example doors need to be opened and computers need to be turned on. The scenario is depicted in the figure below.

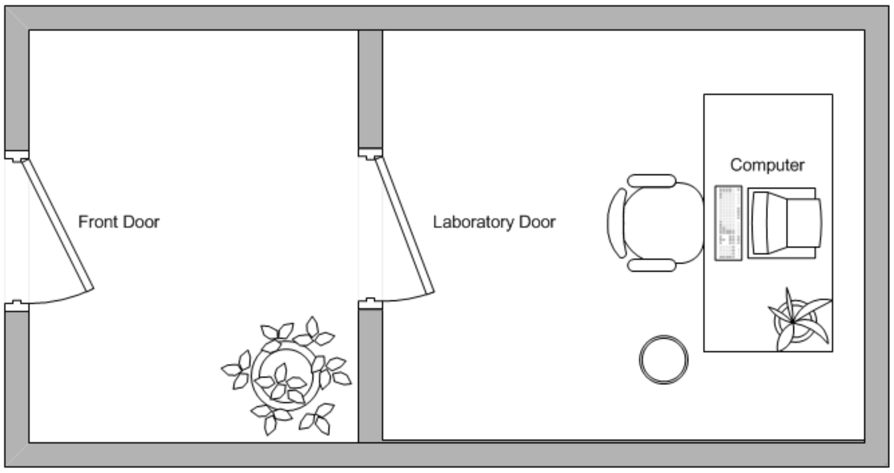


Figure 44 Office environment scenario

In this scenario we have two doors, one laptop, and two virtual workplaces. The virtual workplaces can be accessed via the laptop. The laptop in turn is accessible by going through two doors. Each of the objects belongs to a certain category. The table below show which object belongs to which category.

ObjectCategory	Object
Door	Front door
	Laboratory door
Laptop	Laptop1
Workplace	Workplace1
	Workplace2

Table 2 Object Categories

The *ObjectCategories* used in this scenario are doors, laptops, and workplaces. Each *ObjectCategoy* has a number of *Features*. The *Features* of every *ObjectCategory* is listed in the table below.

ObjectCategory	Feature
Door	Ring bell
	Open door
Laptop	Turn on
	Login
Workplace	Start application A
	Start application B
	Start application C

Table 3 Object Features

There are two *Objects* (front door and laboratory door) which belong to the *ObjectCategory* "door". A door has the *Features* "ring bell" and "open door". In this scenario every *Object* has the same features as the *ObjectCategory* where the *Object* belongs to. Meaning that the both the front door as the laboratory door have the same *Features* as the *ObjectCategory*. The table below gives an overview of all the *Features* of the different *Objects*.

Category	Object	Feature
Door	Front door	Ring bell
		Open door
	Laboratory door	Ring bell
		Open door
Laptop	Laptop1	Turn on
		Login
Workplace	Workplace1	Start application A

		Start application B
		Start application C
	Workplace2	Start application A
		Start application B
		Start application C

Table 4 Overview of Objects

4.1.2 ACCESS CONTROL MECHANISMS

ALL-OR-NOBODY ACCESS CONTROL

This is an access control mechanism in which either everybody has access or nobody has access. This mechanism does not differentiate between different users. Therefore there is no need for additional information. This mechanism may not be very useful in practice, but it does allow us to determine where to put the access control.

For example if we have determined that everybody had access to open all doors, then one access control mechanism could be placed on the category *Door*. But if we suddenly decide that the access to one particular door needs to be denied, then we will need to put an access control mechanism on the "open door" feature of that particular door. When the access control decisions for each instance of a category are the same it is sufficient to put an access control mechanism on the category. An access request on an instance of that category will result in the same decision as the access control decision for that category. When an access control decision of an instance of a category differs from the category, then an access control mechanism has to be put on that particular instance.

Putting access control mechanisms on categories, separate features, and maybe even the different objects can make access control complex. We decided to put the access control mechanism on the features of all the objects. This will reduce the complexity of managing the access control mechanisms, because an access control mechanism is allocated to every feature of an object, instead of the object itself or category or a combination those. As a result the number of access control instances that have to be managed will be relatively high. To illustrate how access control instances are allocated consider the following example:

An object "front door" of the category "door" can have a number of features. Even if all the features of all the doors uses the same access control mechanism and the access control decision is the same for all the features, then all the features of all the doors will still have a separate access control mechanism allocated to it.

How the access control instances are actually managed is outside of the scope of this research.

DISCRETIONARY ACCESS CONTROL

A discretionary access control mechanism is usually implemented using access control lists. In our implementation every *ObjectFeature* has an access control list. The list contains the names of *Subjects*, which have access to that *ObjectFeature*. The DAC mechanism will grant access if a *Subject* is on the list and deny access if the *Subject* is not on the list. When the evaluator receives an access control request containing an *ObjectFeature* the user is prompted to enter his or her name and password. The evaluator then determines if the user is on the list and if the password is correct. Note that verifying the password is authentication and is not part of access control itself. The evaluator returns the value "True" if the user is on the list and the value "False" is returned if the user is not on the list.

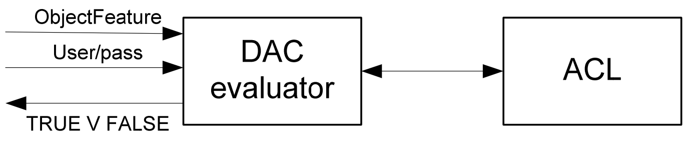


Figure 45 Overview DAC mechanism

4.1.3 IMPLEMENTATION

The demonstrator is implemented using PHP for the programming and MySQL is used as the database server. The underlying database had the following structure:

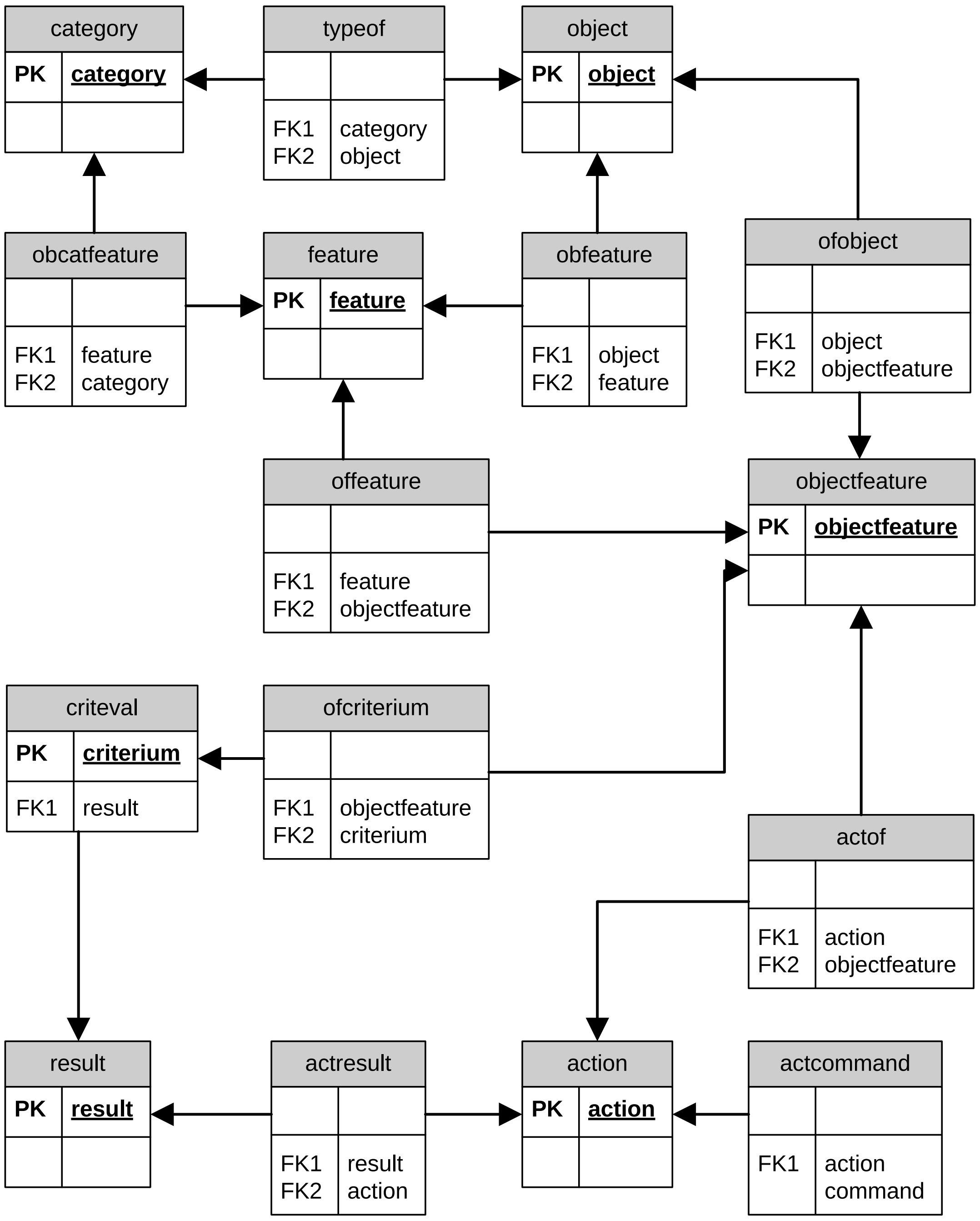


Figure 46 Database diagram office scenario

The *category* table contains the different categories (e.g. Door, Workplace). Each *category* has a number of *features*. *Features* are linked to *categories* by the *obcatfeature* relation. A *category* can have a number of instances. These instances are the *objects* (e.g. Front Door, Laboratory Door). *Objects* have the same features as the *categories* to which they belong. The combination of the *objects* and the *features* are represented by the *objectfeatures*.

An access control mechanism is placed on each *objectfeature*. The type of access control mechanism used depends on the *ofcriterium*. The access control decisions of the different access control mechanism are represented by the *results*. The access control decision is either true or false.

The *action* that will be performed after an access control decision depends on the *result* of that decision and also on the *objectfeature* for which the access control decision was made.

The communication with the database is handled by one PHP class (DB-handler). Another PHP class (Communicator) shows the webpage and handles the input from the user.

When the communicator class receives a command to evaluate an *objectfeature*, it calls the function in the DB-handler class to get the appropriate *action* for this *objectfeature*. The appropriate *action* is selected based on the *Result* of an *objectfeature*. Each *objectfeature* has an *action* for every possible *result*. Which *action* is selected depends on the access control decision of the access control mechanism. In this case the *result* is directly linked to a *criterion* and each *objectfeature* has a *criterion*, meaning that the appropriate *action* can be selected using a “simple” SQL-query:

```
SELECT actshellcommando.shellcommando FROM actshellcommando
        WHERE (SELECT DISTINCT `f1`.`actie` AS `Actie`
                FROM `rfresource`
                LEFT JOIN  ( SELECT DISTINCT isect0.`resourcefeature`, isect0.`actie`
                        FROM ( SELECT DISTINCT fst.`resourcefeature`, snd.`actie`
                                FROM `rfcriterium` AS fst,
                                        ( SELECT DISTINCT fst.`criterium`, snd.`actie`
                                        FROM `criteval` AS fst, `actuitkomst` AS snd
                                        WHERE fst.`uitkomst` = snd.`uitkomst`
                                        ) AS snd
                                WHERE fst.`criterium` = snd.`criterium`
                                ) AS isect0, `actrf` AS isect1
                        WHERE (isect0.`resourcefeature` = isect1.`resourcefeature` AND
                        isect0.`actie` = isect1.`actie`) AND isect0.`resourcefeature` IS NOT
                        NULL
                ) AS f1
        ON `f1`.`resourcefeature`='$rfid'
WHERE `rfresource`.`resourcefeature`='$rfid') = actshellcommando.actie;
```

4.1.4 RESULTS

After making this demonstrator a number of conclusions can be drawn regarding the concepts used and the design decision made. The first remark concerns the relation between the *Criteria* and *Result*. This demonstrator implements an All-or-Nobody access control mechanism. The *Criterion* set for an *ObjectFeature* directly relates to the *Result* and *Action* of an *ObjectFeature*. Unlike most access control mechanisms the All-or-Nobody access control mechanism does not differentiate between the subjects which make the access request. The access control decision for a specific *ObjectFeature* is the same for every subject that request access to it. For most other access control mechanisms the result is not directly dependant on the *Criterion*. The access control mechanism determines the result and not the *Criterion* set for an *ObjectFeature*.

The second remark concerns the number of *Criteria* that need to be set. This implementation requires that each *ObjectFeature* has a *Criterion* set for it, resulting in a high number of *Criteria* that need to be set. Another choice could have been to *Criteria* on *Objects*. This could reduce the number of *Criteria* that need to be set, but also reduces the adaptability of the system. We chose to set a *Criterion* for every *ObjectFeature* to allow a higher adaptability and to reduce the complexity of managing the *Criteria*. The way all the *Criteria* needs to be managed falls outside the scope of this research. The system that manages the *Criteria* should be able to set the *Criteria* for a large range of *ObjectFeatures* and still be able to set a different *Criterion* for a specific *ObjectFeature*. The system which manages the *Criteria* should be able to cope with coarse and fine granularity. In our implementation we manually set the *Criteria* for each *ObjectFeature*.

The third remark concerns the required components in order to make access control decisions. A number of concepts are not essential order to make access control decisions. An access control decision is made based on the *Criterion* set for an *ObjectFeature*, making the *ObjectCategory* redundant and unnecessary. The concept *Action* is also not directly part of access control mechanisms. When all these unnecessary concepts are taken out of the model it will look as follows:

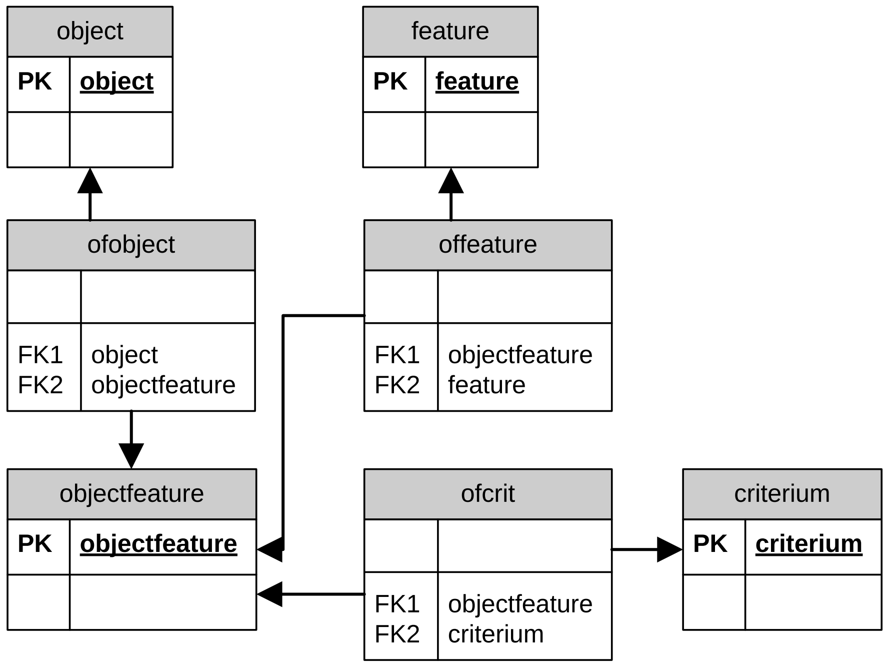


Figure 47 Optimized database diagram office scenario

4.2 DEMONSTRATOR 2: TESTING ADVANCED ACCESS CONTROL MECHANISMS

The first demonstrator has implemented the Access Control Meta-model. In this second demonstrator we use the results of the first demonstrator and implement two additional access control mechanisms. The mechanisms which will be implemented are the Rule Based Access Control mechanism and the Claims Based Access Control mechanism.

Rule Based Access Control can provide access control based on rules. This mechanism will later on be used to evaluate the business rules of the declarative business processes.

Claims Based Access Control is a mechanism to get authenticated information from third parties. One goal of this demonstrator is to show how business rules can be used to provide access control. The other goal of this demonstrator is to gather information from a third party which can be used to make an access control decision. We use the CBAC mechanism only to gather information which will be used by the party that makes the access control decision. This information will later be used to evaluate business rules.

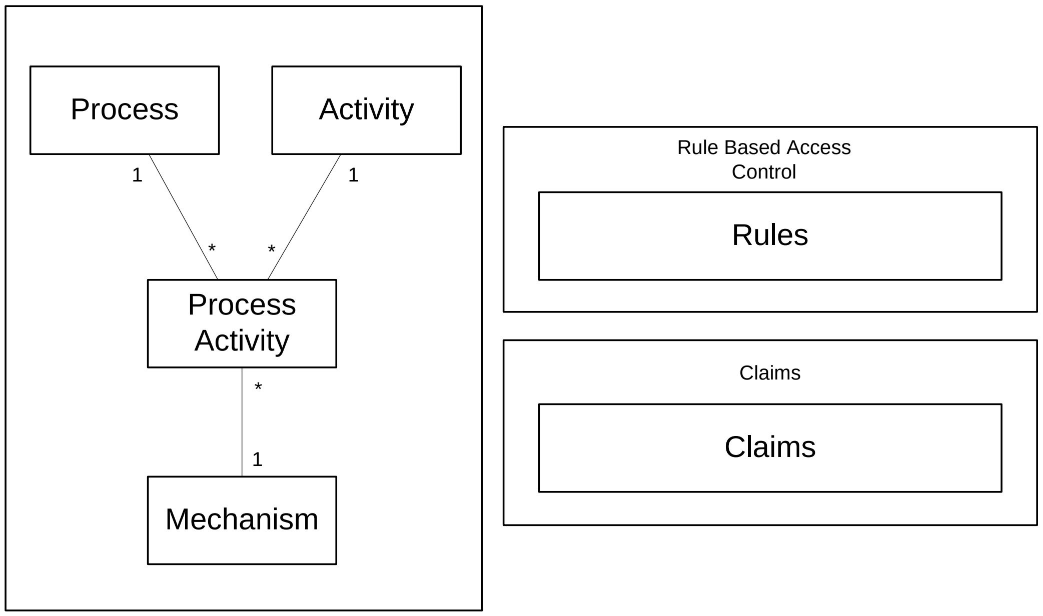


Figure 48 Overview advanced AC mechanisms

4.2.1 RULE BASED ACCESS CONTROL

Rule Based Access Control is a mechanism which enforces access control based on rules. This is the access control mechanism which will be used in Claims Based Working to enforce the data requirements of the business activities.

The Rule Based Access Control mechanism grants access to an object if all the rules which apply to that object are upheld. Access is denied if the rules are violated and also if no decision can be made.

There are two reasons why a decision cannot be made. The first reason is that there are no rules available to evaluate and the second is reason is that there is not enough information available about the terms which occur in the rules. To illustrate the last reason consider the following rule: “High risk customers may not place rush orders.”

The rule engine needs some information about the terms “customer” and the term “order”. If there is no information available about the customer, the rule engine is unable to determine if the customer is a high risk customer or not.

If the rule engine has not got the information available locally, it will request the subject to provide the required information.

IMPLEMENTATION

There are three components necessary in order to evaluate rules. First of all we need a rule engine, a repository containing business rules, and information to evaluate the rules against. The rule engine used to build the Rule Based Access Control mechanism is the Drools rule engine from JBoss. We chose for this rule engine because it is a customizable rule engine. Another reason to use this rule engine is that it is open source. Drools is a business logic integration platform (http://jboss.org/drools/), which is divided into four projects.

The rule engine itself is called Drools Expert. It consists of a number of Java libraries, which can be used to build a custom rule engine. The rules which are used by Drools Expert have the following basic form:

```
rule “RULENAME”
        when
                CONDITION
        then
                ACTION
end
```

The action part of the rule is executed when the condition of the rule is true. This structure is used to get a response from the rule engine which is a “True” response, a “False” response or a “Void” response. In order to do so each business rule will have a “True” rule, a “False” rule and a “Void” rule. The rules which are loaded in the engine looks as follows:

```
rule “True”
        when
                CONDITION
        then
                True
end

rule “False”
        when
                CONDITION
        then
                False
end

rule “Void”
        when
                CONDITION
        then
                Void
end
```

When the rule engine is able to make a decision a response is sent which is either a *True* response or a *False* response. When there is not enough information to make a decision the rule engine sends a *Void* response back including what kind of information is missing.

With the Rule Based Access Control mechanism it is also possible to implement other access control mechanisms. The All-or-Nobody access control mechanism can be implemented as follows.

When everybody as access to an *ObjectFeature* the rules looks like:

```
rule “True”
        when
                1==1
        then
                True
end

rule “False”
        when
                1==2
        then
                False
end
```

The condition of the “True” rule is true, because 1 equals 1. The condition of the “False” rule is false, because 1 does not equal 2. The rule engine will return a true because the “True” rule applies. When nobody has access to an ObjectFeature the rule looks like:

```
rule “True”
        when
                1==2
        then
                True
end

rule “False”
        when
                1==1
        then
                False
end
```

In this case the rule engine will return a false because the “False” rule applies.

The Role Based Access Control mechanism can be implemented in the following way.

If access is allowed for everybody with a “Desk Clerk” role the rules will look like this:

```
rule “True”
        when
                role == “Desk Clerk”
        then
                True
end

rule “False”
        when
                role != “Desk Clerk” AND role != null
        then
                False
end

rule “Void”
        when
                role == null
        then
                Void
role required
end
```

The “True” rule checks whether the role of the Subject is “Desk Clerk”. The “False” rule check whether the role is not “Desk Clerk” and is not null. When a role is null it means that no information is available about the role of the *Subject*. If there is no information available about the role of the *Subject* the rule engine will return void and will state that it requires information about the role of the *Subject*.

The activity diagram below illustrates how the Rule Based Access Control mechanism works.

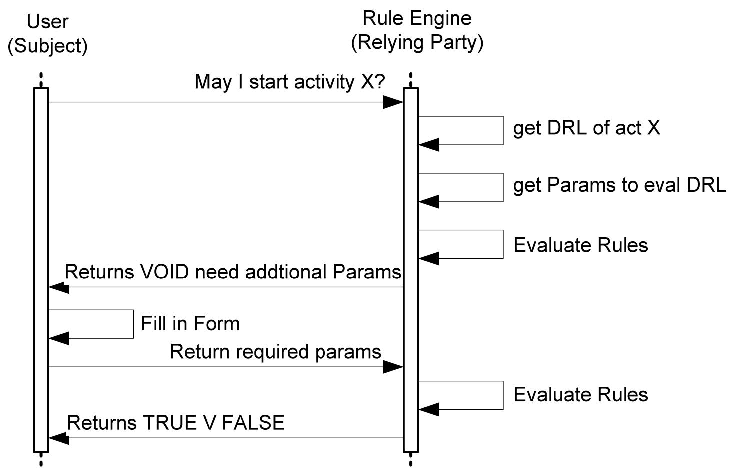


Figure 49 Sequence diagram Rule Based Access Control

The rule engine start working when a *Subject* sends an access request on an *ObjectFeature*, which in this case is *activity X*. The rule engine gets the rules which apply to *activity X* from the rule repository and retrieves any information locally available about the terms which occur in the rules. After the rules and information about the terms are loaded in the rule engine a decision is made. In the diagram above the rule engine is unable to make a decision, so therefore a *Void* response is send back to the *Subject*. The response also states what kind of information is required in order to make a decision.

The *Subject* has to provide this information to the rule engine. In this case the *Subject* provides the requested information directly by filling in a form on the screen. This information is sent back to the rule engine as a new access request. The rule engine will now respond with either a *True* response or a *False* response.

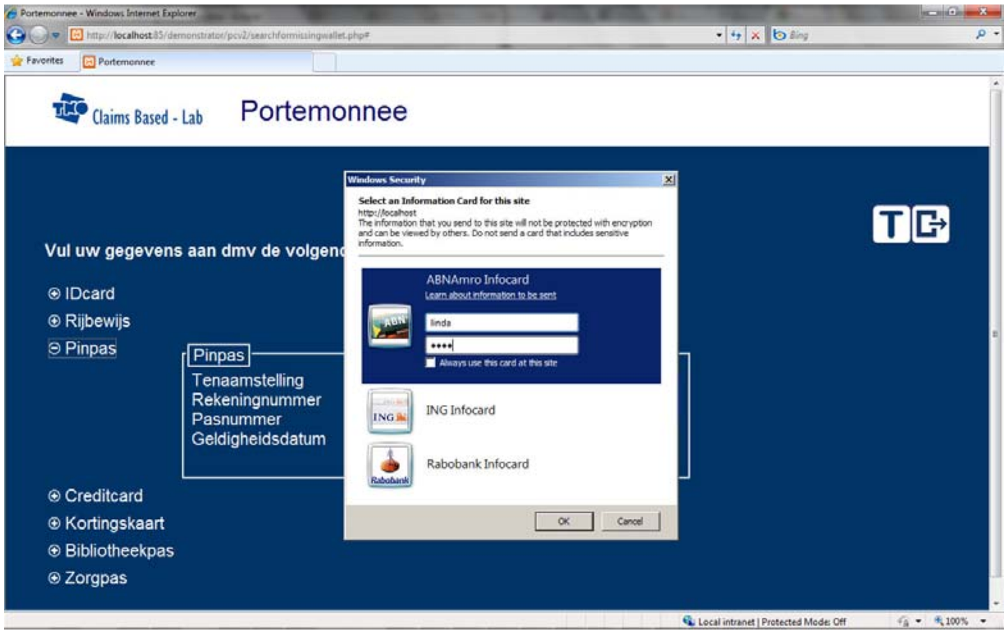
4.2.2 CLAIMS BASED ACCESS CONTROL

The Claims Based Access Control (CBAC) mechanism is a means to make access control decision based on Claims. When the relying party receives the Claims which were requested an access control decision still has to be made. The standard allows for different access control mechanism to be implemented. An access control decision is made based on these Claims. The Claims themselves do not directly determine the outcome of the access control decision. We test the CBAC mechanism by checking whether or not the claims are gathered. If so access is granted and otherwise access is denied.

These Claims can be gathered from external sources. The Claims Based Access Control mechanism from Microsoft implements the WS-* standards which are standardized by OASIS.

The Microsoft solution requires a number of different software packages to work together (Geneva Server, CardSpace). These software packages have to be installed on all the *Relying Parties* and *Identity Providers*. The Subject that requests access only has to have CardSpace installed.

The figure below shows a screenshot of CardSpace. The pop-up window shows the *Identity Providers* that are able to deliver the *Claims* requested by the *Relying Party*. The *Subject* can choose from which *Identity Provider* the *Claims* gathered.



Select an Information Card for this site
http://localhost
The information that you send to this site will not be protected with encryption and can be viewed by others. Do not send a card that includes sensitive information.

Figure 50 Screenshot CardSpace

CardSpace is the piece of software that handles all the communication with the *Relying Parties* and the *Identity Providers*. CardSpace uses InfoCards to communicate between the different parties. In order for a Subject to be able to request access using CBAC at least a few InfoCards need to be installed. The *Subject* needs to download and install InfoCards from every *Identity Provider* which is able to provide Claims about the *Subject*.

An InfoCard is basically a list of all the available Claims types about the *Subject*. The actual values of the Claim types are provided by the *Identity Provider*.

When a Subject request access the Relying Party responds by stating which Claims are required and which Claims are optional and by whom these Claims need to be issued. The CardSpace software automatically shows a list of Identity Providers which are able to deliver these Claims. The Subject can then select one of these Identity Providers. After the Subject is authenticated the Identity Provider issues a certificate containing the requested Claims. This certificate is send to the Relying Party, which then is able to make an access control decision. The figure below shows the communication between the different parties in order to make an access control decision using the CBAC mechanism. In this demonstrator we test whether or not the requested claims can be gathered. If the requested Claims are gathered the access control decision is positive and otherwise a negative access control decision is made.

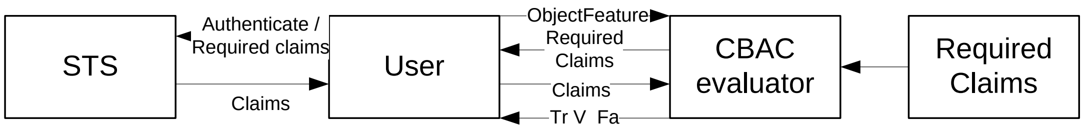


Figure 51 Overview Claims Based Access Control mechanism

4.2.3 RESULTS

The Rule Based Access Control mechanism requires information to be delivered by the *Subject* which requests access. The mechanism expects the Subject to deliver this information directly without authenticating this information. In a later version the Rule Based Access Control mechanism will work together with the Claims Based Access Control to get authenticated information to make an access control decision.

The Claims Based Access Control mechanism is able to get the requested *Claims* from different *Identity Providers*. In this demonstrator the CBAC mechanism is only able to get *Claims* about the *Subject* that requests access. The *Identity Providers* therefore have to be adapted to be able to deliver *Claims* about multiple *Subjects* and *objects* in one request. Information about *objects* can be any information which does not have a direct relation with the *Subject* requesting access, but is necessary in order for the *Relying Party* to make an access control decision. A constraining factor is that CBAC uses *ClaimTypes* to define which *Claims* can be gathered. Due to the fact that *ClaimType* is a type of *Claim* and not an identifier to a specific *Claim*, it is hard to gather *Claims* from an *Identity Provider* about a *Subject* which is not uniquely related to the *Subject* that is gathering the *Claims*.

This adaption would be necessary to gather *Claims* about multiple terms with occur in the business rules, but we have not yet solved this issue. It could be that there are Claims gathering mechanisms out there which have overcome this issue, but have not found them yet. Additional research is needed to find a way to gather Claims about multiple Subjects and entities.

DEMONSTRATOR 3: CLAIMS BASED WORKING

This demonstrator implements a proof of concept of our CBW reference architecture. It builds upon the results of the previous demonstrator to create a prototype of Claims Based Working. The prototype allows us to see how the theories we have developed about Claims Based Working work in practice and allows us to identify the limitations of the demonstrator. The figure below shows how the different components in the Claims Based Working architecture are related.

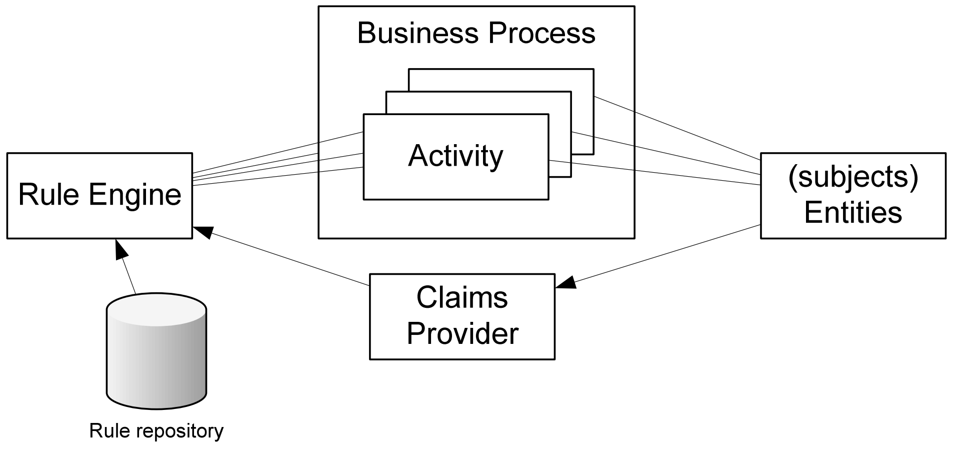


Figure 52 Overview Claims Based working

The basic components we need for Claims Based Working are the following components:

- a component which manages the business processes,

- a component which evaluates business rules,
- a component which is able to generate a list of business rules for each request,
- and a component which is able to deliver information which is required to evaluate the business rules.

We will first discuss the different components which we will combine to form the Claims Based Working architecture. Secondly we will discuss the different scenarios which were implemented using this architecture.

4.3.1 DECORUM COMPONENT

The *Decorum* is the component that manages the business processes and the component which *Subjects* actually see. The component enables subjects to navigate and work through the process. For example a process called *Process X* could contain the activities *A*, *B*, *C*, and *D*. When entering *Activity B* a subject will see the actual implementation of the activity and a number of other potential activities which the subject can perform. The figure below shows a representation of this.

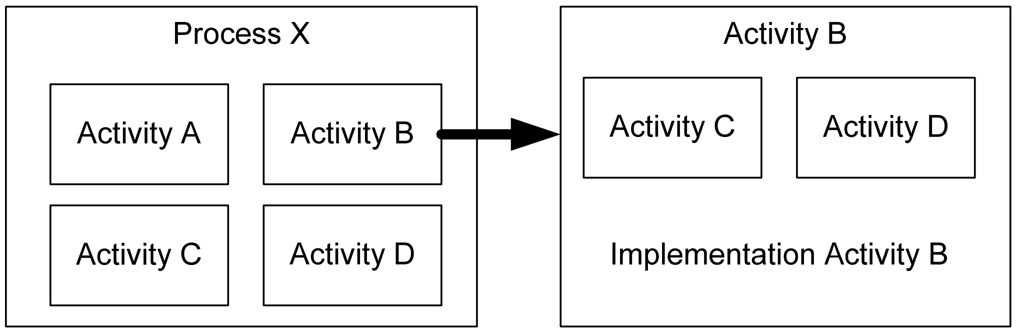
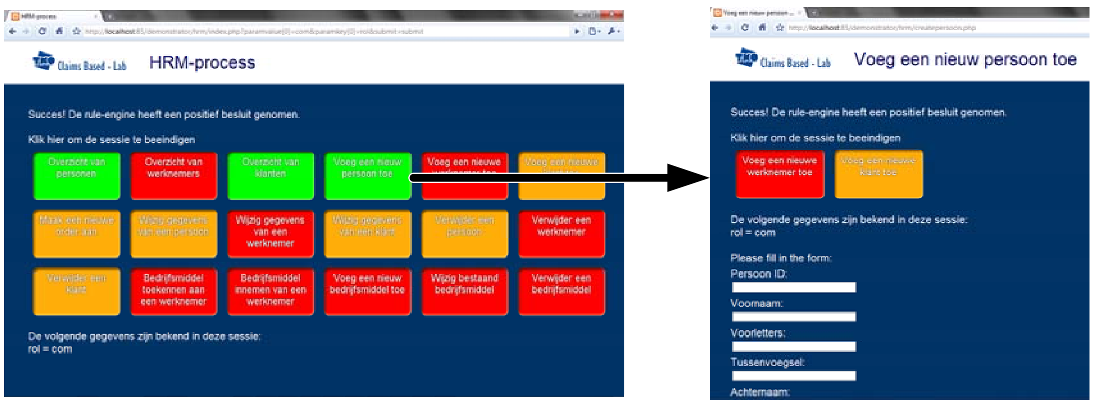


Figure 53 Example Business Process

The potential activities which a subject can perform after completing the current activity are displayed in the activity to enhance user friendliness.

The screenshot below shows the implementation of the *Decorum*. An instance of the *Decorum* has a number of colored buttons. Each of these buttons represents a business activity and has one of three colors, indicating whether or not the activity may be executed. Green colored buttons are links to activities which are allowed to be executed. Red and yellow colored buttons are the activities that may not be executed.


Claims Based - Lab
HRM-process
Overzicht van personen
Overzicht van werknemers
Overzicht van klanten
Voeg een nieuw persoon toe
Voeg een nieuwe werknemer toe
Voeg een nieuwe klant toe
Maak een nieuwe order aan
Wijzig gegevens van een persoon
Wijzig gegevens van een werknemer
Wijzig gegevens van een klant
Verwijder een persoon
Verwijder een werknemer
Verwijder een klant
Bedrijfsmiddel toekennen aan een werknemer
Bedrijfsmiddel innemen van een werknemer
Voeg een nieuw bedrijfsmiddel toe
Wijzig bestaand bedrijfsmiddel
Verwijder een bedrijfsmiddel
Claims Based - Lab
Voeg een nieuw persoon toe
Voeg een nieuwe werknemer toe
Voeg een nieuwe klant toe

Succes! De rule-engine heeft een positief besluit genomen.

Klik hier om de sessie te beeindigen

De volgende gegevens zijn bekend in deze sessie:
rol = com

Succes! De rule-engine heeft een positief besluit genomen.

Klik hier om de sessie te beeindigen

De volgende gegevens zijn bekend in deze sessie:
rol = com

Please fill in the form:
Persoon ID:
Voornaam:
Voorletters:
Tussenvoegsel:
Achternaam:

Figure 54 Screenshot of a business process implementation

The *Decorum* also keeps track of the information which is provided by the Subject to make access control decision. The decision whether or not an *Activity* is allowed to start is not made by the *Decorum*. Whether an activity is allowed to start depends on the decision made by the access control mechanism.

4.3.2 RULE ENGINE COMPONENT

Access control mechanisms make access control decisions. Either access is granted or access is denied. A positive decision is made when an activity is allowed to start and when a negative decision is made when an activity is not allowed to start. It could also be possible that the access control mechanism cannot make a decision. When no decision can be made an activity is not allowed to start.

The access control requirements of an activity are represented by the business rules. The business rules not only mandate what data is required in the activity, but also who can perform the activity. Which activities a user is authorized to perform can depend on the role that user has, but there can also be other factors why the user is or is not authorized to perform activities. Age for example can be such a factor. Depending on the rules set for an activity a subject may or may not be authorized to perform the activity. Even if a subject is authorized to perform an activity, the activity may still not be allowed to start because certain input constraints are not met. In the figure below activities B and C are not allowed to be started. A different role may allow different activities to be executed (e.g. activity A and be are allowed and activity C and D not).

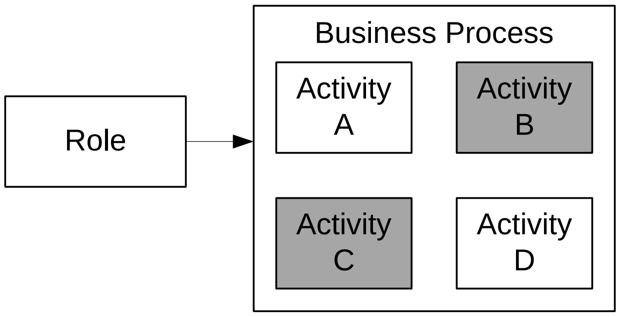


Figure 55 Example business process

The access control mechanism that enables access control based on rules is the Rule Based Access Control mechanism. In order for the rule engine to make an access control decision, it will need a set of rules and information about the terms occurring in the rules. The rules are supplied by a rule repository and information about the terms is provided (indirectly) by the subject.

4.3.3 RULE REPOSITORY COMPONENT

The rule repository is a database supplying the rules that apply to a certain activity. When this component receives for the rules of a certain *ProcessActivity* it sends back the rules which apply to the requested *ProcessActivity*. These rules are in a format which the rule engine can understand.

4.3.4 CLAIMS COMPONENT

In order for the rule engine to evaluate the rules it needs to know information about the terms occurring in the rules. The rule engine expects to receive this information from the *Subject* which requests access to a *ProcessActivity*. The mechanism the *Subject* uses to gather this information is via claims which the *Subject* can gather from an *Identity Provider*.

The figure below show how the claims are gathered by the *Subject*.

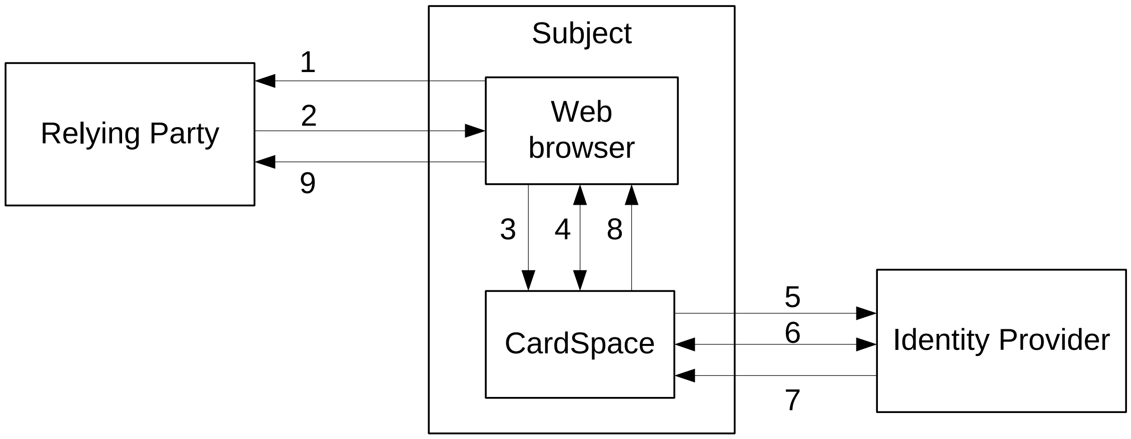


Figure 56 Claims Based Access Control mechanism

There are a total of nine steps in order to acquire the requested claims. It starts when (1) the *Subject* sends an access request to the *Relying Party* using a web browser. (2) The *Relying Party* sends a claims request back to the web browser of the *Subject*. This claims request contains a list of required claims and possibly a list with trusted *Identity Providers*. (3) The web browser sends the claims request to the *CardSpace* application. The *CardSpace* application selects the InfoCard which can be used to retrieve the requested claims. Each InfoCard represents an *Identity Provider*. (4) The list of InfoCards is shown to the *Subject* and the *Subject* selects one of the InfoCards. (5) *CardSpace* sends the claims request to the selected *Identity Provider*. (6) After the *Subject* has authenticated himself, (7) the *Identity Provider* sends a certificate containing the claims back to the *CardSpace* application, which (8) the *CardSpace* application passes on to the web browser. (9) Finally the web browser sends the requested claims to the *Relying Party*.

4.3.5 CLAIMS BASED WORKING ARCHITECTURE

In the Claims Based Working architecture the different components which were discussed above will be combined. The figure below shows which steps are taken when a *Subject* tries to execute an *Activity*.

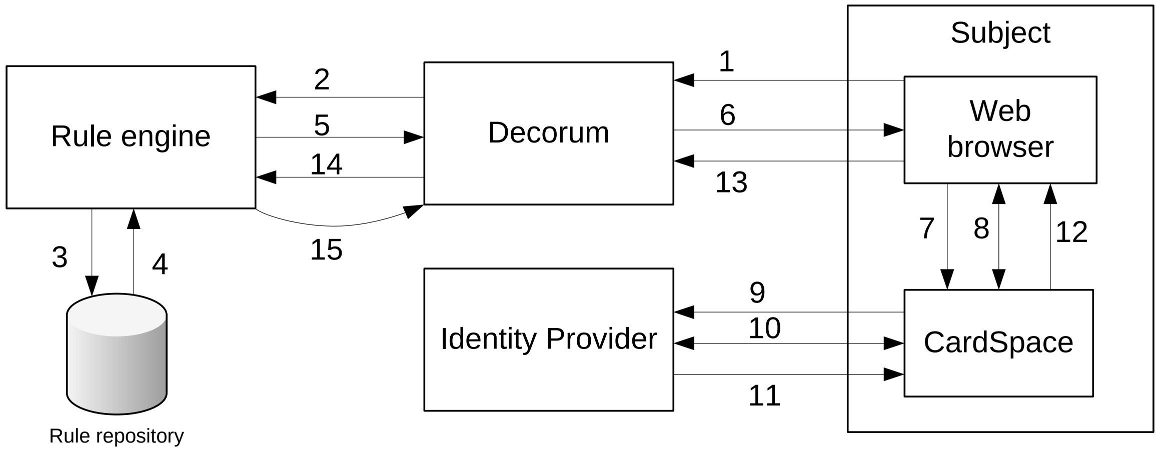


Figure 57 Claims Based Working components

The decision whether or not an activity is allowed to execute is made when the *Subject* tries to execute an activity. (1) Using a web browser the *Subject* requests access to a business activity. (2) The *Decorum* sends this request to the *Rule Engine*, which (3) in turn asks the *Rule Repository* to supply the rules which apply to the activity for which access is requested. The *Rule Repository* generates the rules in a format which the *Rule Engine* can understand and sends it to the *Rule Engine*. From these rules a list of terms is generated for which information is required in order to make an access control decision. (5) The *Rule Engine* sends the list with missing terms to the *Decorum* and the *Decorum* uses this list to generate a claims request. Steps 6 to 13 are taken to acquire the claims. (14) The *Decorum* sends the missing information about the terms to the *Rule Engine* together with the name of the activity for which access was requested by the *Subject*. The Rule Engine uses the information about the terms to evaluate the rules for the requested activity and (15) sends the decision back to the *Decorum*.

In order to demonstrate the practical use of the concepts of Claims Based Working we devised two scenarios. The first scenario is concerning a HRM process and the second one is the missing item scenario which is based on the purple crocodile scenario that was mentioned in the introduction of this thesis.

4.3.6 RESULTS

Having built the Claims Based Working demonstrator a number of remarks can be made about the various components used. The Decorum component is the part of the demonstrator that the users actually see and interact with. The idea behind declarative business processes is that it should give the user the freedom to perform activities as they see fit as long as the constraints are not violated.

In our implementation we represented all the activities of a business process as colored rectangles in a grid. An activity can only be performed if the color of the rectangle representing the activity is green. The activities which are orange or red cannot be performed.

When the number of activities of a business process is small, the user should be able to oversee all the activities. But when the number of activities of a business process is relatively large, so is the number of rectangles on the Decorum. With a high number of activities it might become hard for the user to find the activity that he or she wants to perform.

A solution would be to only show the activities which are allowed to be performed by the user. This will reduce the number of activities on the screen, but still if the number of allowed activities is high enough it would still be hard to find a particular activity. In the later implementations of the Decorum we incorporated a button which allows us to switch between the two views.

We have not discussed the order of representation of the activities. We have chosen to display the activities in a grid like manner, but we do not know if this is the best layout. We believe that the order of activities will also increase the user-friendliness of the Decorum, but this is a topic for another research as it involves interface design.

The rule engine component works in about the same way as in the previous demonstrator. When the rule engine receives an access control request, the rule engine retrieves the set of rules which apply to the business activity for which access is requested. Each activity in the business process has it own set of rules. In our implementation the rules which apply to an activity have be manually set for each activity. It was outside of our scope to build a rule management component which allows the rules for each activity to be managed in a more efficient way.

Another remark with respect to the business rules is that the business rules for each activity are all in a format that the rule engine can directly understand. This format of rules is not as easy to read as the rules written natural language like SBVR (OMG 2008). Building a component that is able to translate natural language business rules to business rules which can be understood by the rule engine was outside of the scope of this research.

In order to make an access control decision the rule engine needs rules and also information about the terms which occur in the rules. In our reference framework we use the Claims Based Access Control mechanism to gather this information. During the implementation of the demonstrator we came across certain inefficiencies while using the Claims Based Access Control mechanism.

When *Claims* need to be gathered, the subject needs to take a number of steps to get the claims from the Identity provider. The user needs to click on an icon on the screen which starts CardSpace. CardSpace displays a number of InfoCards which represents the different Identity Providers which are able to deliver the required Claims as specified by the Relying Party. The Subject has to choose an InfoCards and authenticate him or herself to the Identity Provider, before the Identity Provider delivers the requested Claims.

The Identity Provider does not instantly deliver the requested Claims. After the Subject has inputted the username and password it takes a couple of seconds before the Identity Provider delivers the requested Claims. A couple of seconds might not seem very long, but if Claims need to be gathered multiple times a couple of seconds might become annoying to the Subject. Adding to that is the fact that the Subject has to enter authentication data each time Claims need to be gathered.

For some pieces of information it might not be necessary to get the Claims from an Identity Provider. It may be sufficient that the Subject provides this information him or herself. This can be seen as self-issued Claims. This is in line with the definition of a Claim made by Cameron (2005). Another solution to minimize the number of times a user needs to gather claims is to store the gathered Claims in the session of the Subject. The information in this session can be used multiple times by the rule engine.

5 VALIDATION

The goal of this chapter is to test whether or not the concepts of Claims Based Working can work in practice. The concepts we want to test are related to the way Claims Based Working organizes business processes and how decision making information can be gathered using Claims. We will use the Claims Based Working demonstrator and two scenarios to validate Claims Based Working. In the first scenario we primarily focus on validating the declarativeness of Claims Based Working and in the second scenario we focus on gathering information to decide whether or not an activity is allowed to be performed.

The first scenario is a HRM scenario which involves a number of business activities surrounding employees and customers. We will first work out this scenario using the constraints based approach by Pesic and then we will work out this scenario using Claims Based Working. The two methods will be compared in the result section (5.1.3) of the HRM scenario.

The second scenario is concerns retrieving lost items, which is inspired by the purple crocodile scenario that was introduced at the beginning of this thesis. The users of the system in this scenario are the people who are trying to retrieve the item they have lost. When compared to the first scenario the user in the second scenario can be seen as foreign user and the users of the first scenario can be seen as native users. Decision making information about native users is like to be known to the party that has to make an access control decision, but decision making information about foreign users is less likely to be known. With this last scenario we attempt to show that this otherwise unknown information can be gathered using Claims.

5.1 HRM SCENARIO

The HRM process is the process which relates to the human resources in a company. Our scenario has two kinds of people which are related to the company. The first are the employees of the company and the second are the customers of the company. Company assets can be assigned to employees. An asset can be a mobile phone or a laptop for example. The employees are allowed to have multiple assets. When an employee leaves the company, the employee has to hand in all the assets which were assigned to him or her.

This particular company is very small and can only accommodate four employees at most. Customers are assigned to the orders they place. Customers are not allowed to directly input their orders in the system. The orders are placed by employees of the company on behalf of the customer.

The activities in the HRM process can be performed by a subset of all the employees of the company. Only employees with a certain role are allowed to perform the activities relevant to that role.

Since we have employees and customers we will define two roles. The HRM role is allowed to perform HRM related activities and the commercialist role is allowed to perform commercial activities.

Using this scenario we attempt to validate a number of aspects. One aspect is the expressiveness of the rules used by the rule engine to incorporate different access control mechanisms, like the All-or-Nobody Access Control mechanism and RBAC.

Another aspect is the declarativeness of Claims Based Working. Each activity is allowed to start as long as the constraints applying to the activity are met. The constraints we use do not mention which activity needs to be executed after another activity, but rather focus on the input requirements of an activity. We will compare the concepts of Claims Based Working with the constraint based approach of Pesic (2008). We chose this method, because it also uses constraints to implicitly create a flow through the business process.

5.1.1 CONSTRAINT BASED APPROACH

The approach of Pesic is used to model the scenario as described above. Only the parts which are relevant for this research will be used of the constraint based approach.

A number of activities can be identified from the scenario as described above. The activities we have identified are the following:

$A^{FT} = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}\}$ is the set of activities where:

a_1 = create a person
a_2 = select a person
a_3 = update a person
a_4 = delete a person
a_5 = create an employee
a_6 = select an employee
a_7 = update an employee
a_8 = delete an employee
a_9 = create a customer
a_{10} = select a customer
a_{11} = update a customer
a_{12} = delete a customer
a_{13} = assign a company asset to an employee
a_{14} = reclaim a company asset from an employee
a_{15} = create an order for a customer

The constraints used by Pesic are assigned to the events where the constraints apply to. We will simplify matters a bit by just naming the constraint below.

f_1 = Every a_3 must be preceded by an a_2 activity
f_2 = Every a_4 must be preceded by an a_2 activity
f_3 = Every a_5 must be preceded by an a_2 activity
f_3 = Every a_7 must be preceded by an a_6 activity
f_4 = Every a_8 must be preceded by an a_6 activity
f_5 = Every a_9 must be preceded by an a_2 activity
f_6 = Every a_{11} must be preceded by an a_{10} activity
f_7 = Every a_{12} must be preceded by an a_{10} activity
f_8 = Every a_{13} must be preceded by an a_6 activity
f_9 = Every a_{14} must be preceded by an a_6 activity
f_{10} = Every a_{15} must be preceded by an a_{10} activity

All of the above constraints can be modeled using a precedence constraint. This constraint is looks as follows.

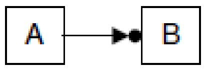


Figure 58 precedence constraint

The precedence constraint means that before activity B can be executed activity A has to be finished.

Conditional constraints also need to be added to the constraint model. The HRM scenario has a number of conditional constraints. Activity a_5 (create employee) may only be executed if the f_3 constraint is satisfied and if the total number of employee is less than five. Another example of an activity where a conditional constraint has to be added is that of activity a_8 (delete an employee). Activity a_8 may only be executed if constraint f_4 has been satisfied and the number of assets that belong where assigned to that employee must be zero.

The method by Pesic uses user maps and activity maps to authorize users to each activity. The user map assigns users to roles and groups and the activity map determines which roles and groups are authorized to perform each activity.

The data perspective is used to determine which data elements needs to be available for each activity, but it is not mentioned where this data must come from. We assume that this information has to locally available.

5.1.2 CONCEPTUAL MODEL CBW

A data model can be created from the scenario as described above. The HRM scenario revolves around people or persons. This person can be known as a person in the system but can also be an employee or a customer. Employees can have company assets assigned to them. Customers on the other hand do not have assets assigned to them, but the customers can be linked to the orders they place. Having this information the following data model can be created.

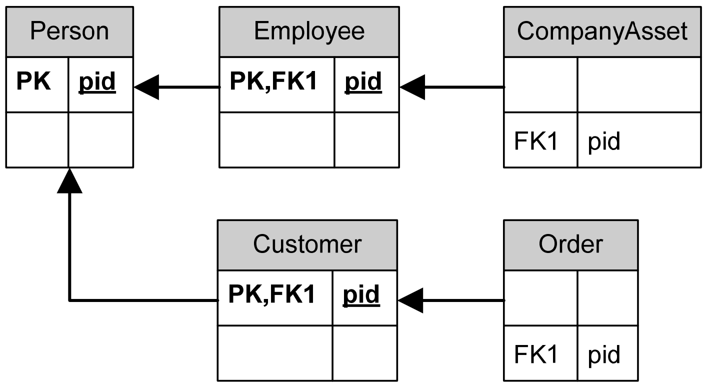


Figure 59 Database Diagram of the HRM scenario

Using this data model we can define a number of activities which support the HRM scenario. In this scenario we use roles to define who is authorized to perform which activity. Users with the HRM role are authorized to perform any employee related activity and users with a commercial role are authorized to perform all the customer related activities. The scenario also mentions other restrictions which will apply the different activities. The diagram below shows all the activities and the restriction which apply to them.

Activity	Authorized role	Restrictions
Create a person	Anybody	
Select a person	Anybody	
Update a person	Anybody	A PersonID must exist

		The Person must exist
Delete a person	Anybody	
Create an employee	HRM	A PersonID must exist The new employee must be a person The total number of employees may be at most 4
Select an employee	HRM	
Update an employee	HRM	A PersonID must exist The Person must be an Employee
Delete an employee	HRM	A PersonID must exist The Person must be an Employee The Employee must not have any Assets
Create a customer	Commercialist	A PersonID must exist The new customer must be a person
Select a customer	Commercialist	
Update a customer	Commercialist	A PersonID must exist The Person must be a Customer
Delete a customer	Commercialist	A PersonID must exist The Person must be a Customer The Customer must not have any Orders
Assign a company asset to an employee	HRM	A PersonID must exist The Person must be an Employee
Reclaim a company asset from an	HRM	A PersonID must exist

employee		The Person must be an Employee The Asset must be assigned to the Employee
Create an order for a customer	Commercialist	A PersonId must exist The Person must be a Customer

Table 5 Activity Authorizations and Constraints

5.1.3 RESULTS

Most activities have constraints on the role of the *Subjects* that are allowed to perform the activity. For these activities the RBAC mechanism is used in the form of rules. Some of activities did not put constraints on the *Subject* performing the activity. For these activities the All-or-Nobody Access Control mechanism is used in the form of rules. How these two mechanisms are implemented is discussed in section 4.2.1. With our approach it is possible to authorize users for each activity using a different access control mechanism. This allows for a more dynamic assignment of users to activities than can be achieved when using user maps and activity maps.

A number of activities also have data constraints. Both the data constraints as the *Subject* constraints have to be met before an activity is allowed to start. In this scenario only the decision making information about the *Subject* constraints are gathered using claims. The decision making information about the data constraints are gathered using the output of activities. For example the “create employee” activity is only allowed to start if information is available about the person who is about to become an employee. This information is gathered using the “select person” activity.

The activities used in this scenario are not explicitly linked to each other. In the CBW approach some activities depend on the output of other activities to meet the constraints that apply to that activity. The constraint based approach uses constraints which explicitly state which activity has to be performed before another activity.

An example is given above about the “create employee” activity. In the case of the CBW approach the “select a person” activity delivers the required output in order for the “create employee” activity to start. But if there is a “select a person 2” activity which output meets the data requirements of the “create employee” activity, then the “select person 2” activity can be used instead of the “select person” activity without changing the constraints of the “create employee” activity. In the case of the constraint based approach the constraints which apply to the “create employee” activity has to be changed to also include the constraints concerning the added activity.

An activity can also be removed from the process, though this has a number of issues for both the CBW approach as the constraint based approach. For the CBW approach the removal of an activity can be an issue due to the fact that the constraints for an activity are based on data requirements of the activity, it is hard to predict which activities are dependent on the output of the removed activity. If there are no activities in the process which can deliver the same output as the removed activity the business process might be unable to finish.

For the constraint based approach the removal of an activity is less of an issue, because the constraint model explicitly state how activities depend on each other. When an activity is about to be removed it can be checked whether or not the activity can be removed without rendering the entire process useless.

There are also other cases where the process is unable to finish. In our scenario we have a constraint that states that the company can have at most four employees. The constraint based approach is able to notify the user before he or she starts the process if the total number of users has been reached. This is not the case with the CBW approach, due to the fact that the constraints used do not directly refer to activities. When a user wants to add another employee by performing a series of activities, then the user will not be notified beforehand if the maximum number of employees has been reached. This will result in the user performing a number of activities but be able to finish the process when there are already four employees present in the system.

Even if the maximum number of employees is not reached beforehand, this can change during runtime. When user A is busy with the process of adding a new employee, user B could have added an employee just before user A was able to, thus resulting in user A unable to finish the process. This is an issue for both approaches. A solution would be to lock certain activities during runtime, but this may hinder other users in their execution of the process.

The data constraints on the activities make the business process declarative by allowing different sequences of activities which result in the same end product. Using this scenario we validated that it is possible to implicitly create a flow through a process without explicitly stating which activities need to be performed in what order, but the flow only becomes apparent during the execution of the process.

5.2 MISSING WALLET SCENARIO

The scenario used here is based on the purple crocodile scenario introduced in section 1. The purple crocodile scenario is about a little girl who has lost her inflatable purple crocodile. Instead of an inflatable purple crocodile we will use wallets.

Wallets often contain a number of cards with information on them which are unique for the owner of the wallet. The cards which are in a wallet can range from ID cards to VIP customer cards. All of these cards are issued by certain companies and it is almost impossible for the owner of the wallet to name each specific piece of information on every card in his or her wallet, but the companies that issued the cards often do have this information. Using a request for *Claims* this information can be retrieved from the original source, which is the company that issued the card. By gathering information about a number of cards in the lost wallet the owner should be able to provide enough evidence that the wallet belongs to him.

This scenario demonstrates an important aspect Claims Based Working, namely gathering Claims. The Claims are used to gather information which is necessary to make an access control decision, but which is unknown by the party that has to make the decision and which is possibly also unknown to the party that requested access.

The constraint based approach of Pesic a data perspective is mentioned in which show which information is available for which activity, but it is unclear where this information actually comes from. We assume that this information has to be available locally. For this reason we will not use the constraint based approach in this scenario and only use the Claims Based Working approach.

5.2.1 CONCEPTUAL MODEL

The workflow of the process as described above can be modelled into a traditional workflow diagram, which is depicted below.

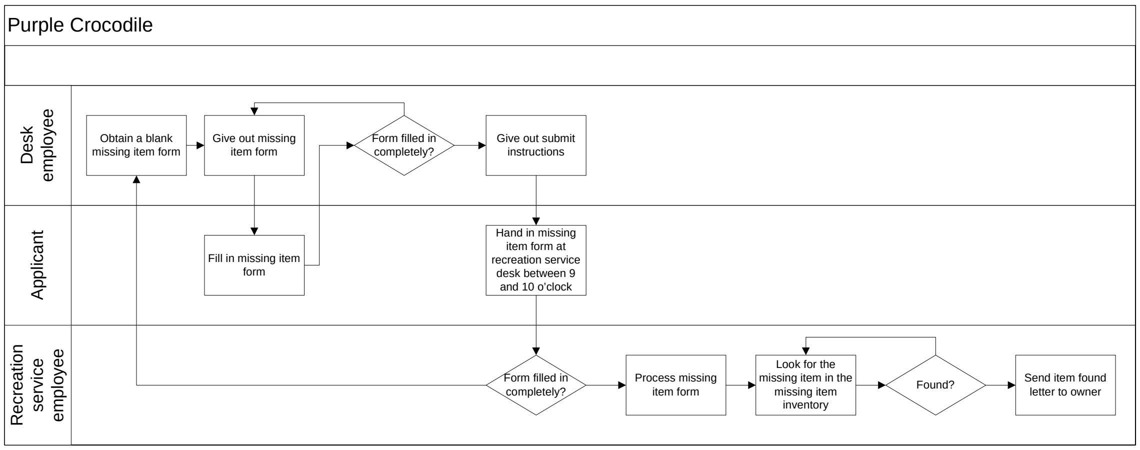

Purple Crocodile
Desk employee
Applicant
Recreation service employee
Obtain a blank missing item form
Give out missing item form
Form filled in completely?
Give out submit instructions
Fill in missing item form
Hand in missing item form at recreation service desk between 9 and 10 o'clock
Form filled in completely?
Process missing item form
Look for the missing item in the missing item inventory
Found?
Send item found letter to owner

Figure 60 Traditional Workflow of the Purple Crocodile scenario

The described scenario requires certain steps to be taken which are unnecessary in this specific case. The remainder of this section models the “Missing Wallet” process in a declarative manner.

The most important activity in this scenario is the activity in which the person that has lost something actually gets the lost item back. There is a constraint which has to be met before the lost item can be handed back to its rightful owner. The rightful ownership of a lost item has to be proven before a lost item can be handed back. The rightful ownership can be proven by letting the potential owner describe all the characteristics of the lost item.

Proving the rightful ownership of a lost item differs with each type of item. For valuable items like a wallet it is far more important to prove the rightful ownership than for a purple crocodile. For wallets the characteristics can be described by naming the content of the cards which are in the wallet. A wallet is likely to contain cards like an ID-card, a drivinglicense, a bank card, a credit card, a VIP card, a library card, and an insurance card. Each of these cards often contains information which is specific for that card.

Card type	Type of information
ID-card	Name
	Date of birth
	Sex
	Burger Service Number (BSN)
	Card number

	Expiration date
driving license	Name
	BSN
	Expiration date
	driving license number
Bank card	Name
	Bank account
	Card number
	Expiration date
Credit card	Name
	Credit card number
	Expiration Date
VIP card	Card number
Library card	Name
	Card number
	Expiration date
Insurance card	Name
	Insurance number
	Date of birth

	Sex
	BSN

Table 6 InfoCard Claimtypes

The table above shows the different cards and what information those cards can contain. Someone who has lost a wallet might not be able to name every piece of information on each card. Most people do not know their BSN let alone the card number of an ID card. Using claims a potential owner is still able to retrieve this information from its source.

We assume that the party that has issued the physical card also has the information which in on the physical card in digital form. Using a claims request this information can be retrieved even if the subject does not know this information by heart. The subject just has to authenticate itself to the claims issuer in order for the claims to be retrieved.

If someone who lost an item cannot proof rightful ownership or cannot find the lost item, a missing item registration form can be submitted. This allows an employee to look for the missing item.

When the rightful ownership cannot be proven, it can mean a number of things. It can mean that the rightful ownership constraints are not met or that the missing item is either not registered or incorrectly registered in the system.

This missing item process consists of a number of activities of which the most important ones are the following three. The first activity is an activity in which a potential owner can prove rightful ownership of a lost wallet. The second activity is an activity which allows a person who lost something to register the lost item. The third activity is the activity that returns the lost object to its rightful owner.

The “Register lost item” activity is only allowed to be performed if the potential owner was unable to prove rightful ownership of a lost item. The “Return lost item” activity is only allowed to be performed if the rightful ownership of a lost item has been proven.

Everybody is allowed to perform the “Prove rightful ownership activity”. The output of this activity is used as decision making information to determine whether or not the activities “Register lost item” and “Return lost item” are allowed to start.

The criteria which we have defined as proof of rightful ownership are the following:

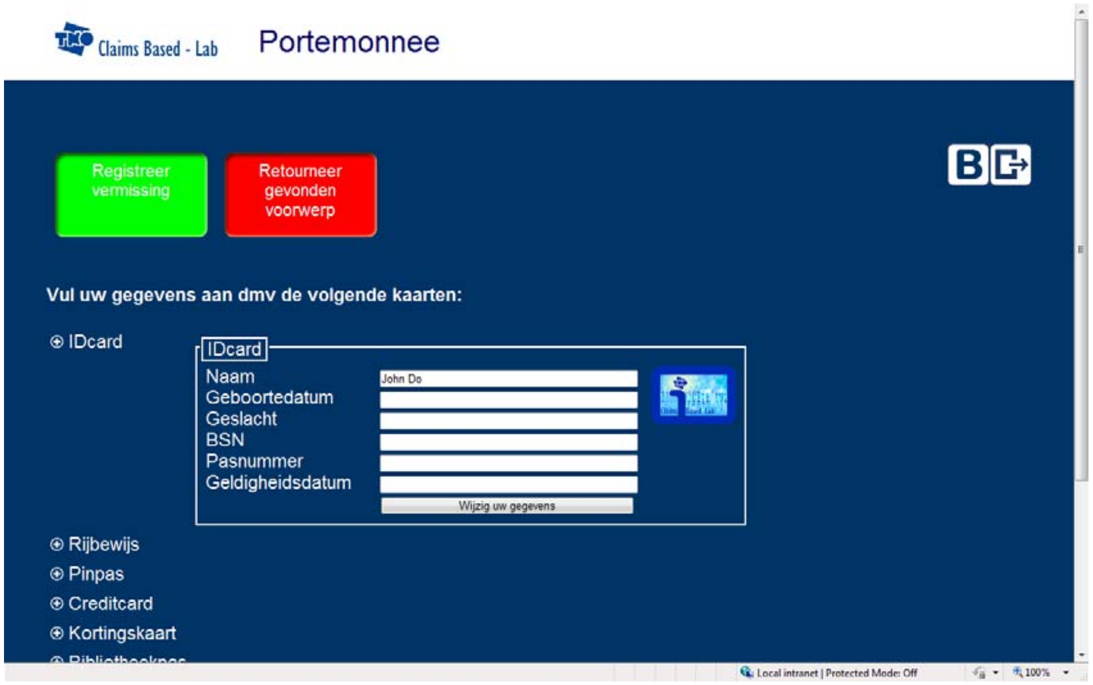
Constraint 1: The subject has provided the correct information of at least two cards which are in a wallet.
Constraint 2: The subject has provided the correct information of all the cards which are in a wallet.
Constraint 3: Constraint 1 is true or Constraint 2 is true.
Constraint 4: Exactly one wallet has been found and Constraint 3 is true.

Constraint 1 and constraint 2 will have the same result if there are two or more cards in the wallet and all the provided information is correct, but if there is only one card in a wallet constraint 1 is always false and constraint 2 can be either true or false. When some information provided differs from the information on the cards which are in the wallet constraint 2 will always be false, because not all the provided information is correct, but if the correct information is provided about two cards constraint 1 will be true.

It could be that some cards are not unique which can mean that some cards in different wallets are the same. The first part of constraint 4 makes sure that the provided information belongs to one unique wallet.

The two figures below show the “Prove rightful ownership” activity. Figure 61 shows a screenshot of this activity when the rightful ownership has not been proven. The subject only provided his name “John Do” of the ID card and there are no lost wallets which have only one card in it which also matches the information the subject provides.

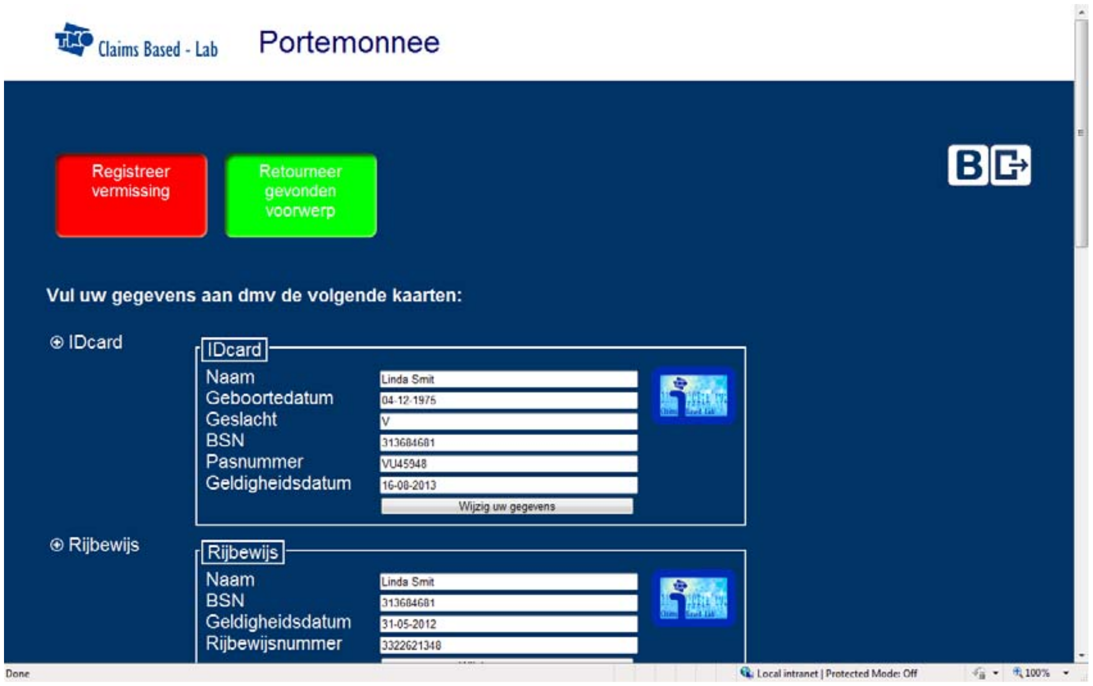
Figure 62 is a screenshot of the activity when the rightful ownership has been proven. The subject provided all the information of all the cards which were in her wallet and apparently there was only one wallet containing those cards with the provided information.



IDcard

Naam	John Do
Geboortedatum	
Geslacht	
BSN	
Pasnummer	
Geldigheidsdatum	

Figure 61 Rightful ownership not proven



Vul uw gegevens aan dmv de volgende kaarten:

⊕ IDcard

IDcard

Naam	Linda Smit
Geboortedatum	04-12-1975
Geslacht	V
BSN	313684681
Pasnummer	VU45948
Geldigheidsdatum	16-08-2013

⊕ Rijbewijs

Rijbewijs

Naam	Linda Smit
BSN	313684681
Geldigheidsdatum	31-05-2012
Rijbewijsnummer	3322621348

Figure 62 Rightful ownership proven

5.2.2 RESULTS

In the HRM scenario whether or not a Subject was authorized to perform an activity was based on the role of the *Subject*. In this scenario everybody is allowed to perform every activity as long as the constraints which apply to an activity are met. The "return missing item" activity for example is only allowed to be performed if the there is enough proof of ownership of a missing wallet.

This proof is delivered by providing information about the contents of a wallet. Some of this information can not be provided by the user himself, because he does not know this information. Still this information can be gathered using Claims provided by a third party. This is an important aspect of Claims Based Working as we explicitly provide a mechanism to gather decision making information, whilst other frameworks just state what information is required in an activity and assume that this information is available. With this scenario we are able to show that decision making information can be gathered using Claims, which were otherwise impossible to gather.

But there are some issues with gathering information using Claims. The Claims gathering mechanism uses InfoCards which have to be installed beforehand. In our scenario this can be a problem if for example a wallet contains a bankcard of a bank for which no InfoCard is installed. The Subject is then unable to select the InfoCard of specific bank and thus is unable to get the Claims. This is a problem considering that the Subject is unable to get the necessary information, because the party that delivers this information is unknown by the decision making party. On the other hand the decision making party needs to trust the information coming from a third party. In order for the decision making party to trust the information coming from that third party the decision making party needs to at least know that third party.

6 CONCLUSIONS

In this chapter the answers to the research questions will be presented first followed by the conclusions that can be drawn from this research. Furthermore the limitations of this research will be discussed. Possible directions for future research will be discussed at the end of this chapter.

In order to answer the main research question we will first answer our sub-questions.

WHAT DOES AN ACCESS CONTROL META-MODEL LOOK LIKE?

A literature research was performed to find the similarities of different access control mechanisms. Using the findings we have formulated a meta-model which was presented in chapter 3.1. The differences between the various access control mechanisms were in the way *Subjects* were authorized to the *Objects*. The DAC mechanism authorizes Subjects directly. The MAC mechanism authorizes *Subjects* based on their classification and RBAC does this based on the role the *Subjects* have.

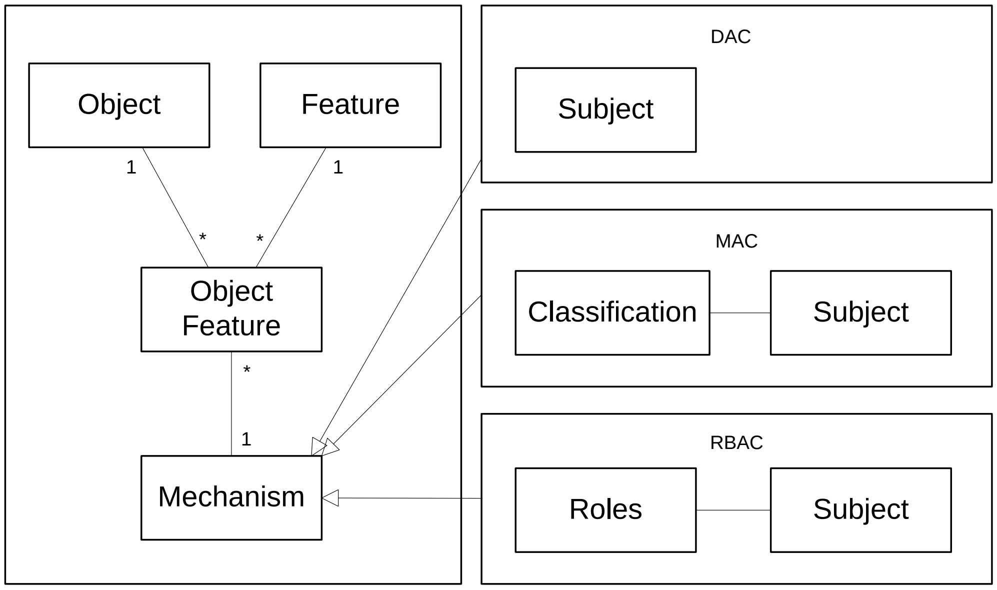


Figure 63 The Access Control Meta-model

In the Access Control Meta-model, depicted above, we have separated the *Subject* from the *Objects* and their *Features*. The model allows for different access control mechanism to authorize a *Subject* to an *Object Feature*.

The Access Control Meta-model in itself is an access control mechanism that enforces access control using a number of different mechanisms. Different access control mechanisms can be used when appropriate. The rule-engine we used in our prototype is able to apply the appropriate Access Control mechanism to different *Object Features*. The implementation of the Rule Based Access Control mechanism for the demonstrator incorporates the Access Control Meta-model. It can incorporate different access control mechanism by using the appropriate rules.

HOW CAN THE PREVIOUSLY DEFINED AC META-MODEL BE INCORPORATED IN DECLARATIVE BUSINESS PROCESSES?

Research found that the business activities in declarative business processes have constraints which determine whether or not the activity is allowed to be executed. The access control meta-model can be applied to enforce access control on the business activities, by only allowing the business activities to be executed if the constraints for that activity are satisfied.

The constraints of the business activities are represented by business rules. The access control mechanism which is used is the Rule Based Access Control mechanism. With this access control mechanism it is possible to implement other access control mechanisms like Discretionary Access Control (DAC), Mandatory Access Control (MAC) and Role Based Access Control (RBAC), by setting the appropriate rules.

HOW CAN BUSINESS PROCESSES BE SPECIFIED IN SUCH A WAY THAT ACCESS CONTROL PRINCIPLES CAN BE APPLIED TO BUSINESS ACTIVITIES (BAAC)?

Using the Bill-of-Materials (BOM) approach the products and constraints of a business process can be identified. The products are the output of business activities and the constraints are used to apply access control on the activities.

The resource perspective represents which entities are authorized to perform certain business activities. These entity authorizations are enforced by the access control mechanism which applies to a business activity.

HOW CAN THE CONCEPTS OF CLAIMS BASED ACCESS CONTROL BE INTEGRATED WITH BAAC TO FORM A REFERENCE ARCHITECTURE FOR CLAIMS-BASED WORKING?

The literature about the mechanism behind Claims Based Access Control describes a mechanism which allows the Subject which request access to deliver the necessary information to the party that enforces access control, for it to make an access control decision.

Information about the terms that occur in business rules can be gathered using the CBAC mechanism. When a *Subject* wants to execute a business activity, the relevant business rules are gathered and a request is sent to the *Subject* to deliver information about the terms which are required to make an access control decision.

HOW TO IMPLEMENT / VALIDATE CBW?

A demonstrator was made as proof of concept. The previous chapter describes a number of demonstrators which were made during this research. A number of scenarios were used to implement the Claims Based Working demonstrator. The first scenario was a HRM scenario which showed how users with different roles are allowed to perform different activities. The demonstrator also illustrated the data dependencies of the business activities. The data required for one activity to start can be delivered by another activity, thus implicitly creating a flow through the process.

Our method of organizing business processes state what needs to be done and not how this should be done. Which activities need to be performed and in which order is not explicitly states this allows activities to be interchanged with another as long as the input requirements of the subsequent activity are met. This in contrast with the method as described by Pesic (2008) which explicitly states the relation of one activity with another. Whether or not certain activities are allowed to be performed with the method described by Pesic is event based and in Claims Based Working this is data based.

The second scenario was the missing wallet scenario, which had a business process which enabled people that lost something to find the lost item using Claims. This scenario is meant to show how information can be gathered which is otherwise unavailable, because neither the *Subject* nor the *Relying Party* knows this information.

Using claims we are able to get information that otherwise are unavailable. The missing wallet scenario demonstrates how information which is otherwise unavailable can be retrieved using claims.

The main research question was:

WHAT DOES A REFERENCE ARCHITECTURE FOR CLAIMS-BASED WORKING LOOK LIKE?

This research question was answered in chapter 3.5, which describes the CBW meta-model and an architecture which supports CBW. The figures below show the CBW meta-model and the CBW architecture.

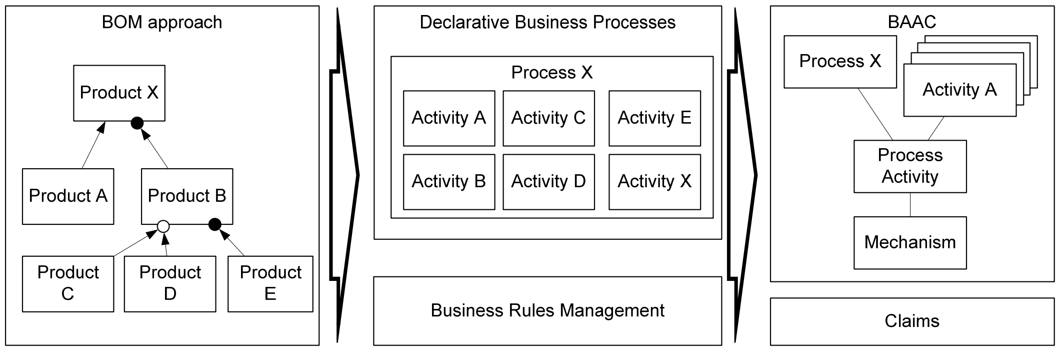


Figure 64 Claims Based Working Meta-model

The Claims Based Working meta-model shows how the business processes can be organized to support Claims Based Working. The BOM approach is used to identify the activities and their data dependencies. This approach can be used to get a Bill-of-Materials of either tangible products or intangible products. The BOM of intangible products is harder to define, because it is harder to visualize than tangible products. It is easier to come up with a BOM of a car than to come up with the BOM of health care.

The BOM approach identifies the different sub-products that activities produce, but the BOM approach does not state how the products are produced. It only states what must go in an activity and what must come out of an activity, but not what happens in between.

The data constraints and the subject constraints are in the form of rules. These rules in conjunction with claims are used by BAAC to enforce access control on the activities.

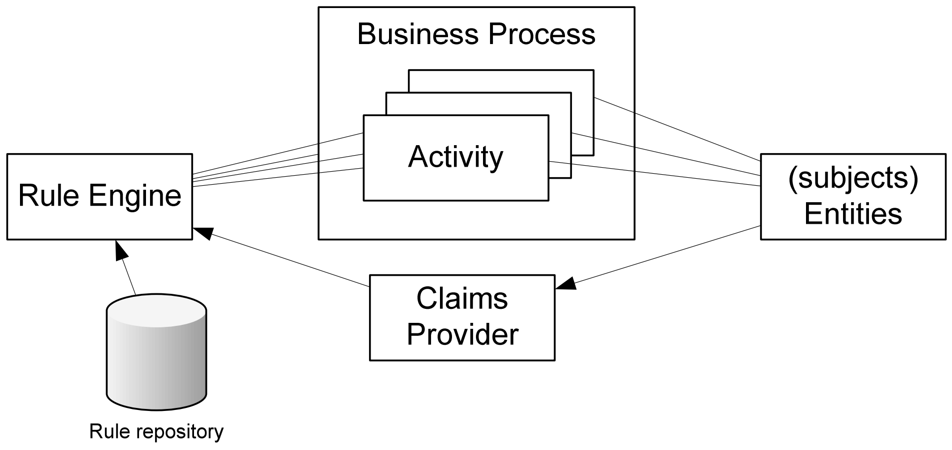


Figure 65 Claims Based Working architecture

The Claims Based Working architecture is built in a modular fashion. The architecture has three basic components, which are the rule engine, the rule repository, the decorum (business processes), and the claims provider. This makes it possible to change the components. The mechanism to gather information for the rule engine to make an access control decision is based on the Identity Metasystem. The claims gathering component uses a custom claims provider, which is able to deliver claims which do not directly concern the *Subjects*, but the claims are used to make an access control decision. The claims provider in this demonstrator uses a static database to return the claims which belong to a *Subject*. In future implementations this information needs to be generated dynamically, but due to time constraints this was outside of the scope of this research.

6.1 CONTRIBUTIONS

Our contributions to the existing body of knowledge are in two related areas. The first is in the area the way business processes are implemented and the second is in the area of information gathering.

Claims Based working provides an alternate method of implementing business processes. In our approach the constraints of one activity do not refer directly to another activity. Instead the constraints can refer to the output activities produce. To determine whether an activity is allowed to be executed only the constraints which apply to that activity have to be evaluated. In the constraints based approach of Pesic all the rules in the process have to be checked each time something happens in the process.

In our approach adapting certain activities will not affect the constraint of the other activities. To illustrate what we mean consider the following example:
Business process X consists of the business activities A and B. The output of activity A is process data X and the output of activity B is process data Y. Activity B has the constraint “process data X must be available”. At the moment activity B is only allowed to be executed if activity A has been performed, because activity A produces process data X. When an activity A’ is added to the process which also produces process data X, nothing happens to the constraints of the activities which were already in the process. Activity B is still allowed to be executed after activity A, but now activity B is also allowed to be executed after activity A’ has been executed.

Adapting individual activities will not have an impact on the constraints of the other activities in the business process. In other approaches where the constraints of an activity refers to other activities adapting a single activity may have impact on the constraints of other activities.

Claims Based Working also provides a mechanism to gather information that required in the business process. Not all required information might be available locally. Using Claims allows us to get that information from external sources. There are a number of reasons why information might not be available locally. It might for example be the case that certain information is not allowed to be stored locally due to certain laws and regulations. Another example is that there is a high variation in the required information and instead of storing all the variation the required information can be retrieved from its source.

The added value of Claim Based Working is that is provides an alternative view on organizing business processes. Claims Based Working enables business processes to be more dynamic. Business processes are easier to adapt, because the constraints for each business activity does not have explicit relations with other activities. Business activities can be added or removed without disrupting the constraints of the other activities.

The declarativeness of Claims Based Working gives the users of the business process more freedom to work according their own insight. The constraints of the activities set the boundaries in which the users can operate.

Using Claims has a number of benefits. The first is that the user of the system can partly decide where the Claims are gathered from. A second benefit is that the gathered Claims are authenticated. The system can assume that the Claims are truthful. A third benefit is that information can be gathered with is unknown by the system and by the user. If the type of Claim is known and the Identity Provider is able to deliver that type of Claim, information can be gathered which was otherwise unavailable to the user and the system.

The benefits of Claims Based Working are promising, but there are a number of limitations to the current state of our approach. These limitations and directions for future research are discussed in the sections below.

6.2 LIMITATIONS

In this research “made up” scenarios were used to evaluate our approach, which makes it difficult to make any fundamental conclusions about the performance and real life usability of Claims Based Working. It is for example not possible to say anything about the user acceptance. Would the actual users like the additional freedom or would they find our solution even more complicated? Testing Claims Based Working in practice was outside the scope of this research.

The scenarios used to evaluate Claims Based Working are fairly straightforward with a relatively small number of activities. We did not provide any solutions to ensure that the process is able to finish. With a high number of activities deadlocks might occur, preventing the process from finishing.

In this research we have focused on the preconditions of business activities and have used various mechanisms to regulate access to business activities. In our research we did not use constraints to ensure that an activity actually produces the intended output. Formulating these post-conditions was outside of the scope of this research.

The reference framework for Claims Based Working currently does not provide anything about how activities need to be performed. The framework mentions what needs to go in an activity and what comes out of an activity. How the input of an activity is transformed to the output of an activity is not mentioned. Due to this fact we cannot make any guarantees that the business process is able to finish. An activity might be started but never ended.

The current mechanism used to gather claims can only gather claims from one source at a time. When different sets of Claims need to gathered the user needs to perform on operation for each set to gather the Claims.

6.3 FUTURE WORK

This thesis combined a number of different concepts, like business processes, business rules, access control, and claims. The basics of CBW were explored and due to the time constraint a number of aspects need further research.

One of these aspects is rule management. Business rules are present in different layers of an organization. The top management can set the boundaries in which the layer below must operate. The lower layers can set their own boundaries within the broader boundaries set by higher layers. It would be interesting to see how these business rules can be gathered to ensure that the business processes and activities operate within the set boundaries.

Additional research is required about the way how business activities are organized. As mentioned before we did not make any statements about how business activities produce their output. Business rules might be required to ensure that activities produce the correct output. The access control enforced on business activities can be seen as the pre-conditions of an activity and the guarantees about the output of an activity might be seen as the post conditions of the activity. More research in needed to explore the importance of setting post conditions on activities.

We used a Microsoft solution to gather claims in order to make access control decisions. The downside of the Microsoft solution is that the InfoCards need to be downloaded beforehand and these InfoCards are specific for user that downloaded the InfoCard. We made a workaround so it would work in our case. Other mechanisms probably exist which do not require the InfoCard mechanism as used by the Microsoft solution. Additional research is required to explore other mechanisms that are able to gather claims.

REFERENCES

Aalst, W. v. d. (1999). "On the automatic generation of workflow processes based on product structures." Computers in Industry **39**: 97-111.

Aalst, W. v. d., M. Pesic, et al. (2009). "Declarative Workflows: Balancing between flexibility and support." Computer Science - Research and Development **23**(2): 99-113.

Bell, D. E. and L. J. LaPadula (1973). Secure Computer Systems: Mathematical Foundations. Techinical Report ESD-RE-278. Springfield, National Technical Information Service. **1**. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=AD770768&Location=U2&doc=GetTRDoc.pdf

Bertocci, V. (2008). "Claims and Identity: On-Premise and Cloud Solutions." The Architecture Journal **16**: 26-32.

Biba, K. J. (1977). Integrity Considerations for Secure Computer Systems. Technical Report TR-3153, NTIS. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA039324&Location=U2&doc=GetTRDoc.pdf

BRG (2000). Defining Business Rules - What Are They Really? Final Report rev. 1.3, The Business Rules Group. URL: http://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf

Cameron, K. (2005). The Laws of Identity. Web Services Technical Articles, Microsoft Corporation.

Davenport, T. H. (1993). Process Innovation - Reengineering Work through Information Technology, Harvard Business School Press.

Dongen, B.F. v., Aalst, W .v.d. (2005) "A Meta Model for Process Mingin Data" Proceedings of the CAiSE'05 Workshops (EMOI-INTEROP Workshop), volume 2, pages 309-320. FEUP, Porto, Portugal.

Halle, B. v. (1997). "How Smart Is Your Enterprise." Enterprises Database Programming & Design. URL: http://www.dbpd.com/vault/9608arch.htm

Hammer, M. and J. Champy (1993). Re-engineering the Corporation: A Manifesto for Business Revolution. New York, Harper Business.

McGregor, D. (1960). The Human Side of Enterprise, McGraw-Hill.

Microsoft (2005). "Microsoft's Vision for an Identity Metasystem." Web Services Technical Articles.

NIST (1993). Glossary of Computer Security Terms V.1, NIST.

Oasis (2007). WS-Trust 1.3, Oasis.

OMG (2008). Semantics of Business Vocabulary and Business Rules (SBVR), v1.0, OMG**:** 434.

Pesic, M. (2008). Constraint-Based Workflow Management Systems: Shifting Control to Users, Technical University of Eindhoven.

Pesic, M. and W. v. d. Aalst (2006). "A Declarative Approach for Flexible Business Process Management." Lecture Notes in Computer Science **4103**: 169-180.

Poppendieck, M. and T. Poppendieck (2006). Implementing Lean Softare Development: From Concept to Cash, Addison-Wesley.

Ross, R. G. (2005). Business Rule Concepts, Business Rule Solutions (LLC).

Ross, R. G. (2010). "My Story: To Play the Game You Need Rules." 2010, from http://www.ronross.info/story.php.

Samarati, P. and S. de Capitani (2001). "Access Control: Policies, Models, and Mechanisms." Lecture Notes in Computer Science **2171**: 137-196.

Sandhu, R. S. and E. J. Coyne (1996). "Role-Based Access Control Models." IEEE Computer **29**(2): 38-47.

Sandhu, R. S. and P. Samarati (1994). "Access Control: Principles and Practice." IEEE Communications Magazine **32**(9): 40-49.

Steinke, G. and C. Nickolette (2003). "Business rules as the basis of an organization's information systems." Industial Management & Data Systems **103**(1): 52-63.

TNO. (2010). "Mission and Strategy."   Retrieved 05-04, 2010, from http://www.tno.nl/content.cfm?context=overtno&content=overtno&item_id=30&Taal=2

Wagner, G. (2005). Rule Modeling and Markup. Reasoning Web. N. Eisinger and J. Maluszynski, Springer Verlag Berlin / Heidelberg. **3564:** 251-274.

Weiden, M., L. Hermans, et al. (2008). "Classification and Representation of Business Rules."