Project Work

Submitted by

Tobias Berger
Matr.No. 31471
Information and Media Technology
tobias.berger@tuhh.de

Supervised by

Hamburg University of Technology
STS – Institute for Software Systems
Prof. Dr. Joachim W. Schmidt
Sebastian Bossung

Vattenfall Europe Information Services
Head of E-Business Solutions
Thomas Schlüter

July 2006

Meeting Future Customer and Quality Requirements with a Multi-sited Content Management System

Affirmation

I affirm that:

This work has been prepared by me, all literal or content based quotations are clearly pointed out, and no other sources or aids than the declared ones have been used.

Hamburg, 06 July 2006

Tobias Berger

Abstract

This study discusses the optimal strategy for time efficient processes in content management systems that are involved in international collaboration.

The study covers an analysis of time efficiency in distributed systems, as well a more specific analysis for content management systems. On the basis of a general content management system model some general rules are developed for the optimal time efficient implementation of certain processes.

Finally, by means of the analysis of an example process taken from the CMS at Vattenfall, the consequences of the model in a real application are shown.

Table of Contents

1. Motivation

In the following the European energy supplier Vattenfall will provide the motivation for a problem that arises in most internationally acting companies.

1.1 Brand Communication in International Collaboration at the Vattenfall Group

1.1.1 The Vattenfall Group

When the German markets for electricity and gas became deregulated in the 1998 the local energy suppliers faced new challenges. Most parts of the supply chain from the generation to the end consumer were opened to competition on the free market, and with them also the companies that were active in those fields.

The Vattenfall Group is one of the biggest energy suppliers in the German and even in the European market. Started as a Swedish state-owned company, Vattenfall grew by acquiring local energy companies and their facilities all over Scandinavia, Germany, Poland and the Netherlands. After the deregulation in Germany they acquired the Hamburger Electrizitätswerke AG HEW and the Berliner Elektrizitäts und Wärme AG Bewag.

1.1.2 Synergies and Common Goals

For a horizontal merger or acquisition, where companies of same businesses are aggregated into a collective one, the main drivers are the possible synergy effects. Different parts of the company, preferably those parts that build up the supporting activities – meaning those functions that only support the primary value creating activities, e.g. administration or IT – will be aggregated and joint in a new and more efficient way. Those departments, formerly serving only the local company, will find themselves in an international environment, collaborating with departments of former local companies and competing with rival companies from all over the world. In the case of Vattenfall this is true for most of the departments of supporting activities. Here the aspect of international collaboration created the biggest impact on the restructuring of hierarchies, processes and communication.

The aim in the context of synergy effects is that all departments strive for the same common goals under the umbrella of Vattenfall. The umbrella is communicated to the outer world, but also internally by the introduction of a common corporate brand and image. The aim is to create “one” Vattenfall that is recognized by existing and potential customers all over Europe. In order to face the challenges of competition all over Europe, it is essential for Vattenfall to be perceived as a strong brand – in a consistent way and conforming to the vision and mission of Vattenfall. Therefore all local departments, which had their hands on marketing activities, have now to be coordinated to support the Vattenfall brand in a conjoint approach. This situation demands for a high degree of international collaboration as an essential element for the success of the whole company.

1.1.3 Working on the Brand Image

An important feature of Web presences is that that they are world wide reachable and offer an international platform for the communication of brand and image – also across borders and in areas where a company is not represented physically. This underlines the need for an international collaborative approach.

Therefore the main tool for communicating the brand image and getting in contact with the customer is today more then ever before the Web. Because most companies, including Vattenfall, have discovered the potential of the possibilities that the Web offers them, efforts in these fields are pushed intensively. Thus Vattenfall has initiated several relaunches of websites all over Europe, now using a content management system as basis for the success on the Web.

Before the start of the umbrella strategy, all the local brands were communicated separately. In Germany there was already the content management system in use that is now the basis for the international relaunches. Therefore a lot of know-how concerning the use of the CMS aggregated is already there.

However the usage of the content management systems changed. Due to the increased involvement of international collaboration editors and authors work on the same content elements conjointly – in parallel, from different sites.

The following paragraph will describe such a scenario.

1.1.4 Scenario: Working on a Content Object in International Collaboration

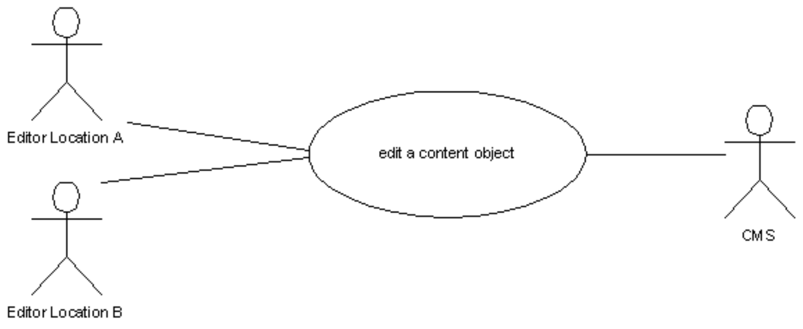


Figure 1 - Use case: edit a content object

The scenario of working on a page in international collaboration involves the use case “edit a content object” and 2 editors and the CMS as actors. It shows a process that happens often in day-to-day business. As responsibilities are distributed over the different parts of the Vattenfall group and international teams are in charge of the different websites, authors and editors from different countries will frequently join to correct mistakes, to discuss about page design or to exchange ideas about new content. A recurring use case in this scenario is the editing of a content object from different locations.

The following activity diagram illustrates the collaboration of the actors.

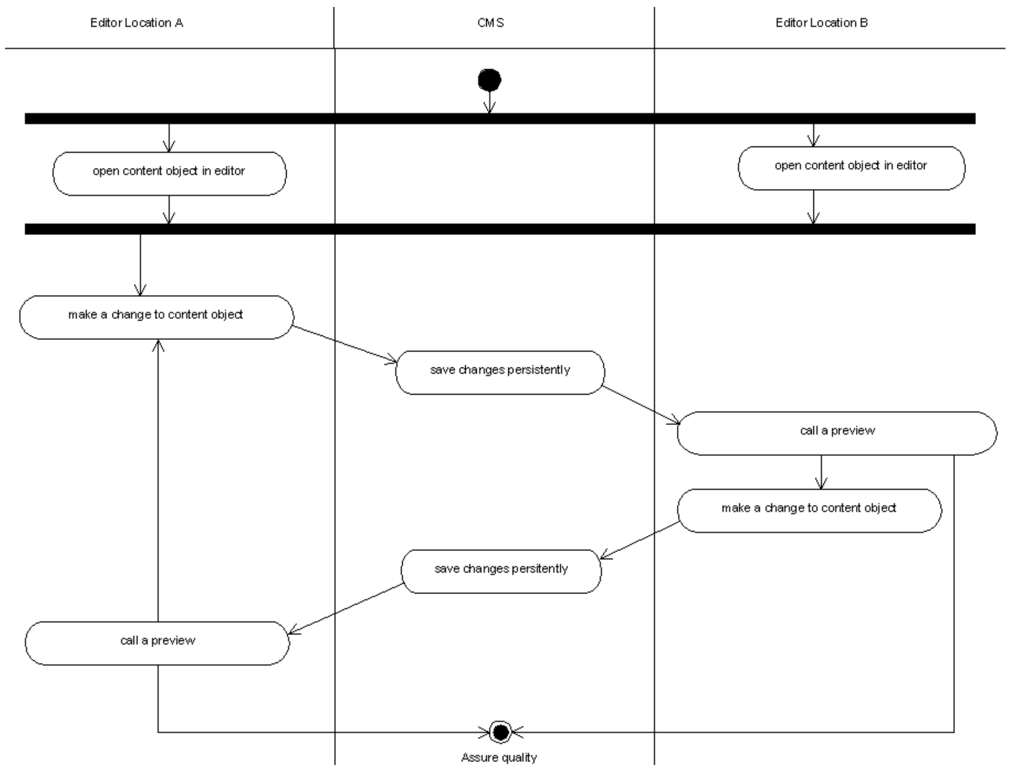

Editor Location A
CMS
Editor Location B
open content object in editor
open content object in editor
make a change to content object
save changes persistently
call a preview
make a change to content object
save changes persistently
call a preview
Assure quality

Figure 2 – Activity diagram: editing content object in international collaboration

In the beginning of the use case that is illustrated by the activity diagram, both editors open the content object that they want to work on in their user interfaces, that presumably is a WYSIWYG[1] editor. This is in so far important as both will work on this content object conjointly, not meaning at the exact same time, but sequential. In parallel they discuss the issues at hand, while on editor is working on a content object.

The three step process that is shown in the activity diagram consists of the following steps:

1. During the "make a change to content object" activity the editor changes something on the object. Taking a webpage as an example, this could mean that the editor adapts some texts on the page.
2. As the editing of the content object is done in collaboration with another editor and this editor is situated at a location that is different from that of editor A. So communication has to be supported by the system. Editor B can only see the changes that A made to the object, if this object is stored persistently. This is initiated by editor A and is done by the system.
3. In order to see the changes that editor A made, editor B has to refresh his view of the WYSIWYG editor, respectively call a fresh preview of the object. This places both editors back on the same basis so that further discussions, exchange of ideas and changes to the content object are possible.

This is exactly what will happen during the next three steps. They are an exact copy of the previous three, with the exception of the direction. After viewing the changes, editor B will make his adaptation of the content object. Then save the changes persistently, so that editor A can view them during step three of the process.

If further changes are needed – which is pretty likely, as the work that is carried out during this use case is creative work – the whole cycle will begin again with changes by editor A.

If one of the editors decides after looking at a fresh preview that no further changes are needed, the use case is will be quit by forwarding the content object to a step of quality assurance, which is no longer part of the use case.

[1] What You See Is What You Get editor, describing an editor that already offers a preview-like looking for the edited object

1.1.5 Slowed Down Processes due to Idle Times

It was recognized during the scenario that there were significant increases in idle times when one editor or even both were not able to work on something or contribute with an idea.

1.1.5.1 Idle Times Annoying Editors

The increase in idle times was firstly recognized when editor A made the changes to the content and B was waiting for them to complete. But this was not a new phenomenon. In former times, when all editors were physically nearer to each other, B also had to wait, e.g. until A had selected an image. The only difference was that B could see that work while it happened and thereby might have had the feeling of being more actively involved.

Secondly the editors recognized the time it took to save page as further idle time. This was a new phenomenon. The saving of the page had become an essential step in the communication of both editors. To involve the second editor as much as possible in the creation of the content, the page had to be saved as often as possible. This had not been the case beforehand. Frequently saving led to a significant increase in idle time of both editors as both had to wait for the process to complete – one saving was measured around 15 to 20 seconds.

2. Problem Analysis

2.1 Idle Times are Creating Opportunity Costs

In the motivational example it was found that besides annoyed authors, the idle times during the editing of pages created costs. The costs are the opportunity costs of the people that work on the elements as the time that they use to wait for the processes to finish is not spend for value creation.

For a calculation of costs, the following numbers will be approximated:

- there are 3,000 webpages that are part of the CMS in the whole Vattenfall group
- each of the webpages is changed in average every 3 months, so 4 times a year
- a change process of a webpage involves conservatively approximated around 5 times persistent saving by calling the save function
- the call of a save function takes around 20 seconds
- 2 editors are set idle during that time

Therefore the time spent waiting is t=3000x4x5x20sx2=2,400,000s per year, which equals to rounded 667 hours. Considering a working day of 8 hours, 83.375 working days per year are paid and spent for waiting.

2.2 Analysis of the Problem Described in the Motivation

2.2.1 International Collaboration Causes Increase in Need of Communication

During the motivation I introduced the wanted effects of a horizontal merger, which demand for an increase in international collaboration. One aim of international collaboration is the creation of synergies, meaning, there shall be achieved incremental savings or growths in revenue by the combination of two or more separate parts that are of the same nature. Therefore it would be of disadvantage, if the synergies also generate costs as a side effect. Consequently a special focus has to be put on these side effects, which can be possible factors for an increase in costs, as this would reduce the positive effects of the synergies on costs and revenues.

Although the calculation, which is the basis of the rationale, is straightforward, it is less trivial to eliminate these cost factors. So what are the reasons for the possible increase in costs?

The most likely cost factor having the highest impact is the increase in time needed for the completion of tasks due to the locally dispersed teams having a significantly higher need for communication [VCT04]. The reasons therefore on the hand may be task-related and on the other hand may be based on social challenges.

2.2.1.1 Working Conjointly

The possible task related problems result from the lack of physical proximity of the team members. In daily business as well as in projects most tasks are not solved by lone fighters but in small teams. Therefore people have to get together to work conjointly on tasks. Especially in day-to-day business, meetings are neither frequently possible nor could such effort be justified when the people would have to join coming from different countries. Moreover most of such problems arise spontaneously and are not foreseeable, a physical meeting is therefore impossible. Virtual meetings must be established.

Usually the virtuality is established by means of information and telecommunication systems. As IT systems in general offer an artificial way of communication and add another step to the communication chain between A to B, they are a potential source for time delays. Moreover people have to familiarize themselves with those new processes and ways of communication to increase efficiency.

The lack of physical proximity and the associated need for virtuality bear a high risk for increases in processing time of certain tasks, i.e. an increase in costs.

2.2.1.2 Team Building Measures

Firstly the physical proximity, which generally provides the basis for social habits and the building of relationships, is missing in the new context. Team building activities that are dedicated to sociality and relationships are necessary and can work as compensation. This surely will create additional costs. Besides the direct cost for the meetings, the indirect costs for not “generating output” during that time will increase the overall costs significantly.

2.2.1.3 Cultural Differences

Secondly the internationality in the collaboration lets people face different cultures and habits of their new team members. Different local as well as corporate cultures create further needs for more intense communication to overcome differences and problems.

2.2.2 Need for Faster Processes in the CMS

Summarizing it can be stated that the problem, raised by the motivation, is an increase in idle times of editors that causes significant costs. The problem is based on changes in working processes. These were triggered by the introduction of international teams and collaboration in the context of Web content management systems that made it essential to use the CMS as a means of communication.

Therefore a need has been discovered for fast processes in those functions of a CMS where international collaboration on the system is in way frequent such that idle times can be reduced to a minimum.

2.3 General Problem Statement

In the context of content management systems, international teams, consisting of locally dispersed members, rely on data-intensive communication via the content management system. Therefore idle times of multi-sited processes have to be minimized to guarantee efficient working.

3. Distributed Systems

3.1 Efficiency in Distributed Systems

3.1.1 Defining Efficiency

3.1.1.1 Economic Efficiency

Given that the striving for efficiency is motivated by a business case that is concerned about increases in costs, efficiency should be defined firstly from an economic perspective. In economics efficiency is basically defined as *the property of society getting the most it can from its scarce resources* [PE04, page 5]. So it is the economic term for creating the biggest possible outcome with the smallest possible costs. Efficiency thereby influences massively the profit as it defines the difference between revenue and cost – being defined as profit. On the one hand this is about maximizing the output while keeping the cost constant. On the other hand it is about minimizing the costs, which can also be defined as wasting as less as possible.

3.1.1.2 Energy Efficiency

As distributed systems are technical systems where information is processed, a technical definition of what efficiency means will be looked upon as well. In physics and engineering the according term is energy efficiency. It is a dimensionless number, with a value between 0 and 1, being defined as

$$efficiency = \frac{power\_ouput}{power\_input} = \frac{W}{energy}$$

Efficiency is the useful work (W) that is done consuming a certain amount of energy.

3.1.1.3 Efficiency in the Context of this Project Work

Bringing the two definitions together one can define efficiency as: *maximizing the useful work* while *keeping the cost for it as small as possible*. As the costs in the context of this work are the investment of time, one can rephrase the definition to *maximizing the useful work* while *keeping the investment of time for it as small as possible.*

3.1.2 Analyzing Efficiency

3.1.2.1 Useful Work is Constant

When thinking of technical, e.g. mechanical or electrical, systems the aim of an optimization of this system often is to reduce useless work as friction or the production of waste heat. By this the efficiency shall improve meaning there will be a higher output of useful work for the same amount of energy. This is very demonstrative for systems of that kind.

In computer systems the improvements most often work the other way around. The system has to fulfil a certain defined task. So the desired, useful output is set and can be seen as constant.

3.1.2.2 Costs are Variable

In contrast to the target, the way that is taken to produce the output is not set and not constant. So there can be taken different approaches with different amounts of work – useful or not – to reach the goal. In the context of distributed systems this work is called an investment. These are the costs for reaching the set goal.

Therefore the useful work, respectively the desired output will be assumed as constant, and the costs as variable. By decreasing the investment the efficiency will increase.

3.2 Trade-offs in Minimizing the Investment

The costs of the work are the investments that are made into the system to reach the desired goal, respectively complete a given task. The investments that are made in this context are investments of time. As the goal is to maximize the efficiency, the aim must be to minimize the invested time, called t_i.

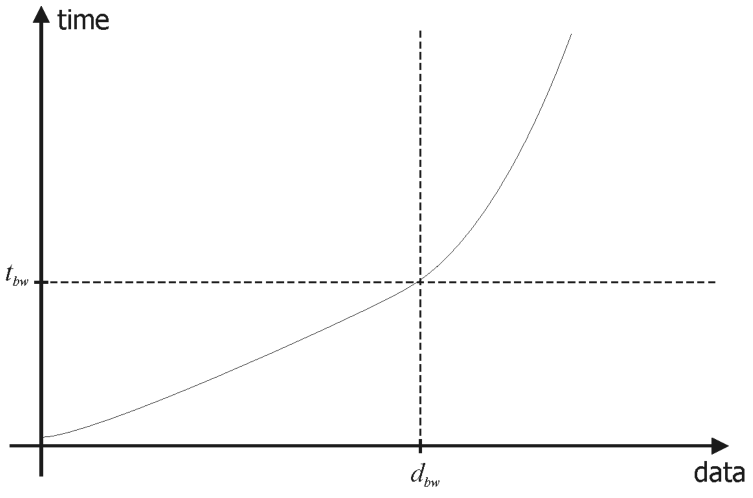
As distributed systems consist of different, and, as the name implies, distributed components most of the operations on data items are carried out remotely. These remote procedure calls involve the problems of limited bandwidth as well as of an increase in round-trip time, e.g. by latency.

3.2.1 Round-trip Time

The *round-trip time RTT* is the time needed for a package of data to be sent to a remote place, be processed and sent back again. The RTT is dependent on the following features.

3.2.1.1 Bandwidth

On the one hand there is limited *bandwidth*, which sets the maximum throughput, i.e. amount of data transferred over a period of time that the system can provide.



Graph 1 - Data to time relationship with limited bandwidth

As the system is able to serve a certain bandwidth, the limitation will have no effect on the time needed to transfer the data up to the limit d_{bw} . It will just take the time that is needed to transfer this particular amount of data. Above the limit this is different. Here the waiting time of later data packages for transmission of their predecessors adds upon the time needed for the transfer.

Whether bandwidth is a problem that has to be taken into account or not when thinking about investments, has to be decided when looking at the concrete system and the data that is transferred in it.

The time for transmission of a certain amount of data will be called t_{tr}. t_{tr} is dependent on the size of the data package that is to be transmitted.

3.2.1.2 Remote Procedure Call

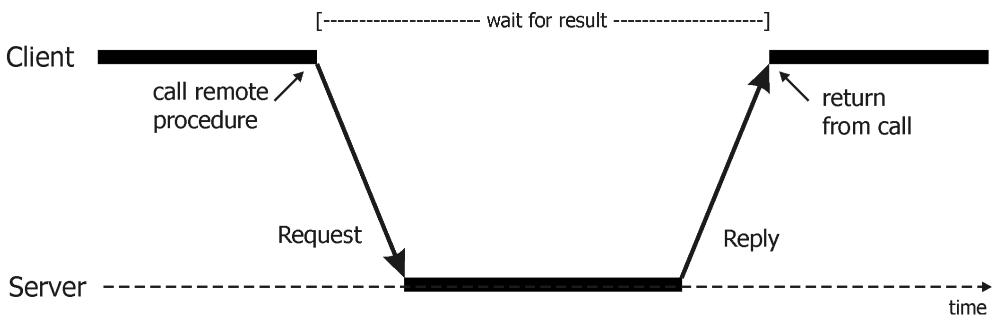


Figure 3 - Principle of RPC between client and server [DS02, page 72]

A basic feature of distributed systems is the *remote procedure call [RPC]*. Operations are spread across different and distributed components. Instead of calling a procedure locally on the machine, the call will be marshalled and sent to a server. The server will analyze the message and decide which procedure to call. The retrieved result will be packed again into a message and will be sent back to the client. There it will be unpacked and the result will be returned to the client procedure that invoked the RPC.

The remote procedure call costs significantly more time than calling a local procedure. The following aspects explain the reasons for its round-trip time.

1. The procedure call must go through a step of *marshalling*, i.e. telling the server how to interpret the message and which procedure to call. This message has later to be unpacked by the server and analyzed. The same is true for the reply. The time needed t_{ms} adds significant overhead to the operation.
2. The time t_l that the data needs to be transferred from the client to the server and back, adds as well on the round-trip time and is called *latency*. Even assuming the data is transmitted at the speed of light, t_l will add a certain amount of time to the round-trip. In a complex network the latency increases especially by the data being processed in intermediate gateways and proxies along the way to its actual target.
3. On large distributed systems it is common to have multiple machine types present. This may lead to differences in the *encoding* of data. Thus to invoke a procedure on the server there might be first needed a conversion of parameters. The time for this step will be called t_e.
4. The last part that adds to the round-trip time is the procedure call itself that happens on the server. The time needed for the operation on the server will be t_{op}.

3.2.2 Granularity

The *Granularity* g specifies the degree up to which the data that is analyzed and sent over the system is subdivided into smaller units. The reason for doing so is that the subdivision of data makes it possible to look individually at the smaller pieces. As it was stated before, there is only a value in sending data that is new or was changed. By looking at the smaller pieces individually one can identify unchanged ones and prevent them to be sent and processed.

There might exist systems that have an optimal granularity of 1, meaning that all of the data will be sent in one block as well as it might be possible to think of a system or a process where the optimal granularity is at a point where the whole data is separated in single bits. Generally the optimum won't be one of theses extremes, but will lie somewhere in between.

Comparing different granularities only makes sense in the context of one process, as the optimal granularity will be affected by all components of the round-trip time as well as of the time that was needed to achieve a valuable granularity, e.g., dividing the data and comparing each piece to an older version.

3.2.3 Time Investment in Data Transmission and Remote Processing

3.2.3.1 Investment for a Single Data Package

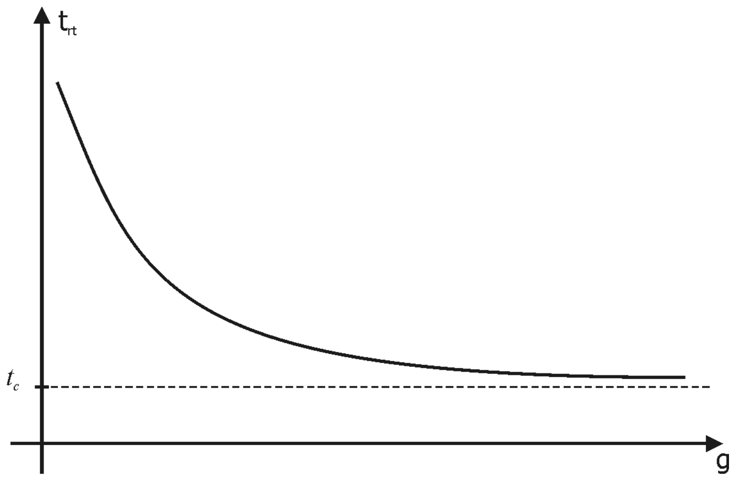
The round-trip time t_{rt} that is needed for one RPC to complete, consists of a constant and a variable part, in respect to the granularity g .

The constant t_c part consists of the times that are needed for the overhead operations, meaning those processings that are made and are not the actual operation on the data that is to be carried out. This covers the times t_{ms} , t_l , and t_e .

The variable part t_v covers those times that change accordingly with a change of g , in respect to a single process. A higher granularity results in smaller data packages, therefore the time needed for transmission t_{tr} and the time for the actual operation on the remote component t_{op} will be smaller in that case. Describing particular process the times t_{tr0} and t_{op_0} are the times that are needed for the whole data to be transmitted, respectively be processed. Therefore the variable part of the time is

$$t_v = \frac{1}{g}(t_{tr\,0} + t_{op\,0}) \, .$$

The complete round-trip time for the processing of a single data package in dependency of g therefore is $t_{rt} = t_v + t_c = \frac{1}{g}(t_{tr\,0} + t_{op\,0}) + t_{ms} + t_l + t_e$



Graph 2 - Round-trip time for a sinlge data package in depedency of the granularity g

3.2.3.2 Investment for the Whole Data of a Process

Regarding the data of a whole process there will presumably be more than a single data package transmitted and processed. The number of packages that will contain changes and therefore will be processed depends on the changes that were made to the data before. To approximate such a value, one will have to look at a particular process and assume a certain pattern of how data is changed, meaning how single changes are distributed over the whole data and how much of the data actually is changed, and the extent of an average change is. Thereby one has to approximate an average percentage value n of single data packages that contain changes.

The difficult thing about this approximation is that on the one hand a higher granularity will lead to a smaller value of n - because more packages will contain unchanged data – and on the other hand the granularity g shall be chosen with regard to the n.

Thus the investment t_p for the whole data of a process will be ng-times the investment for a single process, and results in the following formula for t_p:

$$t_p = ng\left(\frac{1}{g}(t_{tr0} + t_{op0}) + t_{ms} + t_l + t_e\right)$$

3.2.4 Analyzing Data to Be Sent

In one of the previous paragraphs the granularity g has been introduced as the degree of subdivision of a data package. The benefit was described to be the possibility of differentiation between parts of data containing new or changed information in contrast to those that do not contain any new information. Thus it is possible to send and process only those parts that contain changes and save the time t_v of sending and processing the rest. In the case of few changes and a high granularity, it will reduce the time t_v significantly.

3.2.4.1 Order System Example

As an example one could imagine an order system of a manufacturer. The sales agent collects the orders with some software in his notebook over the day and uploads the new orders every night via an access over the internet to the central system at the manufacturer. Due to certain reasons one order is changed the next day and must be updated on the central system as well. As those connections to the internet that the agent uses do in general not offer high bandwidths and are most often expensive as he dials in from different hotels, it is preferred to have as few data send as possible. Therefore the order program he uses keeps a record of the last version that was sent. In order to update the record on the central system the software will extract the parts that were changed in comparison with the last version and will add some information that tells the receiver about where to change or insert that data. Besides the case that all data was changed, the sent data will be significantly fewer than if just all of it would have been sent, regardless of the particular change, to overwrite the whole old version on the central system.

3.2.4.2 Granulate the Data

To reduce the amount of time needed to send and process the data in the system, it has to be detected beforehand which parts of the data contain new or changed information.

In the case of an insert-function it is likely that all of the data that is given is new and needs to be processed. In the case of an update-function this is less likely to be the case, as in general not all of the data was changed, but only parts of it. So the value lies in those parts that are changed, respectively are new.

The general concept is to check the granulated data for changes and then only process the changed parts. The granularity g is decided on beforehand. It might consider a natural differentiation, like different objects in a content management system, but it can also be thought of a granulation looking for changes of words in a longer text, therefore dividing the whole text into single words.

Besides the division one has to invest into the execution of methods that check if there was a change in a particular part. Regarding a string this could be a method that compares the single words or fragments to the previous version, regarding an object one could consider reading out a flag indicating the change of the object.

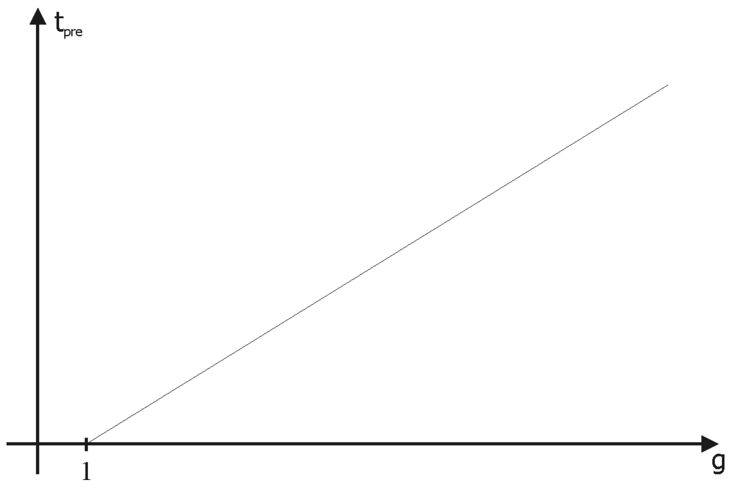
The decision-to-make is on how small or big to choose the granularity g, as a higher granularity includes as well a higher investment in methods for change checking. Although it might offer a very efficient way in terms of transferring and processing the data for certain systems, taking the smallest possible unit into consideration and checking each bit of the bit stream would mean as well to call a method for change checking for every single bit. The investment that is needed to detect the changes might therefore be significantly higher than investing in the transmission and processing of unchanged data. In this case there would be nothing earned but lost.

A further important point that has to be considered and might be even more critical than the investment in the execution of change checking methods is the creation of additional round-trips, each of which will add its overhead time t_c to the total processing time. A higher granularity comes with a higher probability for creation of such extra overhead, and thereby with a higher probability of not earning something but loosing additional time.

3.2.4.3 Time Investment for Division of Data and Checking of Changes

The time t_{div} to divide the data into the single parts, and the time t_{check} for executing methods that check a single part for a change make up the investment. Regarding the complete process the single times are proportional to g, as the data will be split into g parts which afterwards will be separately analyzed, e.g., compared with a previous version of that part. Consequently this operation also happens g-times.

Thus the time that is needed for this pre-analysis is $t_{pre} = (t_{check} + t_{div})g$



Graph 3 - Time for preanalysis of data dependent on granularity g

3.2.5 Trade-offs in Time Investments

The time investments for a process split into two separate parts. The first part is the pre-analysis of the data that was dealt with in the previous paragraph. The second part is the transmission and processing of the data. But regarding the aim of finding the smallest invested time t_i dependent on the granularity g, a better split can be made.

There are two kinds of time investments, those that shrink with an increase of g and those that grow with it.

The time t_p includes all processing, transmission and overhead time for the handling of all changed data packages in a process. This time can be split into the two groups, mentioned before. While the time $ng\ t_v$ - introduced as the variable part in the handling of data packages – shrinks with a growing g, the time $ng\ t_c$ - the times for all overhead operations – increases with a growing g. Increasing with a growing g is as well the time t_{pre} that is needed for the pre-analysis.

Thus distributed systems face a trade-off between the investment in shorter round-trip times and the investment in the locating of the changed information as well as an increase in overhead time.

3.3 Content Management Systems

Content management systems are a special kind of distributed systems, and, as the motivation indicated, wide spread in the business world. Therefore the findings in the domain of distributed systems will be checked and – if necessary – be refined against the general concept of content management systems.

4. Content Management Systems

4.1 Content Management Systems as Means of Communication in International Collaboration

"The Internet is changing many of the unwritten rules that businesses have been following, consciously or otherwise. By connecting people and organizations electronically, and by offering new opportunities for businesses to communicate and interact with customers, partners, employees, and even with actual and potential competitors, the Internet forces [...] all [businesses] to rethink how [...][they] operate and manage web properties." [WCM02, page 3]

The result of the rethinking that is mentioned in the previous citation is most often expressed in the introduction or use of a Web Content Management System. Content management systems are in wide use in any kind of business these days as means of effective administrating the online presence of a company. Already for a long time the systems have outgrown their stage of infancy and established themselves as professional tools for communication of the brand and image of a company and platform for interactive communication with existing and potential customers.

It was shown in the motivation that international acting businesses, especially those that grew by merger and acquisition of different companies, have to restructure the international and national activities for brand communication. Particularly all activities connected to the Web are affected by the establishing of new processes involving international collaboration, as the internet generally is the most international media that companies use.

Due to new structures and not established processes, there does not exist any infrastructure that would have been particularly implemented for the support of the international collaboration. Thus the existing systems must provide the necessary tools.

For the increasing need for communication on an international level and thereby the increasing need in support by the systems for that communication was already argued in the second paragraph in the context of the problem statement. It was argued as well that especially content management systems have to support these needs.

Thus content management systems, as a special kind of distributed systems, will be examined in the following in terms of processes that involve a high potential for idle times of users, in order to find possibilities for optimization regarding their time efficiency.

4.2 Activities on a Content Object

The following diagram describes the general flow of common activities that are carried out on a content object and can be found in such or a similar way in every content management system. The activities are generally carried out by different persons or just by system itself.

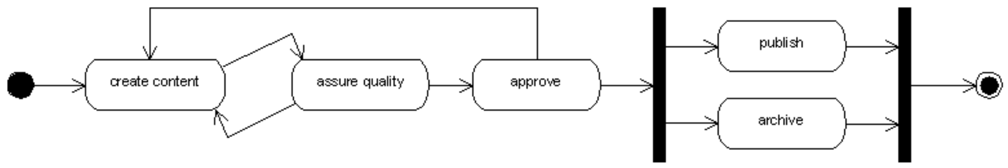


Figure 4 - Activities on a content object

In order to identify the states that are mostly affected by the change in working processes and communication each state will be described and checked in the following.

4.2.1 Activity States

4.2.1.1 State: Create Content

The “create content” state generally is the first action that is performed. Beforehand a new content object might have been created or an author might have decided to change some content of an existing object. Therefore the “create content” action will be triggered.

The action might also be triggered at the end of the “approve” state, when the approval can not be given, and the object has to be re-edited.

During the “create content” state the content of the object will be created newly or changed according to certain specifications for the object. This action highly involves user activity. One or more authors use the provided frontend – mostly a WYSIWYG[2] editor – to create the content.

When finished with the creation or change of the content, the object is pushed further to the quality assurance.

[2] What You See Is What You Get editor, describing an editor that already offers a preview-like looking for the edited object

4.2.1.2 State: Assure Quality

During this state another author or editor is going to assure the quality of the content that has been created during the previous state. In case of a page being the object that it checked, this would include the matter of texts, the right placement of images, the check for disclosure of confidential information, etc.

If the object passes the quality check, it will be pushed forward to the approval. In the case of a failure, it will be rejected and pushed back to the “create content” state, where the failure will be corrected.

4.2.1.3 State: Approve

During this state the primary task of an editor who might be responsible for a particular section of a Web presence – or even the whole, is to check the appropriateness of the provided object into the overall concept of the Web presence. This might be especially demonstrative when thinking about the object being a page.

If the editor refuses to approve the object, this will trigger the “create content” state again, since further editing is required.

If the editor approves the object the “archive” state will be triggered.

4.2.1.4 State: Archive

The “archive” state takes care that all objects that will be published will be archived as well, generally including a version control system so that older version later can be restored, to undo changes or use a previous version as basis for a new object.

The action is executed in parallel to the publishing of the object. The execution of the archive action in parallel to the publish process does not concern any user activity and is completely carried out by the system.

In general the archive action could also be part of the edit activity. This basically depends on the rules and the underlying object model for the archiving. It can be as well thought of a CMS where every object consists of sub-objects. Therefore a rule could be that every change of a sub-object is particularly stored in the archive. But as there is in general more than one suitable way of handling the archiving, in the context of this model it will be assumed that only published objects – meaning approved to be presented – are archived.

4.2.1.5 State: Publish

The approved object is being published to the live website. As it was already the case for the “archive” state, this action does not involve any user activity. Only the system itself is involved as this is a fully automatic process.

4.2.2 The States and International Collaboration

Firstly the states Archive and Publish can be ruled out. Those actions are only carried out by the system without any involvement of a person. Therefore it is straightforward that these states won’t involve interpersonal communication and thus need not to be considered for further investigation - in contrast to the rest of the states.

The other states that are given in the model all can be actively involved in international collaboration as all of these involve interpersonal communication. The question to answer is up to which degree the processes and methods that are carried out during a state create a possibility for idle times in communication.

4.2.2.1 Approve

With regard to the “approve” state the possibility for idle times is very low, respectively not existent. In respect to the described model and especially the description of the “approve” state, the role of the editor that does the approval does not involve much direct communication with other team member, particularly no direct communication while the approval is done. As the approval generally is done by a single person that keeps track of the overall “orchestration” of the objects of a particular section or the website as a whole, the direct involvement of another person is pretty unlikely. Thus this state can also be ruled out from further investigation.

4.2.2.2 Change Content Objects

The remaining two states, “content creation” and “quality assurance”, can not be ruled out. Both states deal directly with the editing of a content object, therefore have a need for human interaction. As it was stated in the paragraphs before this interaction in the context of this work can be assumed to happen on an international basis. Therefore there is a need for using the system as means of communication, thus there is a need for fast processes.

Additionally one can interpret the two states already as representation for the activities during international collaboration, as involving different people in the creation of an object already is similar to one creating the content and the other assuring its quality. The basic difference is that the transition from one state to another and back will probably happen more frequently than in the model.

Although the particular end of each of the two states is different – one is there to create the content, the other is there check for quality – the means both use will presumably be the same processes and methods, it is pretty likely that both will use the same user frontend to make their changes to the data.

Therefore the two states can be generalised in the state “change content object”, as this is what happens primarily during the activities of both states.

4.3 Probability of Idle Time for Users while Working on Content Objects

In order to optimize multi-sited processes in terms of idle times for users, the view has to be focussed further. The pure identification of activity states, where the content management system is used as means of communication, is not enough. Those processes have to be identified that bear a risk for idle times.

The following processes can be assumed to be common for every content management system and very frequently used during international collaboration, therefore having a big impact, if idle times occur.

4.3.1 Edit the Content

4.3.1.1 Process Description

The user frontend offers a particular view to edit certain content objects. For the example of a webpage that might be the possibility of orchestrating different other content objects according to a certain structure, for the example of a text fragment that would mean the offering of a text editor to create new texts or change existing ones. Content objects can be of various kinds, e.g. texts, links, downloads, images.

4.3.1.2 Data Processing

As most of the editing will be done in WYSIWYG[3] editors the changes have to be saved somehow after they are done to be visible for the author. Because this saving is

[3] What You See Is What You Get editor, describing an editor that already offers a preview-like looking for the edited object.

only done to make changes temporarily visible to the author, they will not be stored persistently, but will be kept volatile. This does not only make the whole process a lot faster, but also prevents the system from storing information that were not intended to be stored.

Thus data processing is only done in the presentation layer of the system.

4.3.1.3 Probability of Idle Times

As the operations on the data are only performed in the presentation layer of the system and do not involve communication with the persistent storage of the CMS, the process can be assumed to be fast.

Nevertheless there can also be implementations of the user interface that prohibit such fast acting. Therefore it is not completely ruled out that there is a need for optimization.

4.3.2 Save Changes Persistently

4.3.2.1 Process Description

After the changes were made to the content objects, as described in the previous section, the author triggers the system to store the changes persistently. Therefore the data will be transmitted and processed on the persistent storage of the system.

This process is a necessary precondition for the forwarding of the object to the next activity state, as it was described in the model from the beginning of this paragraph.

4.3.2.2 Data Processing

The persistent saving of an object is a very costly process. The underlying principle is that of a remote procedure call, as it was introduced during the previous paragraph. The even more costly add on to the general overhead time is that the persistent storage of an object involves the interaction with the repository of the CMS. Generally this involves a very time costly step of checking out the object and checking in the changed object.

4.3.2.3 Probability of Idle Times

Due to the information given in the previous section, it is pretty likely that the data processing might take a long time. If it is not optimally implemented, this process will offer a big possibility of performance optimization.

4.3.3 Call a Preview

4.3.3.1 Process Description

A preview of the content object will be rendered to see the current object in its later surroundings, e.g. the real look of a page in a website.

4.3.3.2 Data Processing

There exist at least two possibilities how to implement the preview.

The first way is to let the preview get the data that is needed for the preview from the volatility saved data of the presentation layer and let them be rendered. This would involve some read operations on data stored in the system memory.

The second possibility is to let the preview show the data that is persistently stored in the repository. This would involve some read operations on objects of the repository. It can be assumed to be already slower than reading the memory, but the bigger problem of that process will be that it presumes the saving of all volatile changes beforehand so that they are displayed in the preview – and generally speaking, that is what a preview is used for.

4.3.3.3 Probability of Idle Times

In contrast to the first possible implementation, the second one bears a high risk of idle times for users. As it presumes the saving process as a necessary condition, this implementation would underline even more the importance of fast saving processes.

4.3.4 Refresh View

4.3.4.1 Process Description

Refreshing the view will be done on the one hand automatically in the WYSIWYG editor after changing an object, and on the other hand can be initiated by the user himself. This might be the case when an author has made some changes to an object on a user interface at another site, and another author wants to make the changes visible to him to give the first author some feedback.

4.3.4.2 Data Processing

The first way of refresh would involve only the reading of objects that are stored in the system memory. Therefore this process can be assumed as very fast.

The second way of refreshing the view will include the need for the reading persistently stored data from the repository.

4.3.4.3 Probability of Idle Times

In the second case there is a small probability for idle times as the system infrastructure is unknown and the reading from the repository might involve some extra, time consuming steps.

4.4 Efficiency in Content Management Systems

The previous paragraph about efficiency in distributed systems resulted in a formula for t_i, the investment of time for a process. It was argued that t_i is dependent on particular time constants and the granularity g, defining the degree of splitting the data.

4.4.1 The Data to be Processed

4.4.1.1 Maximum Data Amount

The distributed system which needs to be analyzed in the context of this study is a content management system, in particular a Web content management system. The CMSs that are used in those kinds of businesses that we look at are generally in use to administrate bigger amounts of aggregated Web pages. Therefore it can be assumed that the biggest amount of data that will be processed by the system is a page. In terms of the CMS this would mean the sum of the single objects, references to objects and the information on the structure of the page.

As such a page must as well be recallable for visitors of the website, the amount of data that such a page can contain is very limited.

4.4.1.2 Remoteness of Users

A possible problem might be the remoteness of some users, as we are looking at processes that are in the context of international collaboration. The slowest way for an author to have access to the content management system can be assumed to be by calling in over the internet. However this would only mean that the user interface of the author has to request the same amount of data as the user interface of a normal visitor of the website. Assuming that a goal of the authors is to protect the user from long waiting times, this implies that the authors do not have longer waiting times as well.

4.4.1.3 Bandwidth

Taking into account the assumed amount of data, there will be given no limitation by the bandwidth that would slow down single processes – taking as well common, current system infrastructure into account.

4.4.2 Times

4.4.2.1 Transmission

Due to the argumentation during the previous section the bandwidth will not be of any problem, meaning slowing down the transmission of data. The same is true for the amount of data that is transmitted, as it won't impose a challenge to a common, up-to-date infrastructure. Therefore the time t_{tr}, needed for transmission, can be regarded as minor.

4.4.2.2 Latency

The problem around the latency is that it is very specific to the infrastructure and structuring of the particular content management system. The same is accordingly true for the time t_l that will be consumed due to latency.

Assuming the unlikely case of a very broad distributed CMS, with components located on the whole globe, then the latency will be a problem for sure. Many intermediate steps, processing in gateways and proxies, and the long distance in between the components will cost significant amounts of time.

On the other hand one can assume the exact opposite that components are located physically near to each other, thus distances in between are neglectable small, and only a bit of intermediate processing has to be done. This will not lead to any increase in the round-trip time of a data package.

4.4.2.3 Processing

Again one can reference to the discussion on the amounts of data that are sent over the system. Firstly one has to considering the amount of data to be in the boundaries that have been described before, secondly the operations that are executed on the data are limited to read and write operations of data. No expensive calculation or similar has to be made. Therefore the time for processing of the data can also be assumed to be of minor importance.

For the transmission time as well as for the processing time, it can accordingly be assumed that the reduction of the amount of data that is send and be processed in each data package only has a minor influence on the reduction of idle times of users.

4.4.2.4 Overhead

During the paragraph about efficiency in distributed systems, the – per data package constant – overhead time t_c was introduced as the sum of times for different operations that are needed in a distributed system, e.g., the time for marshalling.

A special feature of content management systems that adds its time up on top of the sum of these times is the check-out and check-in of objects. As it was already mentioned in one of the previous sections, a content management system includes a version control system, where all of its objects are registered. To change one of the objects, the particular object has to go through a check-out step, is changed and goes reversely through a check-in step, where a new version will be created in the system.

In contrast to the previously mentioned possibilities for time delays, these operations for checking in and out are not trivial and might take a significant amount of time. Consequently these operations bear a high probability for idle times of users. The time t_{cico} for a check-in as well as for a check-out can be assumed to be constant and equal for every object, and as well as a must for the persistent change of an object.

4.4.3 Granularity for Content Management Systems

The recently introduced time t_{cico} already sets the direction for the search of an optimal granularity g.

On basis of the previous section it has to be argued that, regarding the processing of a single data package, the following ratios exist.

$$\begin{aligned} & t_{tr} << t_{cico} \wedge t_{op} << t_{cico} \\ \Rightarrow\ & t_v << t_{cico} \\ \Rightarrow\ & t_v << t_c \end{aligned}$$

Considering the formula for the round-trip time t_{rt}, it must be argued that $t_{rt} \approx t_c$ per data package – in the case that one data package does not include data from more than one package.

For the processing of the whole data of a single process this leads to the time $t_p = (t_v + t_c)ng \approx t_{cico}ng$.

4.4.3.1 The Upper Boundary of the Granularity

The consequence of this is the argumentation that in order to reduce the processing time t_p, the number of data packages containing changes ng must be minimized. Thus the granularity g must be as small as possible. All of this argumentation still is done under the assumption that a data package does not contain information of more than one object.

4.4.3.2 The Lower Boundary of the Granularity

The assumption that a data package does not contain information of more than one object has to be made because otherwise there will be more than one check-out and check-in cycle involved per data package. Due to the adapted formula of t_p, $t_p \approx t_{cico}ng$, this would mean for the processing of information fragments, belonging to k different objects, the processing time will be as well $k\,t_p$. This is the same amount of time that would be needed to send and process the k data packages individually. From this point of view, the lowest boundary for g is the number of objects that the data belongs to.

4.4.3.3 The Optimal Granularity

Bringing the upper and the lower boundary together, the result is straightforward. With regard to the previous argumentation the optimal granularity g for content management systems can be defined as the number of content management objects that are involved in a particular change process.

In cases where only one object is modified, there is no sense for further splitting; in the case of more than one object being modified at once, e.g., a modification of various elements on a page, there is no value in wrapping all objects or data together into one.

As this splitting is somehow a natural one, as the objects are already given by the system, it can be assumed that there will be no need for investments into that splitting. Thus this process of splitting has no influence on the choice of the optimal granularity.

4.4.4 Checking Objects for Changes

The question that is not answered by deciding the optimal granulation is whether or not costs are reduced by checking each individual objects for a change instead of processing blindly all objects. While the costs for an extra round-trip are known, $t_{rt} \approx t_c$, the costs for the analysis still have to be determined.

The time t_{check} that is needed to check a single object for a change is very system dependent and cannot be completely generalised. Assuming that the content management system confirm with the paradigm of object orientation, it would enable a very cost efficient solution for the checking. Each object that was changed could have a certain flag set, that identifies it as being changed, or a reference to the object could be added to a list that then is processed. The checking for change mechanism would thereby not take very long time, thus would be very low cost. But again, this can not be generalized for all systems, as there might be external factors that prohibit this or another solution.

In contrast to the low cost checking method, the costs for another round-trip can be assumed as high, as already explained before.

Therefore the recommendation should be to have a checking mechanism implemented that decides if an object was changed or not and takes care that only changed objects are processed.

5. Practical Application: The save function at the Vattenfall CMS

The business case that motivates this project work shall now build the basis for a practical application of the findings of rules for optimization.

During the previous paragraph those activities and processes of a content management system were identified that bear the highest risk for idle times of users. The process to save changes persistently that were made to an object while editing it, is one of the most called methods during the editing and bears a high risk of idle times.

This process also occurs in the business case. Therefore it seems to be the optimal process to be examined as an example.

5.1 The CMS at Vattenfall

Vattenfall is using the OpenText LiveLink Web Content Management Server in Version 9.5. On top of the middle layer of the editorial system the team of Vattenfall Europe Information Services E-Business Solutions has created an own middleware layer as well as presentation layer.

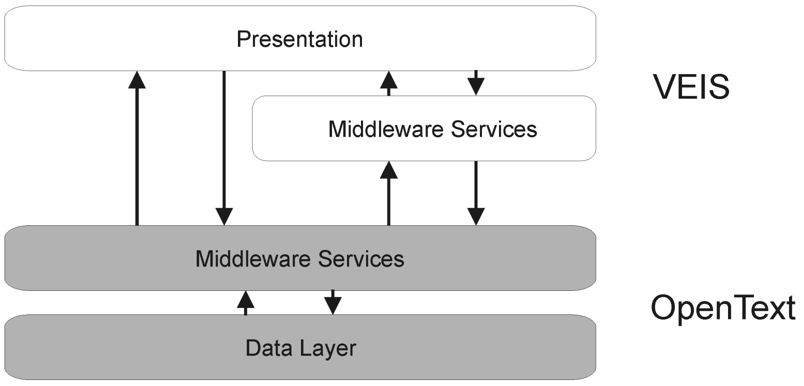


Figure 5 - Layers of the CMS at Vattenfall

Due to complex and sophisticated requirements at the time when Vattenfall started introducing a CMS for all of their Websites, it was decided to create a custom object model and user interface.

5.2 Editing a Content Object

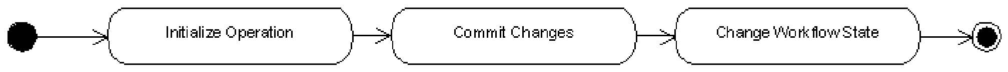
The activity “editing a content object” was already described during the motivation. The activity diagram in figure 1 illustrates how this is carried out at Vattenfall when involved in international collaboration.

When comparing it with the model that was given in the forth paragraph in figure 4 one will discover that this activity is like a wrapper around the activities “create content” and “assure quality” of the figure. Thus there is a definite match of the concrete example of the “edit a content object” activity at the Vattenfall CMS and the model. The CMS at Vattenfall is a particular implementation of the model given in the forth paragraph. Therefore the general results that were found for the model will also be applicable for this concrete example.

5.3 The Save Method Process

Due to the argumentation in the previous paragraphs for the importance of the save method, it will be examined further. The following activity diagram will show the particular activities that are carried out in the “save” method.

5.3.1 Activities of the Save Method



5.3.1.1 Initialize Operation

During the activity to initialize the operation all needed components will be initialized. The time needed for this part of this part was found to not increase the overall time of the process in a significant manner.

5.3.1.2 Commit the Changes of the Object

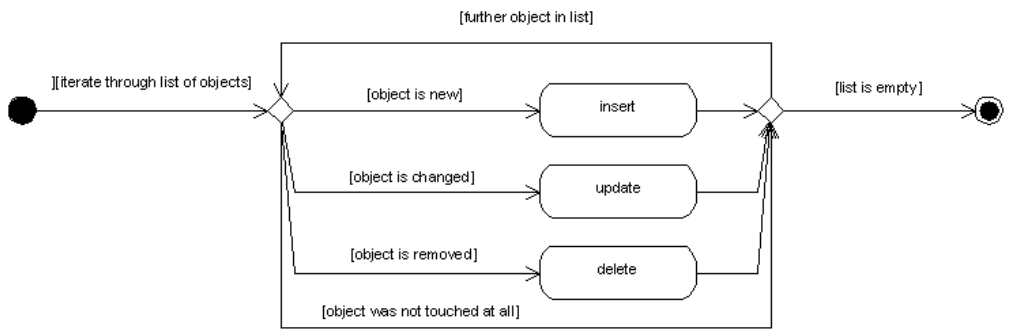


Figure 6 – Activities during committing changes of objects

During the activity “commit changes” a list of objects will be parsed through a mechanism that checks whether or not the object was changed in a certain manner. The mechanism checks a flag value for one of the four possible cases shown in the diagram.

The three activities contain further actions that will lead to the execution of methods that will do the according processing. Independent of the activity being an insert, an update or a delete, a check-out of the object on the repository will be performed. After the particular action is done, e.g., updating the object by copying the fresh content, a check-in is performed as well. Before the check-in is performed, all objects are archived in there previous version, so that they can be restored later.

In the example of editing a webpage, all text blocks are single content objects and will be handled individually. The deleting or change of one of these text blocks would firstly lead to the performing of the according action and secondly lead to the previous version being archived.

If the flag is not set at all, it indicates that the object was not changed. Therefore no further action is needed.

5.3.1.3 Change Workflow State

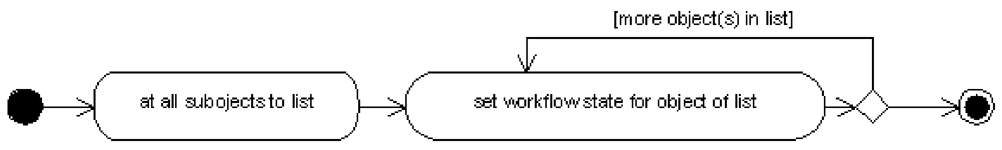


Figure 7 – Activities to change the workflow state

In the beginning of the activity the repository is scanned to create a list containing the object as well as all of its sub-objects. This list is being made each time from scratch, in order not to miss a newly inserted object.

After the list is filled, each object of the list will get its workflow state set. As this activity is situated at the end of the editing-activity of an object, the state will consequently be set to “EDITED”.

5.4 Performance Measures

5.4.1 Performance Measurement Tool JAMon 2.0

The Java Application Monitor (JAMon) is a Java API that allows monitoring the timely performance of applications. JAMon can gather aggregate performance statistics such as hits, execution times (total, average, minimum, maximum, standard deviation). It can be thought of being an aggregated database that is held in the memory. Instead of saving all single hits, only the aggregations are saved and new entries are set against the aggregated numbers. [JA06, par. 1]

This behaviour allows JAMon to have a very small memory footprint and thereby not influencing performance measures in a negative way by blocking memory for the actual applications. [JA06, par. 2] This advantage comes on the other hand with the disadvantage of not very sophisticated possibilities of measuring performance and provided analysis of the data – but actually enough for the purpose that was here at hand.

5.4.1.1 Monitoring Methods

The performance measurement is done by defining so called *monitors*. A monitor is initialized by giving it a name. A monitor can be started and stopped. It is straightforward to put the start of a monitor in the beginning of a process or sub-process and the stop at the end of it. The time difference between the start and the stop of the monitor than will be set against the already gathered time intervals of this process.

5.4.1.2 Analyzing the Measurements

JAMon also comes with a report page, where all of the processes and sub-processes are listed with their average execution time, the number of calls of this process, the maximum as well as the minimum value.

Different monitors can be set in different steps of a method, so that the report shows where the time consuming parts are situated.

5.4.2 Performance Measurements

For a detailed list of the performance measurements by the different monitors, please see the appendix.

Times of activities of particular interest will be given when discussing them in the following.

5.5 Identified Bottlenecks

In general both main activities, disregarding the initialization, split up the time for the whole save process among each other.

Various time measurements have resulted in an average time to complete for the “commit changes” activity of 4.865ms, and for the “change object list” activity of 9.290ms.

5.5.1 Commit Changes Activity

The implementation of the activity is in general matching the suggestions for an optimal approach that were made in the fourth paragraph.

5.5.1.1 Granularity

The granularity that was chosen in the implementation corresponds to the number of objects. Neither is an object further divided into parts, nor are objects aggregated into one huge object. Therefore the optimal solution is already implemented.

The average time that was spent for executing an “insert” operation on an object was 2.703ms, of an “update” operation was 4.404ms and of a “delete” operation 3.485ms.

5.5.1.2 Checking objects for change

What is implemented as well in the way, as it was proposed for the general model of a content management system, is a strategy to check objects for changes. Unnecessary communication is avoided by the checking a certain flag value of every object for information about the change that might have been happened to that object. Using the flag assures that the time t_{check} spent on checking the object is kept as small as possible. The here used strategy goes even one step further than in the model proposed. The flag does not only tell “that” an object was changed, but also tells about “what” change occurred – if it was an insert, an update or a delete. This speeds up the processing of the object in the further steps.

5.5.1.3 Archiving

As already mentioned during the description of the activity, the archiving method is called in each of the operations “update” and “delete”. In average this method takes 2.686ms per call. Taking the average times of 4.404ms and 3.485ms into account, this is more than half of the time that is spent for archiving the object. Thus the rule of every object being archived slows down the whole process significantly.

5.5.2 Change Object List

In this process there cannot be found the bottleneck, as this would mean that there is a particular outstanding part in the process that slows down the whole one.

The problem here is the process as such. The iteration through every object is what slows down the process. A single iteration takes on average about 500ms, and as the list of objects can easily count up to 20 objects, the change of the workflow state costs around 10s.

5.5.2.1 List of Objects

The content model provides every single object, may it be a page or just a text block, for containing a workflow state. This is also true for some administrative objects that are only used for the editing of the object. As they are – according to the repository structure – sub-objects of the particular object, they will as well be added to the list of objects and therefore the change of the workflow state will also be executed on them.

5.5.2.2 Checking objects for change

As well as it is already done during the “commit changes” activity, also the elements of the list for the change of the workflow state could be checked, whether or not a change is needed. But this would only be a sort of second level solution. A better one would include a complete redesign on the workflow concept. More on this in the next paragraph

6. Advices to Vattenfall

6.1 Reduction of Content Objects

Due to the examinations in the fifth paragraph it was found that the “commit changes” process already implements the optimal strategy in terms of the granularity that is used and the pre-processing that is made. But there is a possibility for a reduction in processing time that is not covered by the model.

The model recommends as optimal granularity the number of objects. If there would not have been given the lower boundary, the general rule would be to have as few objects as possible. What the model does, it presumes that the number of objects in the system is set. But considering a redesign of the content object model this is not necessarily the case.

6.1.1 Advice

Therefore my advice is to analyse the advantages and the possibilities of cost reduction by the redesign of the content object model, with the aim to reduce the overall number of object, e.g., by aggregating objects with their sub-objects, to reduce the number of round-trip times needed to save changes on objects persistently.

6.1.2 Beware of Object Size

A basic feature of the model is that it presumes the processing time that is done on a single object as minor. An aggregation could change that view. Therefore the size of the aggregated object has to be carefully analysed, if it still matches the assumptions of the model.

6.1.3 Additional Benefits

Besides the possible decrease of the round-trip time of the saving process, a reduction of objects might also have additional performance advantages. As the cache offers space only to a limited number of objects, the reduction of objects might as well speed up the whole application by a more efficient use of the cache.

6.2 Redesign of Rules for Archiving

In the analysis of the previous graph the reoccurring call of the archive function was mentioned. It could be seen that the time that it consumed took more than half of the time needed for the complete “update” or “delete” process.

Bringing again the work on a webpage as an example, this webpage consists of several sub-objects, e.g., text blocks. The current archiving rule asks for an archiving of every object when it is changed – this is as well true for every change of a text block, or similar kinds of objects, although the author might have just have wanted to call the preview in an intermediate step. Thus the rule is set in a way that also unnecessary tracking of object versions is done.

The same that was true about round-trip times and granularity for changes of objects is as well true for the archiving of objects. The paradigm to only transmit and process the useful data holds here as well.

6.2.1.1 Advice

Therefore my advice is to analyse the archiving rule, with the goal to reduce the archiving of objects to a reasonable number. It must be found a basis of rules that decides which of the archiving activities are useful work and which not.

6.3 Redesign of Workflow Concept

Regarding the “change object list” activity three advices can be given.

6.3.1 Reduction of Content Objects

The first possibility to reduce the number of workflow operations that are made during the “change object list” activity was already given in the beginning of this paragraph. An aggregation of objects into bigger ones would also reduce the number of workflow objects and therefore reduce the processing time during that activity.

Further explanations have already given in the according section.

6.3.2 Shift of Workflow State Changes to the End of the Edit Activity

The second variant of speeding up the “change object list” activity would be to eliminate it from the save function and shift it instead at the end of the editing phase.

The end of the editing phase could, e.g., be the forward of the object to the quality assurance.

Handling the workflow states like this can prevent the system from performing not useful work, meaning blind setting of state that have already been set before.

6.3.2.1 Advice

Analyse the workflow model with regard to the possibility of shifting workflow operations to the end of the editing phase, in order to prevent the system from unnecessary changes in states of objects.

As the workflow is a central issue and many components are based on it, these kinds of analyses have to be done very carefully and with an eye on the fitting into the whole system

6.3.3 Introduction of Independent Workflow Object

The third possibility for a faster “change object list” activity would be the reduction of necessary iterations through objects by the reduction of workflow objects in the whole system.

The first way would be to detach the workflow state from each object and an own workflow object is created that keeps the state of a group of objects centrally. A group of objects could be all objects that belong to a webpage.

A second, similar way would be to have the root object of a group of objects hold the state for all of its sub-objects – again, a webpage object would keep the state of all its sub-objects.

6.3.3.1 Advice

Analyse the workflow model for the possibility of reduction of workflow objects in one of the previous explained ways.

7. Conclusion and Outlook

7.1 General

In summation this study showed that content management systems in general can cope with the new challenges that have to be faced due to their use in international collaboration. CMSs face the new role to be means of communication.

The study showed that the most important factors for efficiency in content management systems – the granularity and the check for changes in objects – have a defined optimal level, where it is straightforward for a system to implement them. The Vattenfall example showed that such an implementation was already chosen before the new challenges had to be faced.

Further observations could be made that time efficient processes as such are not enough to guarantee short idle times. It is of the same importance that the models and rules that decide about the use of these efficient processes are also optimized to support time efficient processes.

This study focussed very much on the collaboration of authors for editing a content object. Further investigation has to be made in terms of analysis of the workflow processes that exist in the international teams, if there will come up further requirements for changes in processes, or higher efficiency.

What should be done as well in a further step is an analysis on how changes that are suggested in this study have influence on other activities then the edit activity, especially in terms of disadvantages.

7.2 Vattenfall

The analysis of the example of the save process in the CMS at Vattenfall showed that the suggested optimal strategy was in general implemented. Instead it revealed some further problems that could also be related to efficiency in distributed systems, respectively in content management systems.

The next steps for Vattenfall should be to do the analysises that they were advised to in the previous paragraph. As the analysis of the save function showed it is still far from an optimal solution. But the changes to the system that were suggested should be done with attention, as all of them involve a whole redesign of models and rules, thereby affecting not only the activity of the example. Therefore it must be part of the further steps to analyse the opportunities and threads that the suggested changes bring to the system as a whole.

Bibliography

[VCT04]	Virtual and Collaborative Teams: Process, Technologies and Practice Susan H. Godar, 2004
[DS02]	Distributed Systems, Principles and Paradigms Andrew S. Tanenbaum, Maarten van Steen; 2002
[MT02]	Management Technologien Gerhard Versteegen; 2002
[PE04]	Principles of Economics N. Gregory Mankiw; 2004
[WCM02]	Web Content Management, A Collaborative Approach Russel Nakano; 2002
[CMP03]	Content Management in der Praxis Oliver Christ, 2003
[JUG06]	Java Application Monitor – User Guide Steve Souza, 2/2006; http://jamonapi.sourceforge.net

Appendix

- Table of time measurements