

Institute of
Applied Informatics and
Formal Description Methods
University Karlsruhe (TH)



ASSIGNMENT

„Modeling Workflow Patterns through a
Control-flow perspective using BPMN and the
BPM Modeler BizAgi“

cand. inform. Marcus Goetz

Rehberg 25
74858 Aglasterhausen
marcus-goetz@gmx.de

Referee: Prof. Dr. Detlef Seese
Prof. Dr. Rudi Studer
Adviser: Dipl.-Wi.-Ing. Hagen Buchwald

Index of Contents

1	Abstract.....	1
1.1	Preamble	1
1.2	Motivation.....	1
1.3	Business Process Management.....	2
2	BizAgi	2
2.1	The company BizAgi	2
2.2	The product BizAgi.....	3
3	Workflowpatterns.....	3
4	Implementation of the patterns.....	4
4.1	Basic Control Flow Patterns	4
4.1.1	Sequence (WCP 01).....	4
4.1.2	Parallel Split (WCP 02)	5
4.1.3	Synchronization (WCP 03).....	6
4.1.4	Exclusive Choice (WCP 04)	6
4.1.5	Simple Merge (WCP 05)	7
4.2	Advanced Branching and Synchronization Patterns.....	8
4.2.1	Multi-Choice (WCP 06)	8
4.2.2	Structured Synchronizing Merge (WCP 07)	9
4.2.3	Multi Merge (WCP 08)	10
4.2.4	Structured Discriminator (WCP 09).....	11
4.2.5	Blocking Discriminator (WCP 28).....	11
4.2.6	Cancelling Discriminator (WCP 29).....	12
4.2.7	Structured Partial Join (WCP 30).....	12
4.2.8	Blocking Partial Join (WCP 31).....	13
4.2.9	Cancelling Partial Join (WCP 32)	14
4.2.10	Generalized AND-Join (WCP 33).....	14
4.2.11	Local Synchronizing Merge (WCP 37)	15
4.2.12	General Synchronizing Merge (WCP 38)	16
4.2.13	Thread Merge (WCP 41)	16
4.2.14	Thread Split (WCP 42).....	18
4.3	Multiple Instance Patterns	20
4.3.1	Multiple Instances without Synchronization (WCP 12)	20
4.3.2	Multiple Instances with a Priori Design-Time Knowledge (WCP 13)	22
4.3.3	Multiple Instances with a Priori Run-Time Knowledge (WCP 14).....	24
4.3.4	Multiple Instances without a Priori Run-Time Knowledge (WCP 15).....	25

4.3.5	Static Partial Join for Multiple Instances (WCP 34).....	26
4.3.6	Cancelling Partial Join for Multiple Instances (WCP 35).....	26
4.3.6.1	Dynamic Partial Join for Multiple Instances (WCP 36)	27
4.4	State-based Patterns.....	27
4.4.1	Deferred Choice (WCP 16)	27
4.4.2	Interleaved Parallel Routing (WCP 17).....	28
4.4.3	Milestone (WCP 18).....	28
4.4.4	Critical Section (WCP 39)	29
4.4.5	Interleaved Routing (WCP 40)	29
4.5	Cancellation and Force Completion Patterns.....	30
4.5.1	Cancel Task (WCP 19).....	30
4.5.2	Cancel Case (WCP 20)	31
4.5.3	Cancel Region (WCP 25)	32
4.5.4	Cancel Multiple Instance Activity (WCP 26)	33
4.5.5	Complete Multiple Instance Activity (WCP 27)	34
4.6	Iteration Patterns.....	34
4.6.1	Arbitrary Cycles (WCP 10).....	34
4.6.2	Structured Loop (WCP 21).....	35
4.6.2.1	Normal.....	35
4.6.2.2	Post.....	36
4.6.2.3	Pre.....	37
4.6.3	Recursion (WCP 22)	39
4.7	Termination Patterns.....	39
4.7.1	Implicit Termination (WCP 11).....	39
4.7.2	Explicit Termination (WCP 43).....	40
4.8	Trigger Patterns	41
4.8.1	Transient Trigger (WCP 23)	41
4.8.2	Persistent Trigger (WCP 24)	41
5	Conclusion.....	42
5.1	Overview about the results.....	42
5.1.1	Basic Control Flow Patterns	42
5.1.2	Advanced Branching and Synchronisation Patterns.....	43
5.1.3	Multiple Instance Patterns	43
5.1.4	State-based Patterns.....	44
5.1.5	Cancellation and Force Completion Patterns	44
5.1.6	Iteration Patterns	44

5.1.7	Termination Patterns.....	44
5.1.8	Trigger Patterns	44
5.2	Wrap Up.....	44
5.3	Forecast	45
6	List of Literature.....	46
7	List of Illustrations.....	46

1 Abstract

This assignment is about Workflowpatterns and the Business Process Management Software Solutions of the company BizAgi. In the first chapter you will get a brief introduction to the area of Business Process Management. Afterwards I will inform you a bit about the company BizAgi and their product BizAgi. After that chapter you will find more information about Workflowpatterns in chapter three. The main part of my work is chapter 4 where I am trying to realize the different workflowpatterns in BizAgi. Concluding to this is my conclusion and forecast.

I assert that I have written this assignment completely on my own. All used sources of information are marked or quoted in the text.

1.1 Preamble

First of all I would like to thank everybody who supported me during the work on my assignment.

A special thanks goes to Professor Dr. Detlef Seese and Professor Dr. Rudi Studer for being my two mentors and supporters for this assignment.

I also would like to thank Dipl.-Wi.-Ing. Hagen Buchwald and Dipl.-Wi.-Ing. Oliver Schöll for supporting me during my practical studies in this assignment. I've been able to profit a lot from their huge knowledge in the area of Business Process Management and their good skills in the BizAgi BPM Suite.

Another person I would like to thank is Gustavo Gomez, the CEO of BizAgi. He made this assignment possible supplying me with a free-of-charge license of their BizAgi BPM Suite.

Additionally I would like to thank Marcel Manser who was my direct contact person at BizAgi. He always tried to help me with every issue I had and he was a great support for my product evaluation.

1.2 Motivation

This assignment is about the BPM Suite of the company BizAgi (which also is called BizAgi) and the 43 workflow patterns (control-flow perspective) of Prof. van der Aalst. What motivated me for this assignment was the following:

Prof. van der Aalst has invented 43 Patterns for Workflows (out of a control flow perspective) which describe different flows in Business Processes and Workflows which recur very often in day-to-day business problems. He has grouped those patterns in different categories. I was curious to see if a particular BPM Tool was able to map all those patterns – and if not, to find out why. Since the BPM Modeler and the BPM Suite of BizAgi is one of the market leader application software [BizAgi09] I wanted to find out whether this ranking was appropriate or not – so I decided to evaluate the BizAgi BPM Tools.

1.3 Business Process Management

Business Process Management is a Management discipline which contains the Evaluation, Development, Documentation and Optimization of Business Processes. A Business Process describes a series of single activities which are executed step by step trying to reach a predefined business goal. In contrast to a project a process can be traversed several times. A Business process can be part of another business process or it can contain other business processes. A central question in the field of Business Process Management is “Who does what, when, how and with the help of what”? [Becker09][Ehlers06] The goal of business process management is to use existing information about the own business processes for aligning the strategy with the customer resulting in a better way of achieving the company goals. Essentially for this is to know your own business processes, to optimize them, to do documentation about them (maybe law dictates this), to reduce costs, to be as flexible as possible to transform exceptions into rules and to define clear interfaces between different business processes. In achieving all these goals a so called business process system or business process suite (sometimes also called workflow management system or WFMC) can support the company tremendously. Such a system defines, creates and manages the execution of business processes through the use of software, running on one or more business process engines, which is able to interpret the process definition, interact with the participants and, where required, invoke the use of IT tools and applications [Uslar04]. Deploying such a system should especially achieve several goals:

- Decrement of processing time
- Controlled flow of data and documents
- Automation of activities / tasks
- Optimization of time and resource usage
- Abolishment of modal fragmentation
- Creation of transparent reusable business processes
- Systematization of activities
- Reduction of costs caused by processes

Despite the fact that a good business process system can achieve all these goals there are also some disadvantages arising with the introduction of such a system :

- High initial costs [Uslar04]
- High initial expenditure of time thru the need for education [Uslar04]
- The company must know its processes [Uslar04]
- Employees should accept the system [Uslar04]

2 BizAgi

In the following chapters you will learn more about the company BizAgi and its product – the BizAgi BPM Modeler / Suite.

2.1 The company BizAgi

The company BizAgi was founded in 1989. Its shareholders and executive all have

the same objective: "Offer organizations agile and flexible solutions that enable them to achieve greater productivity and profitability in a modern and changeable business environment" [BizAgi09].

With a strong foundation in the main European and Latin American financial markets and more than 20 years of study and analysis of organizational processes, BizAgi has been able to consolidate itself as the leading BPM solution for automating and improving human processes using the minimum amount of programming. The company and its employees are 100 percent dedicated to the continuous improvement of the performance of the processes of their customers, trying to make them more productive, efficient and agile compared to their competitors. The principle of the company could be described as the following "The Process IS the Application" [BizAgi09].

The company's Corporate Headquarter is located in the UK, St Mary's Court - The Broadway - Amersham, Buckinghamshire HP7 0UT. For more information please refer to www.bizagi.com or mail to marketing@bizagi.com [BizAgi09].

2.2 The product BizAgi

BizAgi is the BPM solution enabling you and your organization to design, model, integrate, automate, and monitor your business processes through a graphic environment. It is the quickest and most efficient way to achieve continuous improvement of your processes.

BizAgi has earned the best opinions from BPM researchers and analysts worldwide, thus positioning the company as a leader in this field. According to Enix (one of the main European BPM analysts): BizAgi offers "a fundamentally new level of capability not yet seen in other approaches"; likewise, Gartner, the worldwide renowned IT research organization considers BizAgi as a "visionary" solution in their BPMPP quadrant. BPMG, an important worldwide analyst in the subject, recognizes BizAgi as "one of the leaders in the elite group of BPM products for the most demanding challenges". SODAN, an organization dedicated to IT research and assessment in the UK and Europe, rates us as "the most business oriented BPM system reviewed so far". BizAgi's clients, worldwide recognition of the solutions, the support from BizAgi's partners, excellent teams of management and professional experts have made BizAgi the global leader in BPM.

3 Workflowpatterns

In the early beginning for Workflowmanagementsystems and Business Process Management Systems the systems were distributed without any conceptual basis which lead to huge disaffection at customer's side. The fast changing processes in enterprises were too complex for the available systems – they simply couldn't map them correctly. A so called Mobile Approach tried to classify requirements for those systems. This classification has been focused on a control flow perspective. Since the Mobile Approach just had a moderate success Professor van der Aalst and Professor Hofstede transferred the well known Design Patterns onto control structures in the year 1999 [Uslar04]. They both said that their work is the scientific answer to the concepts of company consultants to the topic

Workflowmanagement. With their work the to scientists are trying to systematically describe business process requirements. The identified patterns – the so called workflowpatterns – appear very often in real life business processes. With these patterns consultants are able to measure whether a system is able to map the required patterns or not – so the patterns are a qualitative way to measure the possibilities of a Workflowsystem. For more information please refer to [Uslar04].

4 Implementation of the patterns

In the following chapters the different patterns and their implementation in BizAgi is described. For each pattern there's a general description. Furthermore there is a distinct Use Case from the area of Cluster Management for every pattern helping to create a much more realistic implementation scenario. Additionally these Use Cases should create a connection between theoretic science analysis and practical business work.

4.1 Basic Control Flow Patterns

This group of patterns captures elementary aspects of process control.

4.1.1 Sequence (WCP 01)

General Description

Several tasks of a process are executed after each other. A task is enabled after the preceding task has finished and before the proceeding task has been started [Aalst09].

Use Case

An applicant reads the AGB and afterwards he or she decides to become a member.

Implementation

Since a sequence is standard functionality in BPMN BizAgi is naturally able to implement this by connecting two or more activities with a Sequence Flow arrow.

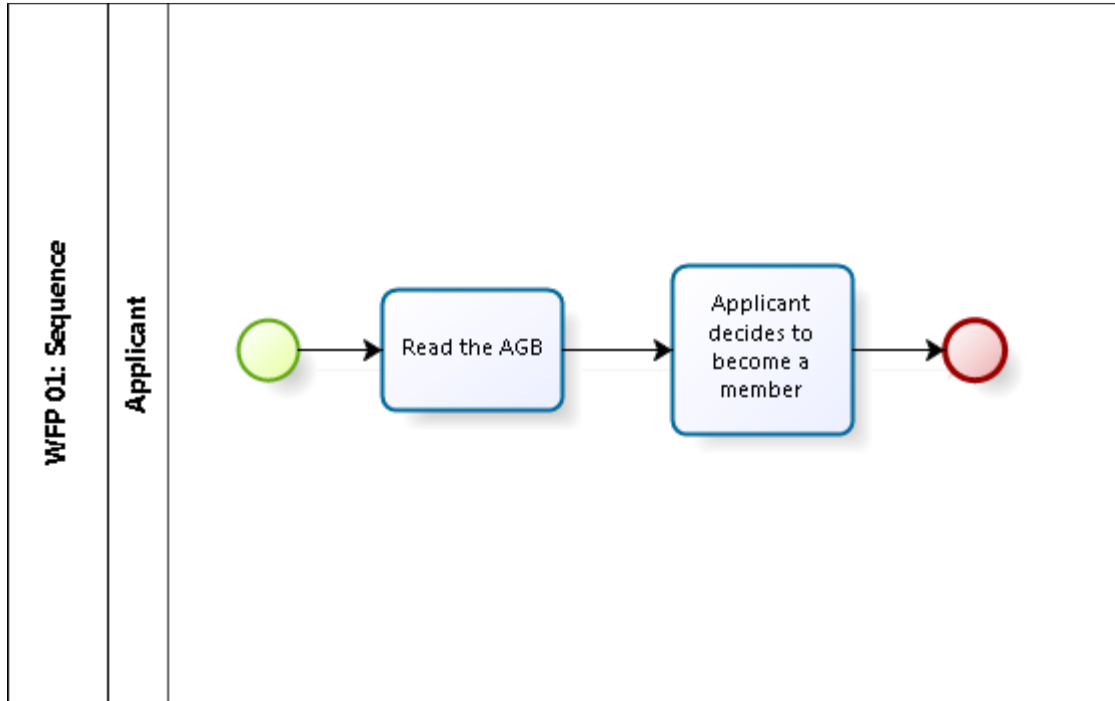


Illustration 1: Implementation of the "Sequence" pattern

4.1.2 Parallel Split (WCP 02)

General Description

A Parallel Split is a distinct point in a business process where a single branch is divided into two or more parallel branches which are executed concurrently [Aalst09].

Use Case

An Account manager has advertised a new member. Afterwards the manager has to do two things: he has to request an information brochure from the media department and he has to request a goody from the marketing department.

Implementation

Since a parallel split is standard functionality in BPMN BizAgi is naturally able to implement this by using a so called Parallel Gateway.

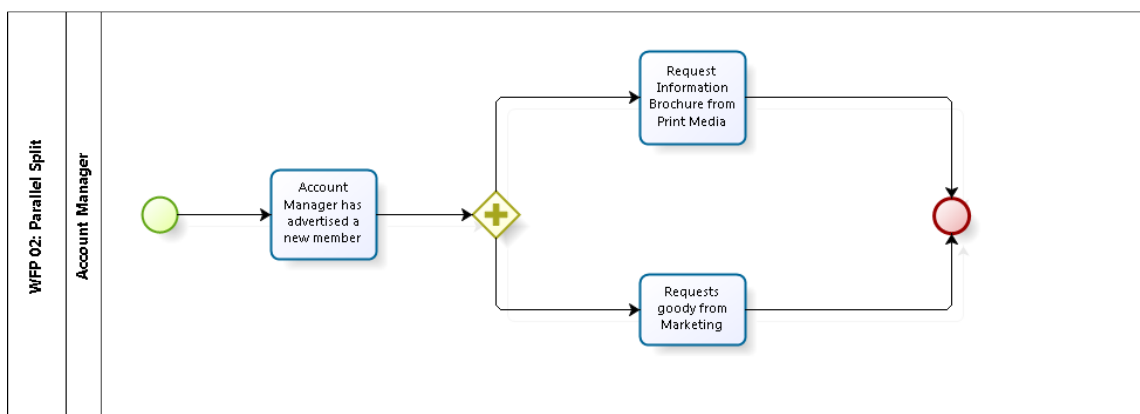


Illustration 2: Implementation of the "Parallel Split" pattern

4.1.3 Synchronization (WCP 03)

General Description

A Synchronization is a distinct point in a business process where two or more different branches are merged into one single branch. It is called Synchronization because it expects all merged branches to be completed before going ahead with the process [Aalst09].

Use Case

The Use case for this pattern is similar to the one presented for pattern number two. An Account manager has advertised a new member. Afterwards the manager has to do three things: he has to request an information brochure from the media department, he has to request a goody from the marketing department and he has to forward the clients core data to the office assistant. As soon as possible the media department sends the information brochure to the office assistant and also the marketing department will send the goody to the office assistant. The office assistant now waits for the information brochure, for the goody and for the account creation till she sends all the stuff to the customer.

Implementation

Since a Synchronization is standard functionality in BPMN BizAgi is naturally able to implement this by using a so called Parallel Gateway for merging branches.

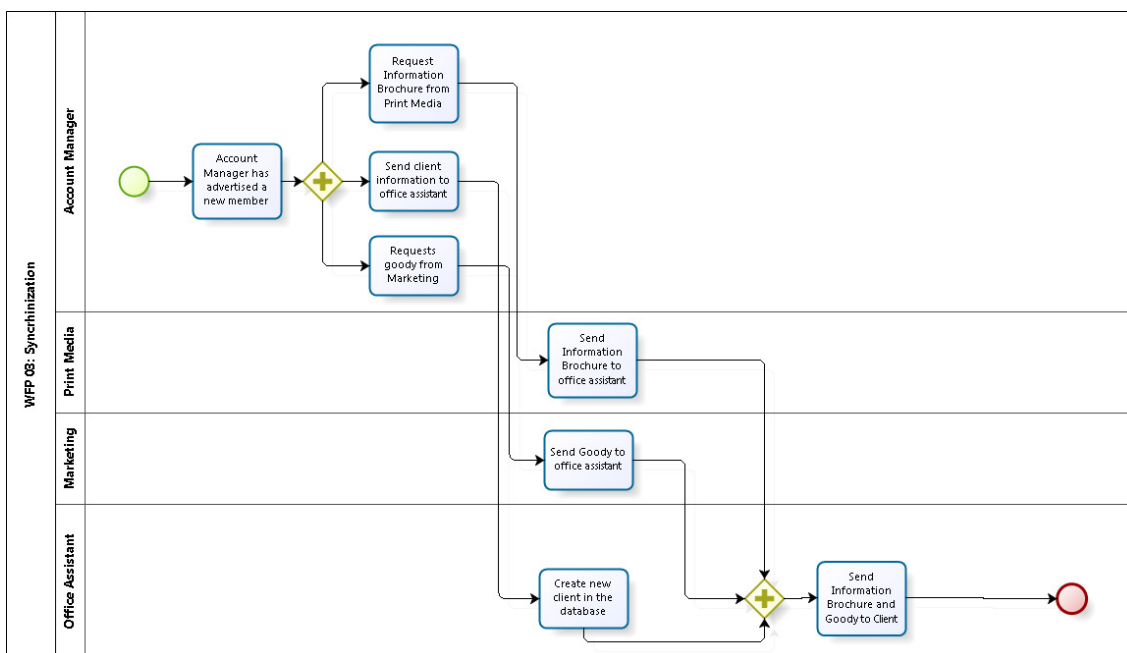


Illustration 3: Implementation of the "Synchronization" pattern

4.1.4 Exclusive Choice (WCP 04)

General Description

An Exclusive Choice is a distinct point in a business process where a branch is divided into two or more branches enabling just one of these several branches [Aalst09].

Use Case

An applicant has read the AGB and decides now to become a member. He has now to decide whether he wants a sponsor contract or an ordinary membership application. Doing both is not possible.

Implementation

Since an Exclusive Choice is standard functionality in BPMN BizAgi is naturally able to implement this by using a so called Exclusive Gateway. Editing the conditions for the gateway by defining expressions is simply done by clicking on the gateway with the right button of the computer mouse and switching to the tab Advanced.

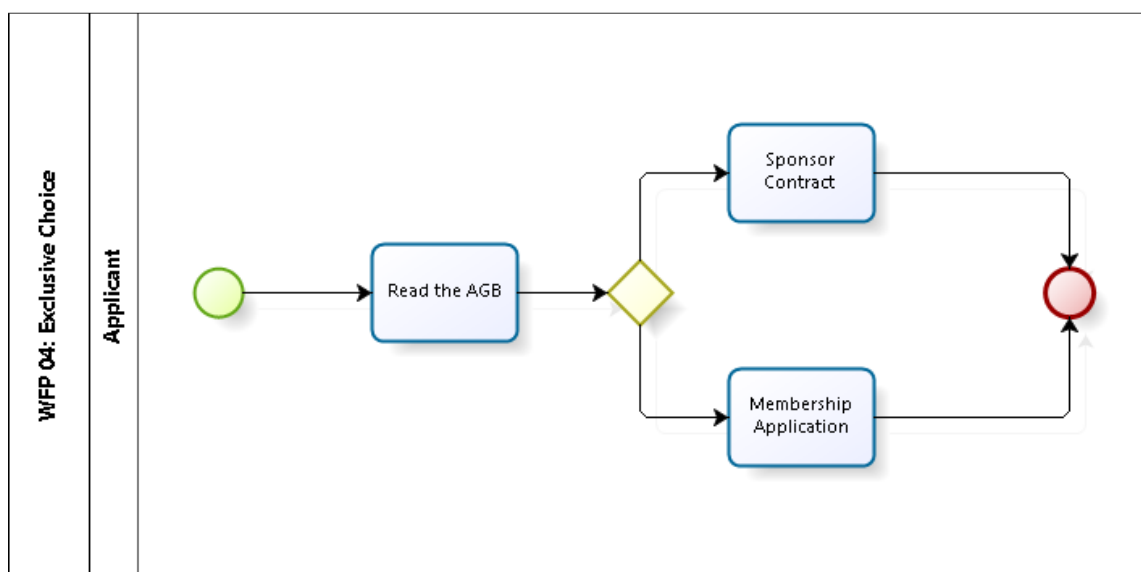


Illustration 4: Implementation of the "Exclusive Choice" pattern

4.1.5 Simple Merge (WCP 05)

General Description

A Simple Merge is a distinct point in a business process where two or more branches are merged into one single branch. Each incoming branch then activates the subsequent branch [Aalst09].

Use Case

The office assistant wants to make an appointment for an interview with an applicant by sending a proposal via e-mail. The applicant can then accept the appointment via phone and/or via e-mail. The office assistant will then define the appointment as fixed.

Implementation

Since Simple Merge is standard functionality in BPMN BizAgi is naturally able to implement this by simply connecting two or more process flow arrows in one single activity.

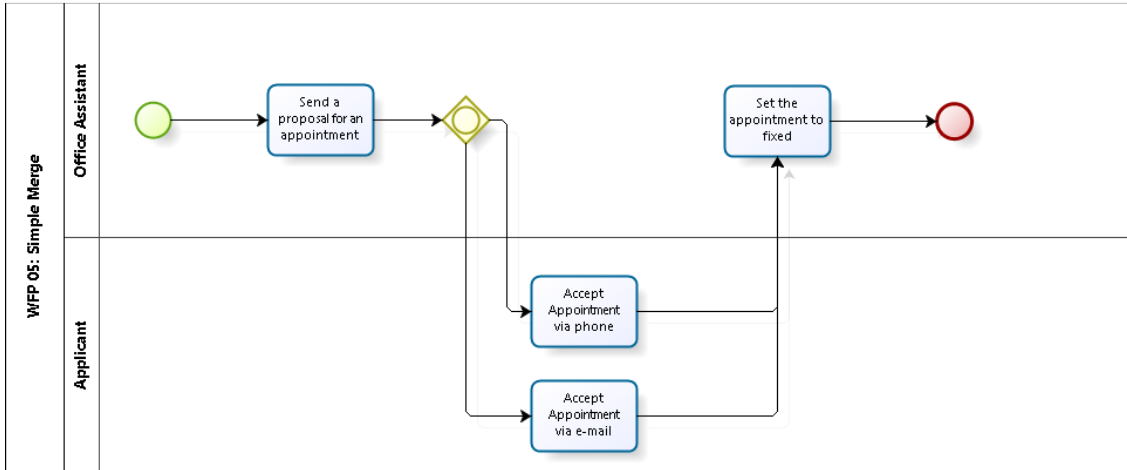


Illustration 5: Implementation of the "Simple Merge" pattern

4.2 Advanced Branching and Synchronization Patterns

This group presents several patterns which characterize more complex branching and merging concepts which arise in business processes.

4.2.1 Multi-Choice (WCP 06)

General Description

The Multi-Choice pattern describes the splitting of one single branch in two or more parallel branches. As soon as the incoming branch is enabled the thread is immediately passed to one or more of the outgoing branches. Which outgoing branches are selected depends thereby on an internal mechanism or individual decision [Aalst09].

Use Case

An applicant has read the AGB and decides now to become a member. He has now to decide whether he wants a sponsor contract or an ordinary membership application. Doing both is also possible.

Implementation

A realization of the Multi-Choice pattern is available within BizAgi. The realization is done via a so called Inclusive Gateway which activates one or more outgoing branches of the gateway.

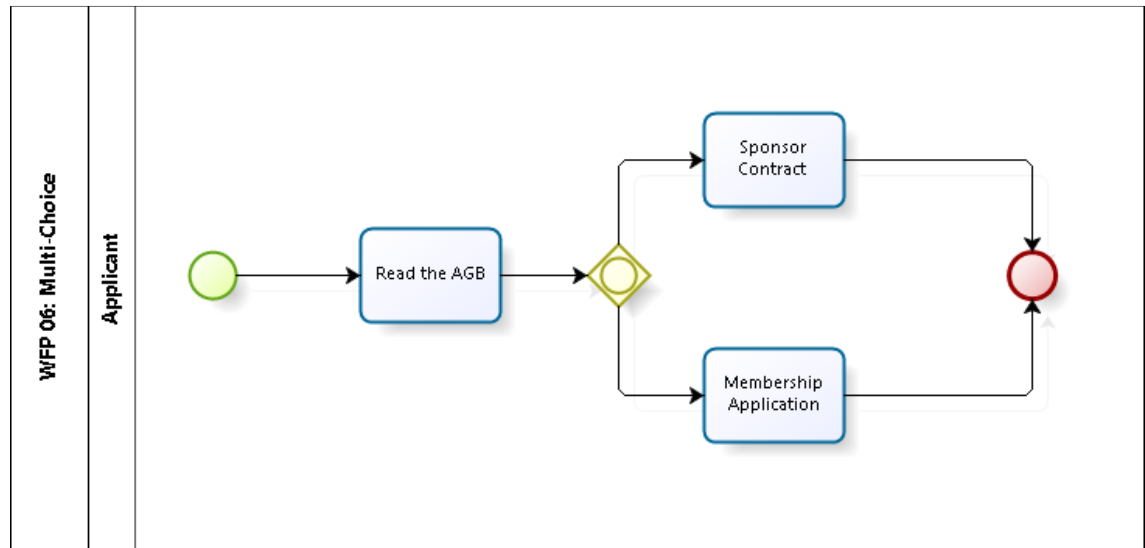


Illustration 6: Implementation of the "Multi-Choice" pattern

4.2.2 Structured Synchronizing Merge (WCP 07)

General Description

The Structured Synchronizing Merge pattern describes the merger of two or more branches which have been split at a uniquely identifiable point of time earlier in the business process into one single branch. The thread of control is passed to the proceeding branch as soon as all incoming branches have been enabled. This pattern occurs in a structured context so it is essential that there is a Single Multi-Choice (refer to page 8) earlier in the Business Process and the Structured Synchronizing Merge Pattern has to merge all of the branches emanating from the Multi-Choice [Aalst09].

Use Case

This Use Case is based on the Use Case for the Multi-Choice Pattern (refer to page 8). After an applicant has fill out the membership application and/or the sponsor contract (Multi-Choice) the documents are sent to an Employee for revision. The employee has then to revise the membership application or the sponsor contract. If there are both documents he has to do the revision for both of them for work completion.

Implementation

A realization of the Structured Synchronizing Merge pattern is available within BizAgi. The realization is done via a so called Inclusive Gateway for merging two or more branches.

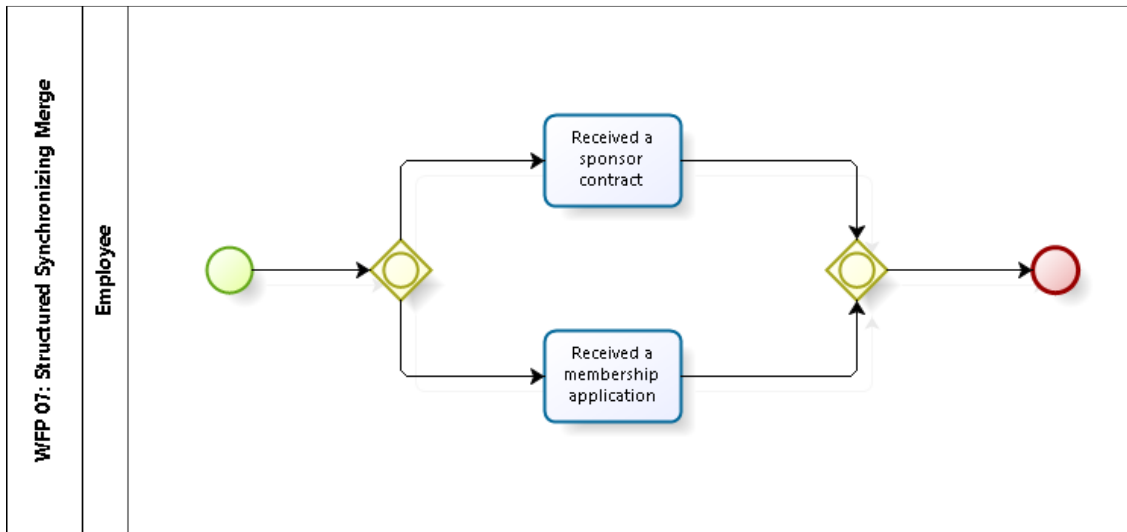


Illustration 7: Implementation of the "Structured Synchronizing Merge" pattern

4.2.3 Multi Merge (WCP 08)

General Description

The Multi Merge pattern describes the convergence two or more parallel branches into one single branch. Special about this pattern is that each enablement of an incoming branch results in the activation of the proceeding activity within the business process [Aalst09].

Use Case

This Use Case is similar to the one presented for the Structured Synchronizing Merge (refer to page 9). The Employee receives one or even two documents and then does a revision for each of them.

Implementation

A realization of the Multi Merge pattern is available within BizAgi. It is done by simply connecting the two or more branches to a proceeding single activity by the use of Business Flow Arrows.

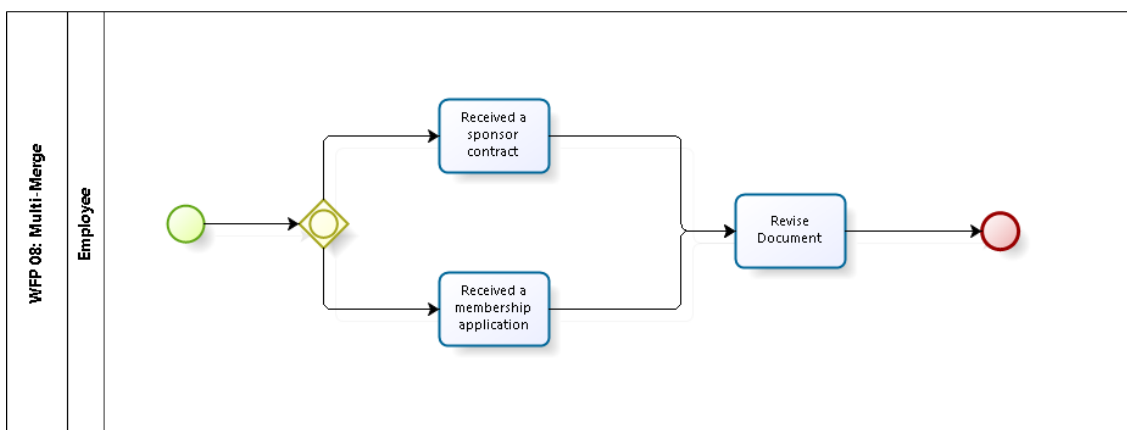


Illustration 8: Implementation of the "Multi Merge" pattern

4.2.4 Structured Discriminator (WCP 09)

General Description

The Structured Discriminator pattern describes the merger of two or more branches into one single branch with a corresponding split beforehand somewhere in the business process. The thread of control is passed to the proceeding activity as soon as one incoming branch has been enabled regardless of the progress of the other incoming branches. This pattern occurs in a structured context so it is essential that there is one single Parallel Split construct somewhere earlier in the business process with which the Structured Discriminator is associated. The Structured Discriminator must merge all of the branches coming from the split [Aalst09].

Use Case

The marketing department has just one advertising space available on an event. The marketing department then tries to find a customer for this ad space on two different channels - the internet and the daily newspaper. As soon as one customer books the advertising space it doesn't matter what the other customers in the two advertising channels do. The contract gets signed and the ad space is booked from the customer.

Implementation

Although BPMN supports this pattern according to the BPMN definition it is unclear how the incoming Condition express on the COMPLEX-join gateway should be specified. However the described Use Case can be modeled through a work-around using several splits, merges and loops.

4.2.5 Blocking Discriminator (WCP 28)

General Description

The Blocking Discriminator Pattern describes the fusion of two or more individual branches into one single branch with corresponding antecedent divergences. The outgoing branch of the Blocking Discriminator is activated as soon as the first incoming branch has been enabled regardless of the state of the other branches. The pattern resets itself as soon as all active incoming branches have been enabled once for the same process instance. All subsequent enablements of incoming branches are blocking until reset [Aalst09].

Use Case

The Use Case for the Blocking Discriminator is similar to the one presented for the Structured Discriminator (WCP 09, please refer to page 11). The marketing department has again just one advertising space available on an event and they try to find a customer for this via internet and daily newspaper. The difference in this Use Case is that after the successful booking of the advertising space the marketing department has to wait until the customer has used both channels (internet and newspaper). This will help the marketing department in doing detailed analysis about the efficiency of the different channels.

Implementation

Although BPMN supports this pattern according to the BPMN definition it is unclear how the incoming Condition express on the COMPLEX-join gateway should be specified. However the described Use Case can be modeled through a work-around using several splits, merges and loops.

4.2.6 Cancelling Discriminator (WCP 29)

General Description

The Cancelling Discriminator Pattern describes merging two or more branches into one single subsequent branch with corresponding divergences beforehand. The thread of control is passed to the outgoing branch as soon as the first incoming branch has been enabled. All other incoming branches are canceled regardless of their current state [Aalst09].

Use Case

The Use Case for the Cancelling Discriminator is similar to the one presented for the Structured Discriminator (WCP 09, please refer to page 11). The marketing department has again just one advertising space available on an event and they try to find a customer for this via internet and daily newspaper. The difference in this Use Case is that after the reservation of the ad space trough one of the two channels the other channel will be closed so that there is no way of doing another reservation trough the closed channel.

Implementation

This pattern / Use Case is available within BizAgi. It is done by including the incoming branches and the OR-join in a sub process which passes control to the proceeding activity once the first branch has completed. Additionally there is an error type intermediate event required for cancelling the remaining activities.

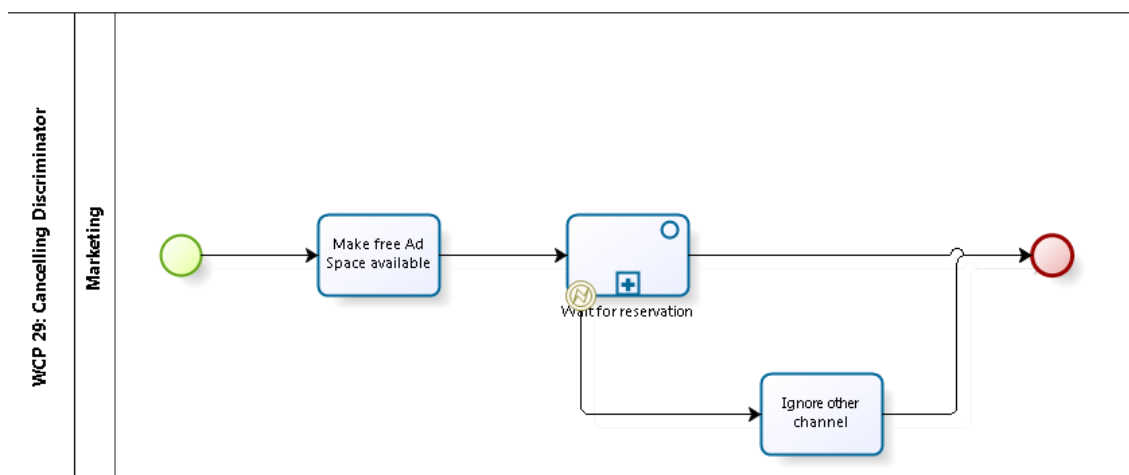


Illustration 9: Implementation of the "Cancelling Discriminator" pattern

4.2.7 Structured Partial Join (WCP 30)

General Description

The Structured Partial Join Pattern is about the convergence of two or more (let's

pretend m) branches into one single subsequent branch with corresponding splits somewhere earlier in the business process. The thread of control is passed to the subsequent branch of the pattern as soon as n incoming branches (n is less than m) are enabled. All other k incoming branches afterwards ($k = m - n$) do not result in the thread of control being passed on. The pattern resets itself as soon as all active incoming branches have been enabled. The pattern occurs in a structured context which means that there must be a single Parallel Split pattern somewhere earlier in the business process and the Join must merge all of the branches emanating from the Parallel Split [Aalst09].

Use Case

This Use Case is again the area of booking advertising spaces. The marketing department now has three different channels for promoting an ad space - daily newspaper, internet and phone. The phone is used by the employees of marketing trying to call former customers. As soon as the employees have offers through two of the three channels they can decide who to choose. The third channel is then irrelevant.

Implementation

Although BPMN supports this pattern according to the BPMN definition it is unclear how the incoming Condition express on the COMPLEX-join gateway should be specified. However the described Use Case can be modeled in BizAgi with a work-around using several splits, merges and loops.

4.2.8 Blocking Partial Join (WCP 31)

General Description

The Blocking Partial Join Pattern describes the fusion of two or more branches (let's pretend again m) into one single following branch with corresponding divergences somewhere earlier in the business process. The thread of control is passed to the outgoing branch as soon as n (n is less than m) incoming branches has been enabled. The join resets itself as soon as all active incoming branches have been enabled once for the same process instance. All other subsequent enabled activities are blocked until the join is reset [Aalst09].

Use Case

This Use Case is similar to the one presented for WCP 28 (Blocking Discriminator, please refer to page 11). The difference is that there are three channels (phone, daily newspaper and internet) and the decision is made as soon as there are offers on two of the three channels.

Implementation

Although BPMN supports this pattern according to the BPMN definition it is unclear how the incoming Condition express on the COMPLEX-join gateway should be specified. However the described Use Case can be modeled in BizAgi with a work-around using several splits, merges and loops.

4.2.9 Cancelling Partial Join (WCP 32)

General Description

This pattern is a adaption of WCP 31 (Blocking Partial Join) described beforehand. The Cancelling Partial Join also describes the convergence of two or more branches (let's pretend m again) into one single outgoing branch with corresponding divergences beforehand in the business process. The thread of control is passed to the outgoing branch of the join as soon as n (with n less than m) incoming branches have been enabled. As soon as the outgoing branch is enabled all other incoming branches are cancelled regardless of their current state and the join is reset [Aalst09].

Use Case

This Use Case is similar to the one presented for WCP 29 (Cancelling Discriminator, please refer to page 12). The difference is that there are three channels (phone, daily newspaper and internet) and the decision is made as soon as there are offers on two of the three channels.

Implementation

Although BPMN supports this pattern according to the BPMN definition it is unclear how the incoming Condition express on the COMPLEX-join gateway should be specified. However the described Use Case can be modeled in BizAgi with a work-around using several splits, merges and loops.

4.2.10 Generalized AND-Join (WCP 33)

General Description

The Generalized AND-Join pattern describes merging two or more branches into one single outgoing branch. The thread of control is passed by as soon as all incoming branches have been enabled [Aalst09].

Use Case

The Use Case for this pattern is the same as for pattern number 03 (Synchronization, please refer to page 6). An Account manager has advertised a new member. Afterwards the manager has to do three things: he has to request an information brochure from the media department, he has to request a goody from the marketing department and he has to forward the clients core data to the office assistant. As soon as possible the media department sends the information brochure to the office assistant and also the marketing department will send the goody to the office assistant. The office assistant now waits for the information brochure, for the goody and for the account creation till she sends all the stuff to the customer.

Implementation

You can easily realize this pattern in BizAgi by using a so called Parallel Gateway for merging branches.

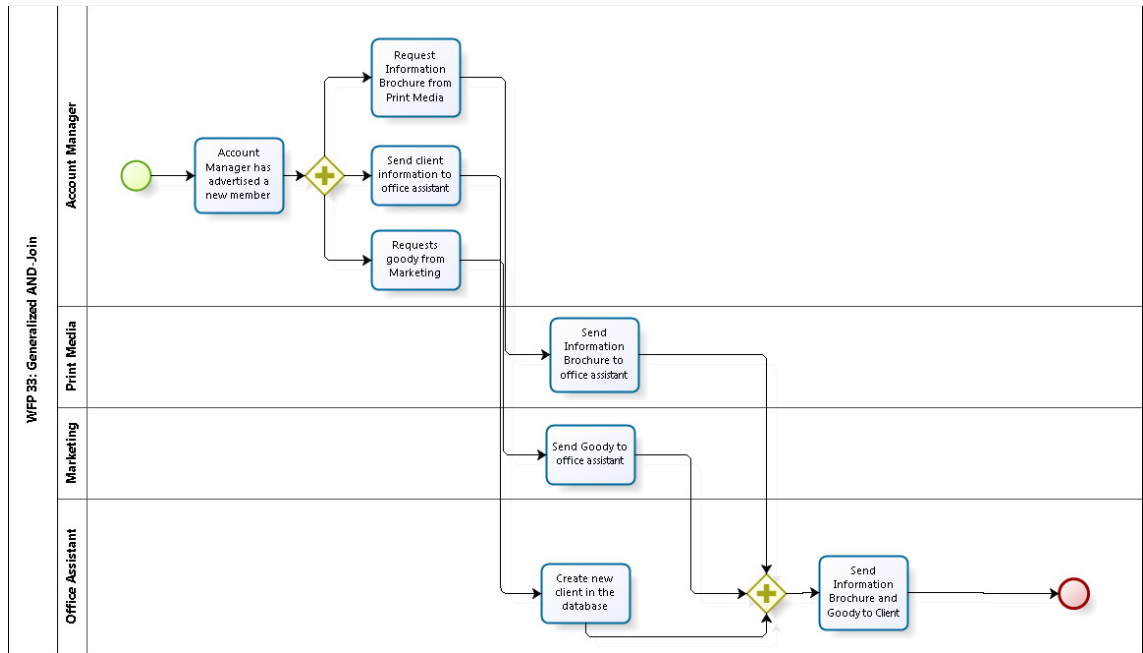


Illustration 10: Implementation of the "Generalized AND-Join" pattern

4.2.11 Local Synchronizing Merge (WCP 37)

General Description

This pattern describes the convergence of two or more single branches which split earlier in the business process into one single outgoing branch. The thread of control is passed to the subsequent branch as soon as each active incoming branch has been enabled. Determination of how many branches require synchronization is made on the basis of information locally available to the merge construct. This may be communicated directly to the merge by the preceding diverging construct or alternatively it can be determined on the basis of local data such as the threads of control arriving at the merge [Aalst09].

Use Case

The Use Case for the Local Synchronizing Merge is the following: the marketing department has three ad spaces available for their customers - the promotion of these three spaces is done via phone, internet and daily newspaper. If the marketing department decides to promote the spaces via internet or phone both channels need to be synced. If marketing decides to use the daily newspaper as promotion channel they can decide later whether they have to sync the channels or not. So it is possible that there is no valid offer through the newspaper channel.

Implementation

This pattern is available within BizAgi by using an Inclusive Gateway for merging. Important for this pattern is, that the gateway is used in a structured context.

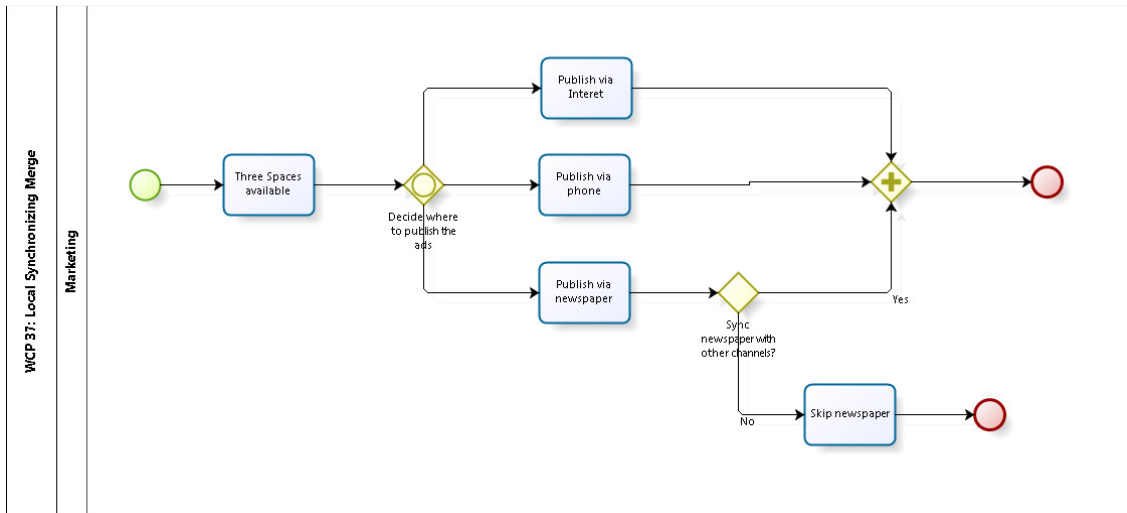


Illustration 11: Implementation of the "Local Synchronizing Merge" pattern

4.2.12 General Synchronizing Merge (WCP 38)

General Description

The General Synchronizing Merge pattern describes merging two or more branches which diverged earlier in the business process into one single outgoing branch. The thread of control is passed to the subsequent branch as soon as either each active incoming branch has been enabled or as soon as it is evident that there is no other branch which hasn't yet been enabled that will be enabled in future [Aalst09].

Use Case

This Use Case is very similar to the one presented for the Local Synchronizing Merge Pattern (WCP 37, please refer to page 15) before. The difference is that the marketing department is able to initiate the newspaper channel several times.

Implementation

This pattern is not directly supported within BizAgi because there is no way of assessing whether an Inclusive gateway should fire based on a complete state analysis of the process instance. However the describes Use Case can be modeled via a work-around using several splits, merges and loops.

4.2.13 Thread Merge (WCP 41)

General Description

The Thread Merge pattern describes a particular point of time in a business process where a distinct number of execution threads in single branches within the same process instance should be merged into one single thread of execution [Aalst09].

Use Case

The pattern for this Use Case is about testing several applicants. As soon as twenty

applicants confirmed their participation in the test, the Test Coordinator can start with the preparation.

Implementation

You can implement this Use Case / pattern in BizAgi by adding a Multiple Instances Activity and afterwards a normal activity.

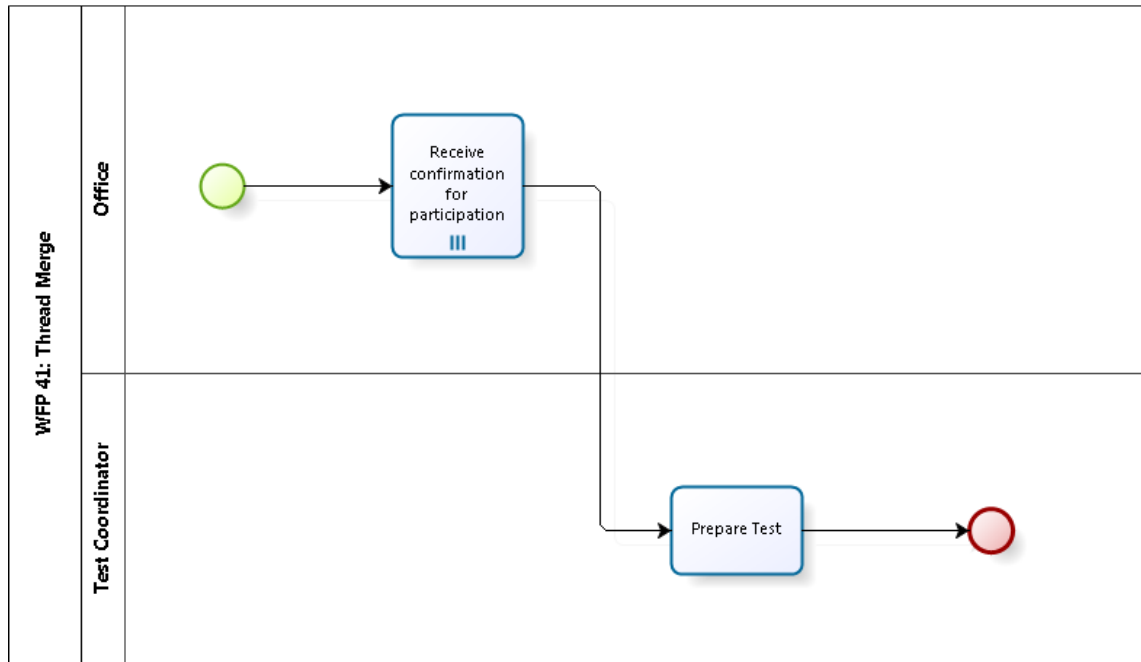


Illustration 12: Implementation of the "Thread Merge" pattern

Simply Change the Start Quantity of the normal activity to 20 or your desired value. You can edit the Start Quantity Value by doing a right click on the activity (or pressing F4) and selecting the Advanced Tab in the Element properties box.

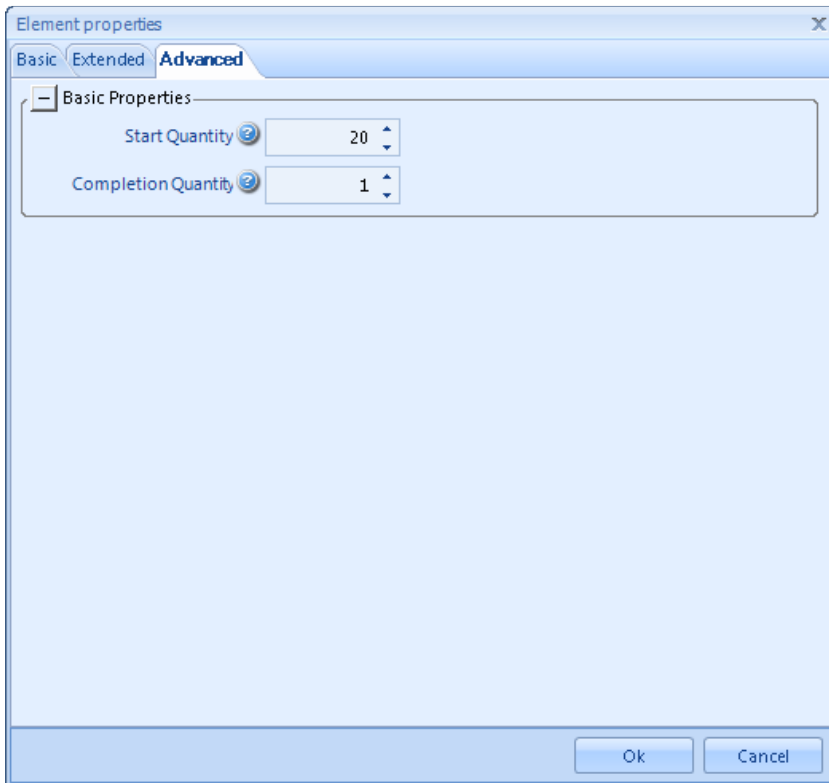


Illustration 13: Required properties settings for the "Thread Merge" pattern

4.2.14 Thread Split (WCP 42)

General Description

The Thread Split pattern describes a particular point of time in a business process where a nominated number of execution threads can be initiated in a single branch of the same process instance [Aalst09].

Use Case

The Use Case for this pattern is quasi the prosecution of the Use Case presented for WCP 41 (Thread Merge, please refer to page 16). After the test is done all twenty tests need to be revised. The way the revision is done is always the same.

Implementation

There are two ways to implement this Use Case in BizAgi.

1. You create two normal activities and change the Completion Quantity value of the second activity to 20 (or your desired value). This will cause that the Thread of control is passed to the preceding activity as soon as 20 instances

have arrived.

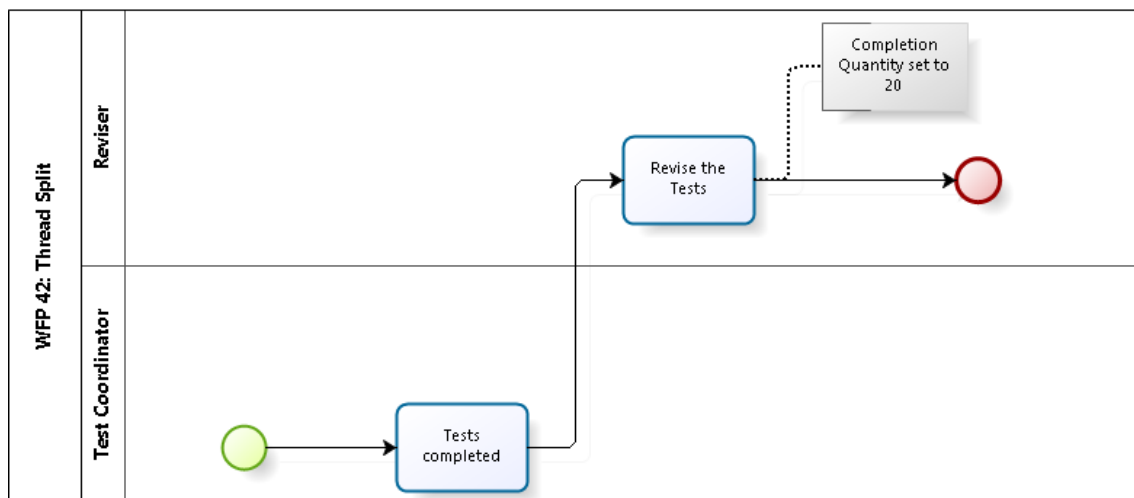


Illustration 14: Implementation of the "Thread Split" pattern using a basic activity and the "Completion Quantity" value

You can edit the Completion Quantity Value by doing a right click on the activity and switching to the Advanced Tab in the properties box.

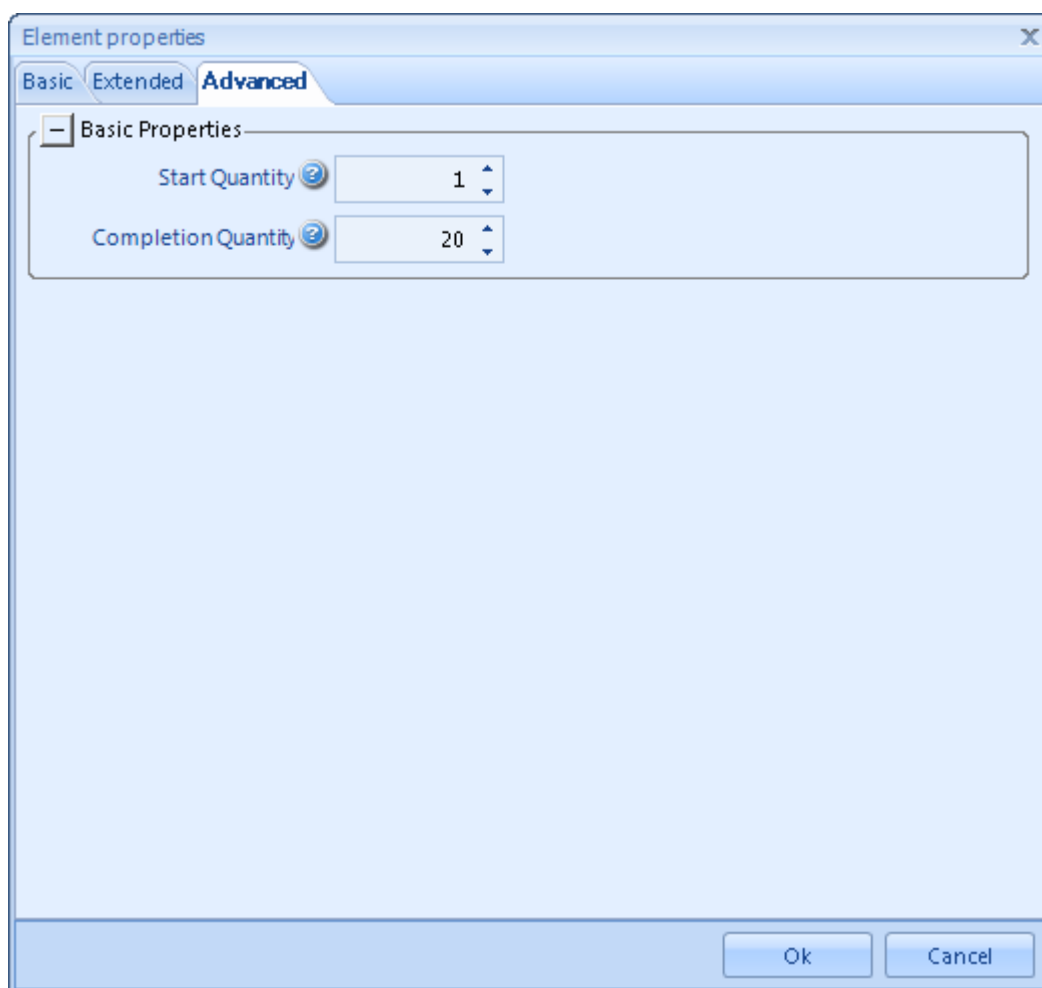


Illustration 15: Required properties settings for WCP 42

2. You can arrange the second activity as a Multiple Instances Activity defining that there are 20 different activities.

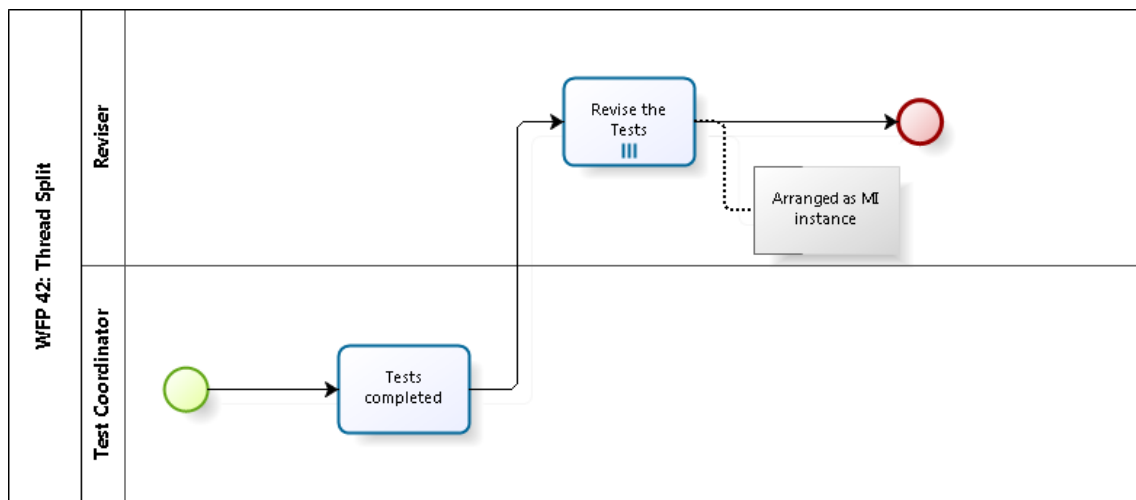


Illustration 16: Implementation of the "Thread Split" pattern using a MI activity

4.3 Multiple Instance Patterns

This group describes patterns with multiple active threads in a process flow which relate to the same activity. Multiple instances can arise through three different reasons:

1. An activity is able to initiate multiple instances of itself
2. An activity is initiated multiple times as a consequence of receiving several independent triggers for example as a part of a loop
3. Two or more activities in a process share the same implementation definition

4.3.1 Multiple Instances without Synchronization (WCP 12)

General Description

The pattern 12 describes that each multiple instance task that is created must be executed within the context of the process instance from which they were started and each of them must be executed independently from and without reference to the task that started them [Aalst09].

Use Case

Every month at the distinct point of time the IT Department changes the W-Lan Access Key and sends the new key to all employees via email. These then have to change the key manually on their local PC/laptop - the IT department is no longer involved.

Implementation

This pattern is available within the BizAgi Suite. Simply create a Multiple Instances Activity and set the MI Ordering to Parallel and the MI Flow Condition to None.

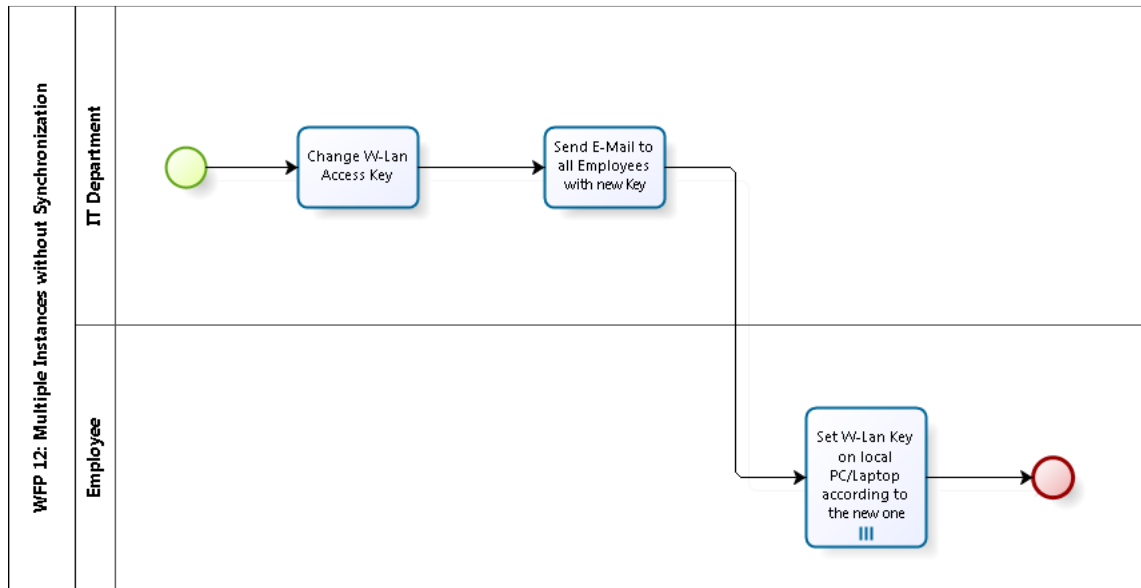


Illustration 17: Implementation of the "Multiple Instances without Synchronization" pattern

You can edit those values by doing a right click on the MI activity (or simply pressing F4 on the keyboard) and switching to the Advanced tab in the properties box.

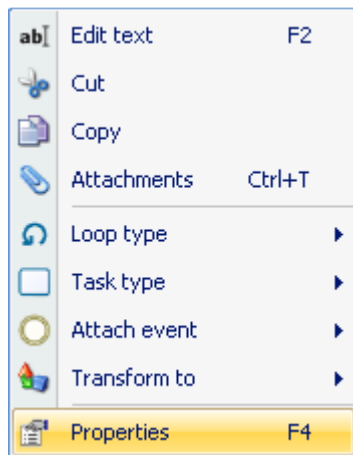


Illustration 18: Menu for the right click on an activity

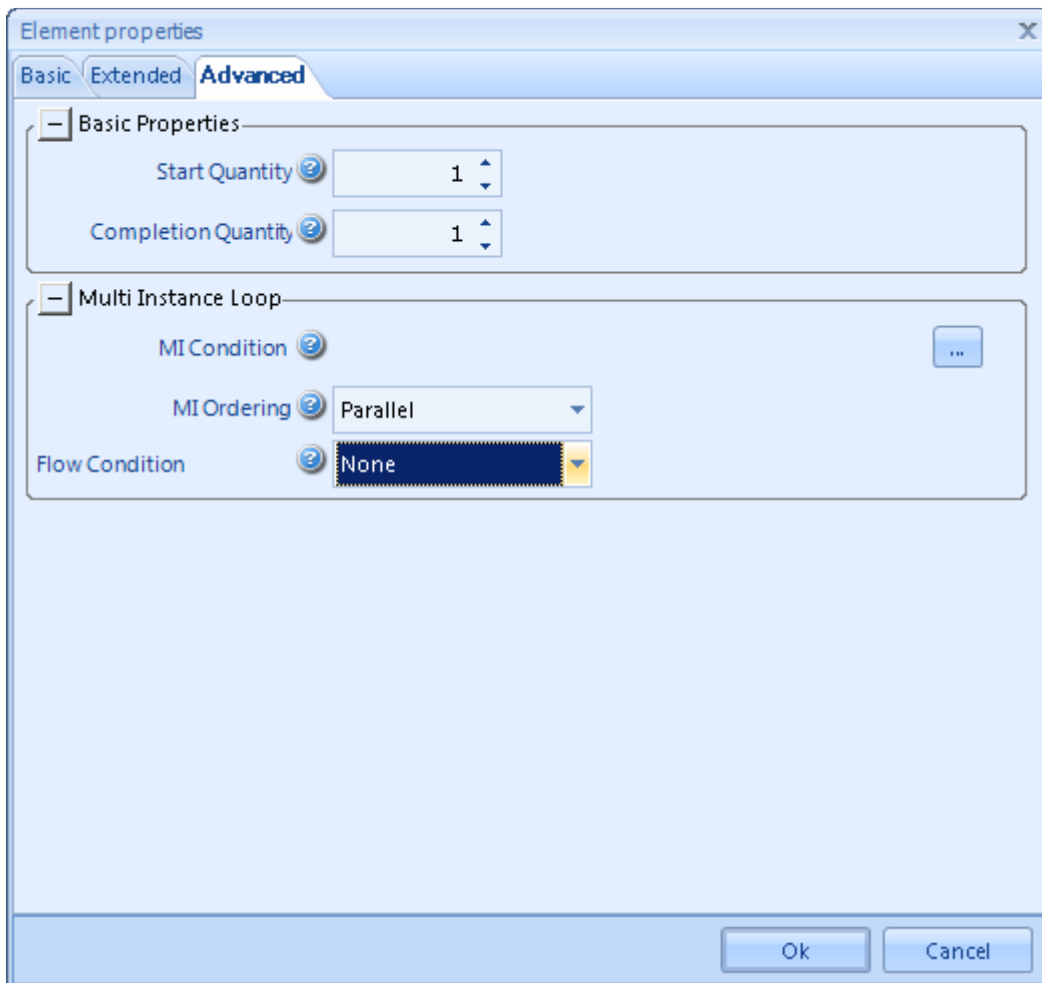


Illustration 19: Properties window for an activity with the required settings

4.3.2 Multiple Instances with a Priori Design-Time Knowledge (WCP 13)

General Description

This pattern describes that a given number of instances are independent of each other and are run concurrently. It is necessary to synchronize the different task instances after completion before any subsequent tasks can be enabled [Aalst09].

Use Case

There's an event coming soon. The number of participants is known beforehand. The Office sends an invitation to each of the twenty participants. As soon as all of them have replied to the invitation the Office can continue the event preparation.

Implementation

You can realize this pattern in BizAgi by using a Multiple Instances activity. Simply change the MI Ordering to Parallel, the MI Flow Condition to All and the Completion Quantity to 20.

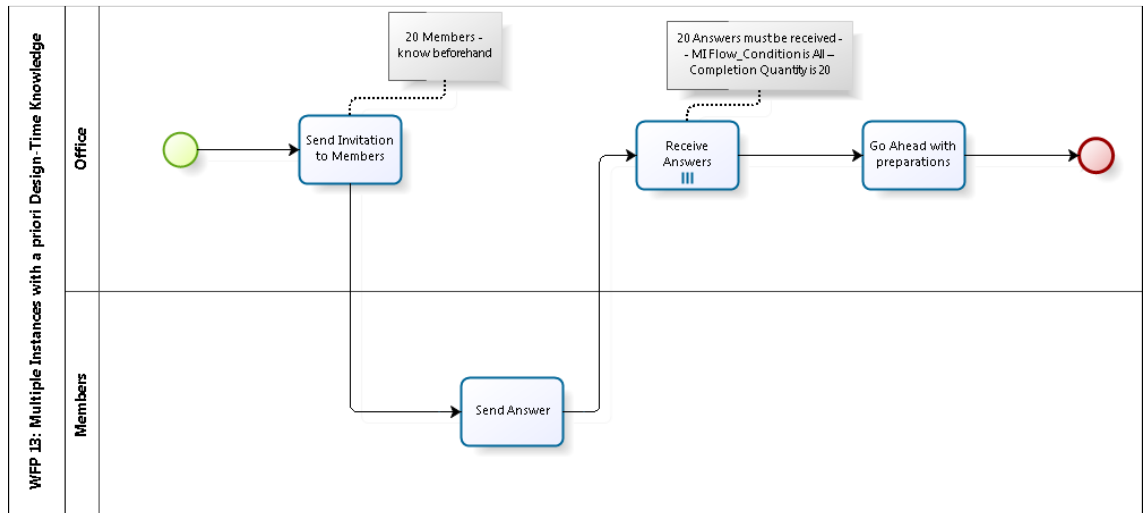


Illustration 20: Implementation of the "Multiple Instances with a Priori Design-Time Knowledge" pattern

You can edit all these values by doing a right click on the activity (or pressing F4 on your keyboard) and switching to the Advanced tab in the properties box.

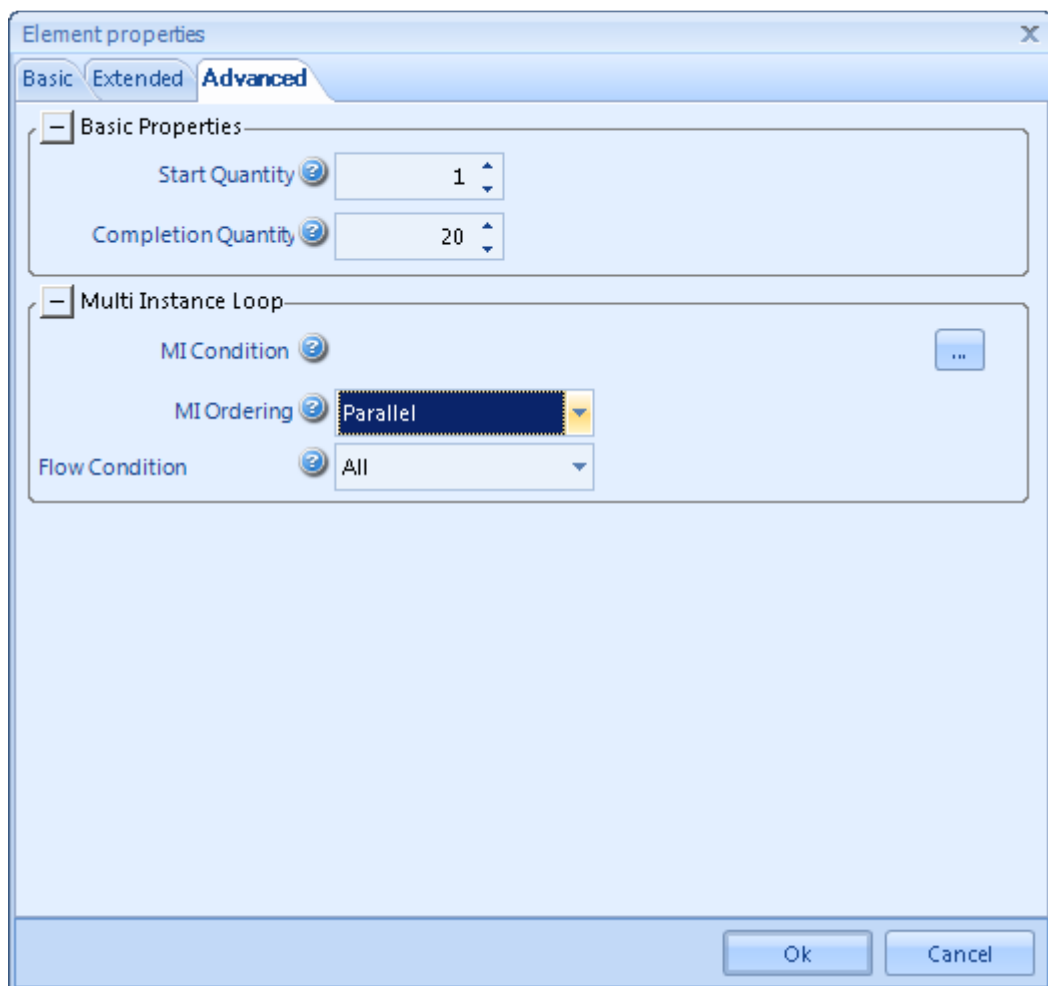


Illustration 21: Required properties settings for WCP 13

4.3.3 Multiple Instances with a Priori Run-Time Knowledge (WCP 14)

General Description

This pattern is similar to WCP 13. The only difference is that the amount of instances is not known beforehand but during execution before the task instances must be created. It is necessary to synchronize the different task instances after completion before any subsequent tasks can be enabled [Aalst09].

Use Case

This Use Case is similar to the one presented before for WCP 13. The only difference is that the amount of messages sent out by the office is not known beforehand but dynamically during the planning phase.

Implementation

You can realize this Use Case with an Multiple Instance activity. Change the value of MI Ordering to Parallel, the Flow Condition to Complex and define a complex mutable Condition on the MI Instance.

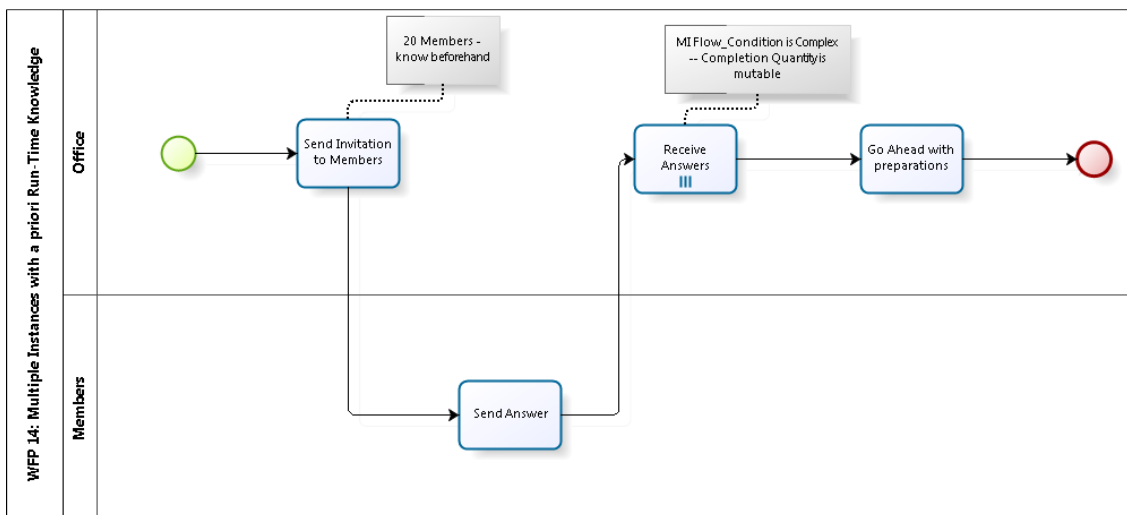


Illustration 22: Implementation of the "Multiple Instances with a Priori Run-Time Knowledge" pattern

You can change these values by simply doing a right click on the activity and switching to the Advanced tab of the properties box.

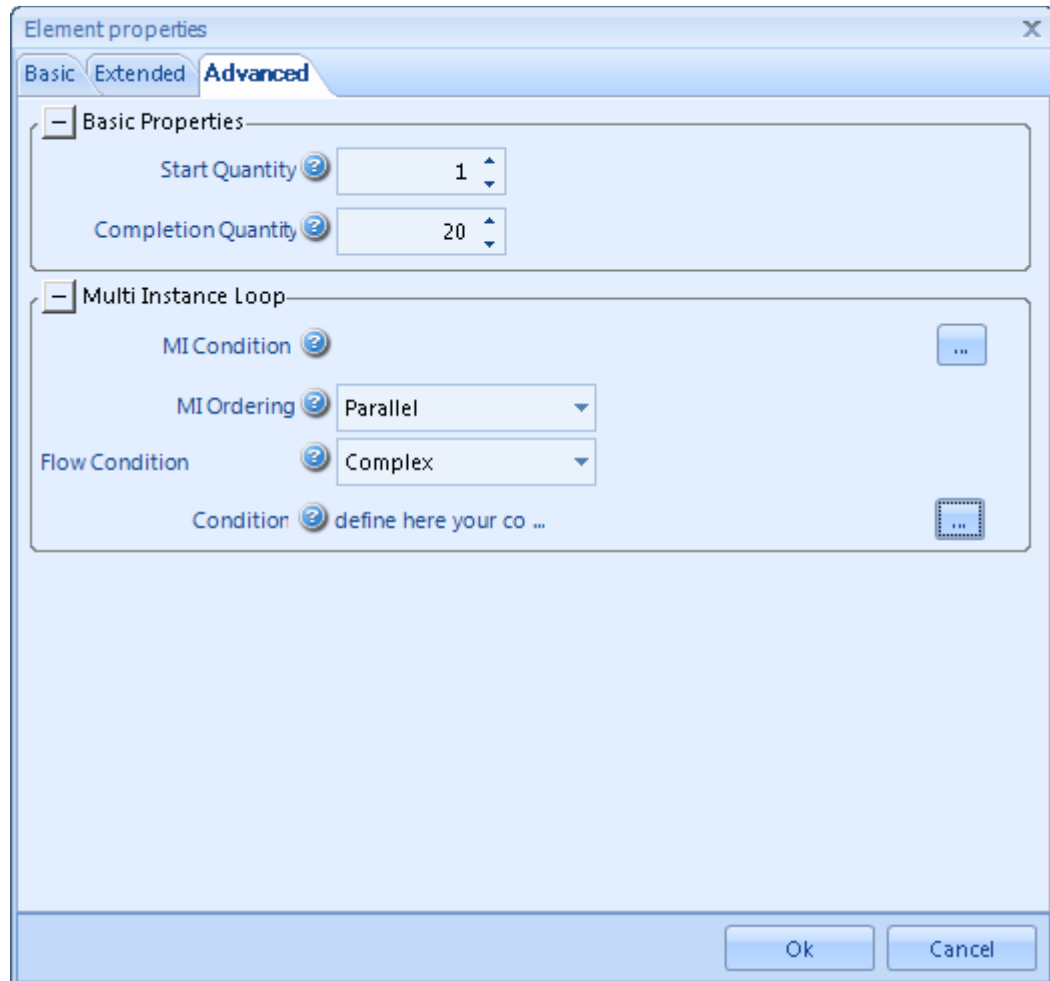


Illustration 23: Required properties settings for WCP 14

4.3.4 Multiple Instances without a Priori Run-Time Knowledge (WCP 15)

General Description

This pattern is similar to WCP 13. The only difference is that the amount of instances is not known until the final instance has completed. At any time, whilst instances are running, it is possible for additional instances to be initiated. It is necessary to synchronize the instances at completion before any subsequent tasks can be triggered [Aalst09].

Use Case

This Use Case is similar to the one presented before for WCP 14. The only difference is that the Office continues inviting people until the required amount of participants has been reached.

Implementation

This pattern is not supported by BizAgi because there is no way of adding a dynamic number of further instances to a Multiple Instance activity when the MI activity already has been started.

4.3.5 Static Partial Join for Multiple Instances (WCP 34)

General Description

This pattern describes that within a given process instance, multiple concurrent instances of a task let's pretend m) can be created. Once n of the task instances have completed (where n is less than m), the next task in the process is triggered. Subsequent completions of the remaining $m-n$ instances are inconsequential, however all instances must have completed in order for the join construct to reset and be subsequently re-enabled [Aalst09].

Use Case

A questionnaire is sent out to a fixed number (let's pretend m) of members. The Office waits now for at least n (n is less than m) replies to this questionnaire before they can start with the evaluation.

Implementation

In theory this pattern is supported in BizAgi by using multiple instance activities. Change the MI Ordering to Parallel, the MI Flow Condition to Complex and enter an expression that evaluates to true when exactly M instances have completed for the Condition. You can find all these values by simply doing a right click on the multiple instance activity and switching to the tab Advanced. However it is unclear how the Condition for the Complex Flow should be exactly specified.

4.3.6 Cancelling Partial Join for Multiple Instances (WCP 35)

General Description

This pattern is very similar to WCP 34 (please refer to page 25). The only difference is that as soon as n task instances have been completed the preceding task in the business process is enabled and the remaining k tasks ($k = m - n$) are cancelled and withdrawn [Aalst09].

Use Case

An invitation about a work outing to Italy is sent out to a fixed number (let's pretend m) of members. In the invitation it is mentioned that there are only n (n is less than m) spaces available and that the registration process is FCFS (first come first serve). As soon as all spaces are occupied all other members are informed that they can no longer participate in the work outing.

Implementation

In theory this pattern is supported in BizAgi by using multiple instance activities. Change the MI Ordering to Parallel, the MI Flow Condition to Complex and enter an expression that evaluates to true when exactly M instances have completed for the Condition. You can find all these values by simply doing a right click on the multiple instance activity and switching to the tab Advanced. Additionally an error type intermediate event trigger at the boundary of the MI activity is required which will terminate any remaining MI activities after a cancel event. However it is unclear how the Condition for the Complex Flow should be exactly specified.

4.3.7 Dynamic Partial Join for Multiple Instances (WCP 36)

General Description

This pattern describes that within a given process instance, multiple concurrent instances of a task can be created. The required number of instances may depend on a number of runtime factors and is not known until the final instance has completed. At any time, whilst instances are running, it is possible for additional instances to be initiated. A completion condition is specified which is evaluated each time an instance of the task completes. Once the completion condition evaluates to true, the preceding task in the process is triggered regardless of the state of the remaining task instances [Aalst09].

Use Case

The Development department has a problem and it commissions a dynamic number of developers to solve this problem. If one of the developers submits a solution the team leader decides whether this solution is sufficient or not. As long as there is no solution which satisfies the team leader 100 percent all other developers can submit their solutions and other developers can be hired. However the management can declare a hiring freeze resulting in a constant number of developers. The developers who have already been hired however can continue their work on finding a solution for the problem.

Implementation

This pattern is not supported by BizAgi because there is no way of adding a dynamic number of further instances to a Multiple Instance activity when the MI activity already has been started. However the describes Use Case can be modeled via a work-around using several splits, merges and loops.

4.4 State-based Patterns

The group of state-based patterns reflects situations for which solutions are most easily accomplished by the notion of states.

4.4.1 Deferred Choice (WCP 16)

General Description

The Deferred Choice pattern describes the individual selected of one branch from several branches. The decision is based on interaction with the environment [Aalst09].

Use Case

An applicant has read the AGB and decides now to become a member. He has now to decide whether he wants a sponsor contract or an ordinary membership application. Doing both is not possible. The decision is manual.

Implementation

This pattern is supported by BizAgi by using an event-based exclusive gateway.

Subsequent to this gateway there is either an intermediate event using message-based triggers or a receive task required.

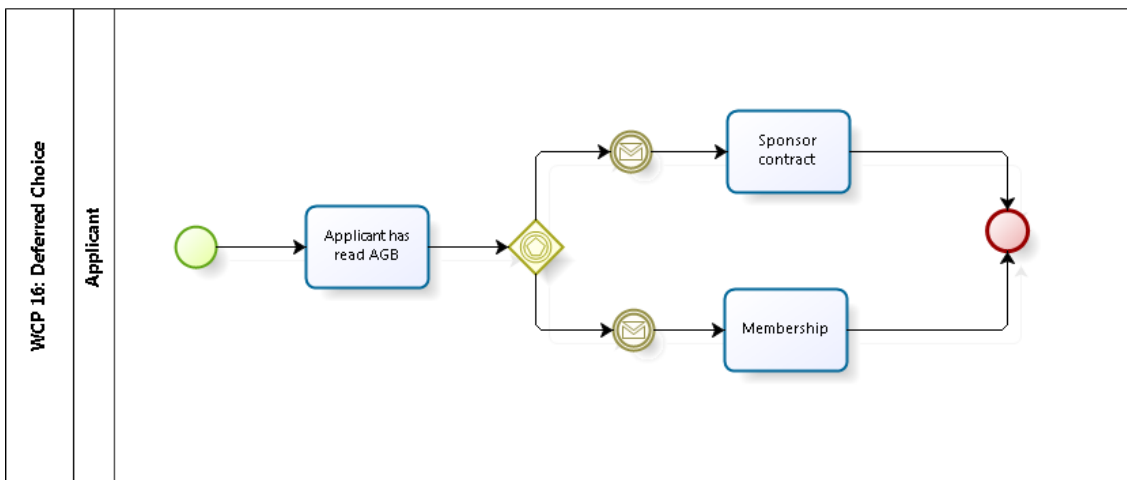


Illustration 24: Implementation of the "Deferred Choice" pattern

4.4.2 Interleaved Parallel Routing (WCP 17)

General Description

The Interleaved Parallel Routing pattern describes a set of tasks with a partial ordering. Each task in the set must be completed once and they can be executed in any order that fits with the partial order. Additionally the tasks cannot be executed at the same time [Aalst09].

Use Case

The manager of the education department informs the employee that he or she should do three different phases during the aptitude test. A psychological test, an intelligence test and a round of introduction. The order in which the three phases are accomplished is relevant. The intelligence test must be revised before doing the psychological test. The round of introduction can be accomplished as first phase, as last phase or also in the middle of the other two phases. However you are not allowed to do two phases at the same time.

Implementation

This pattern is partly supported by BizAgi. For simple tasks you can use an ad-hoc process but for interleaving groups or sequences of tasks there is no support. However the describes Use Case can be modeled via a work-around using several splits, merges and loops.

4.4.3 Milestone (WCP 18)

General Description

This pattern defines that an activity can only be enabled when the process instance is in a specific state. This state is assumed to be a specific execution point (also known as a milestone) in the business process. As soon as this point has been

reached the nominated task can be enabled. If the current state is already beyond this state the task cannot be enabled any longer (e.g. a deadline has expired) [Aalst09].

Use Case

In a project the employees have to do a documentation and a presentation. If the presentation is ready they can wait until the documentation is ready for refining the presentation. If the documentation has been sent to the manager (before or after completing the presentation) there is no further chance of editing the presentation.

Implementation

This pattern is not supported in BizAgi.

4.4.4 Critical Section (WCP 39)

General Description

The Critical Section pattern describes that two or more connected sub-processes are identified as critical sections. When one of these critical sections is active - which means that an activity inside this section is enabled - no other critical section can be activated. The business process waits for completion or halting one critical section to be able to activate another critical section [Aalst09].

Use Case

Two administrators have to access a server for changing some settings. As long as one admin is currently on the server there is no chance for the other admin to access the server too. He or she has to wait until the other admin has left the server.

Implementation

This pattern is not directly supported in BizAgi because there is no way to limit the concurrent execution of a set of activities. However the describes Use Case can be modeled via a work-around using several splits, merges and loops.

4.4.5 Interleaved Routing (WCP 40)

General Description

This pattern describes that each member of a set of tasks must be executed once. They can be executed in any order but at no point of time two or more tasks can be executed simultaneously. Once all of the tasks have completed, the next task in the process can be initiated [Aalst09].

Use Case

This Use Case is similar to the one presented for WCP 17. The difference is that the order in which the three phases are accomplished is no longer relevant but you are not allowed to do two phases at one single point of time.

Implementation

This pattern is partly supported by BizAgi via a so called ad-hoc process. This ad-

hoc process should contain all of the activities interleaved. The AdHocOrdering must be set to Sequential and there needs to be a CompletionCondition defined. You can change these values by simply doing a right click on the activity and switching to the Advanced tab. Unfortunately it is unclear how the required CompletionCondition should look like to ensure that each activity has been executed exactly once. However the describes Use Case can be modeled via a work-around using several splits, merges and loops.

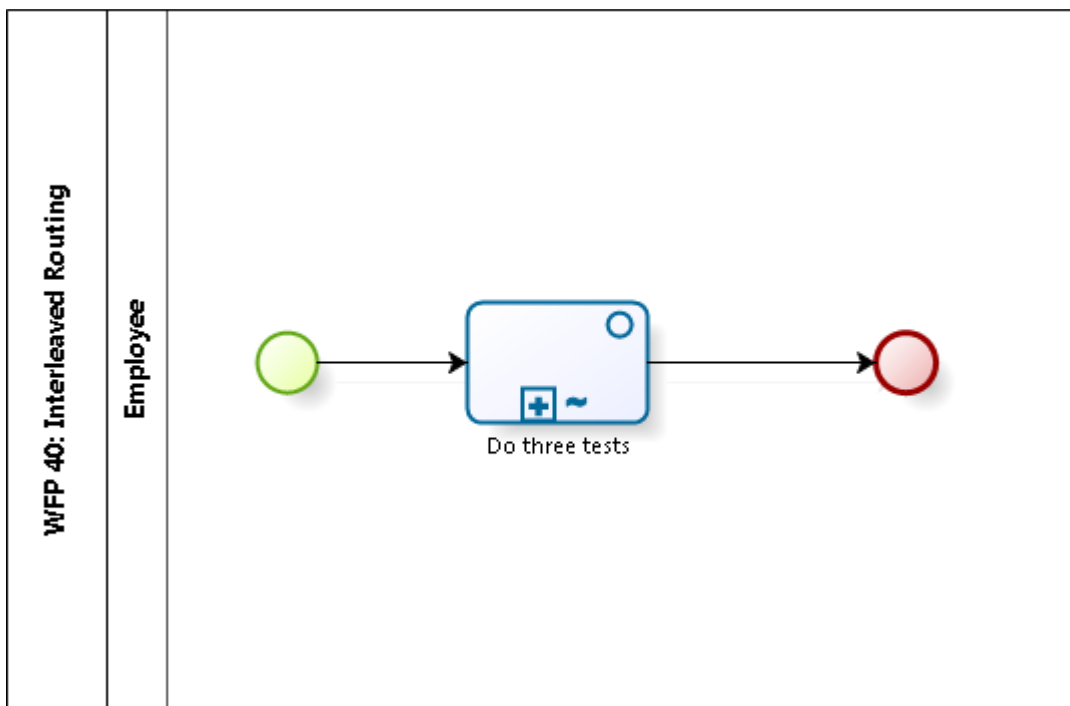


Illustration 25: Implementation of the "Interleaved Routing" pattern using an Ad-Hoc activity

4.5 Cancellation and Force Completion Patterns

The patterns in this group utilize the concept of cancelling or withdrawing activities.

4.5.1 Cancel Task (WCP 19)

General Description

The Cancel Task pattern describes the possibility of canceling or withdrawing an enabled task. If the task has already started it is disabled and the currently running instance is halted and removed [Aalst09].

Use Case

The Office Assistant creates a PowerPoint presentation for a manager. If the manager sends a message that the presentation is no longer required the assistant stops working on the presentation continuing with the normal work.

Implementation

You can realize this Use Case by adding an attached error event to the Create Presentation Activity. When triggered the error event then leads to the Go ahead without presentation activity.

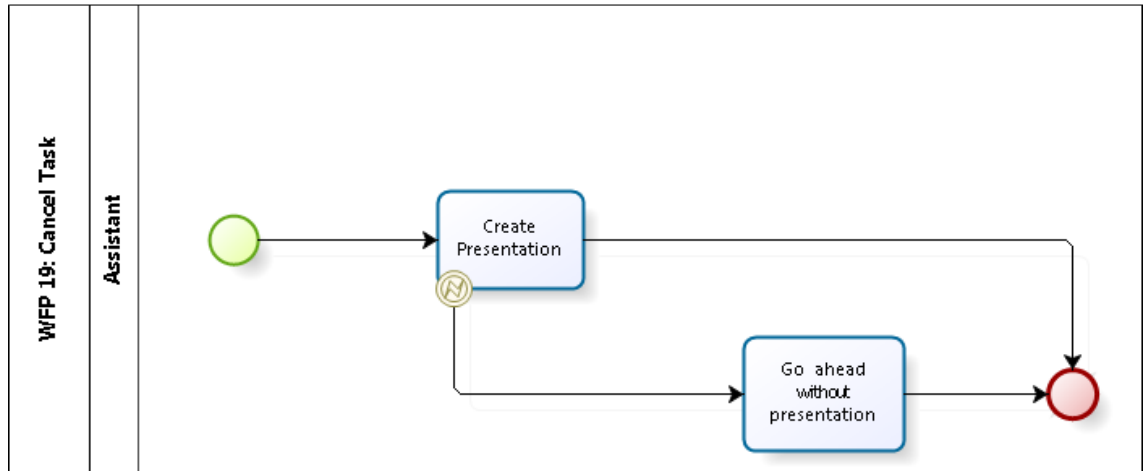


Illustration 26: Implementation of the "Cancel Task" pattern

4.5.2 Cancel Case (WCP 20)

General Description

The Cancel Case pattern describes the removal of a complete process instance. This includes currently running tasks as well as those which may be executed at some time in future. Additionally all sub-processes are also removed and the process instance is regarded as having completed without success [Aalst09].

Use Case

This Use Case is an extension of the Use Cases presented for WCP 06 and WCP 07 (please refer to page 8 and 9). An Applicant has read the AGB and decides to become a member. He or she can do this by signing a sponsor contract and/or a membership application. After submitting his date the applicant might change his or her mind and cancel his or her application.

Implementation

This pattern is available in BizAgi by including the entire process in a sub process and adding a cancel type intermediate event trigger at the boundary of the sub process. This event will terminate all activities associated with a process instance.

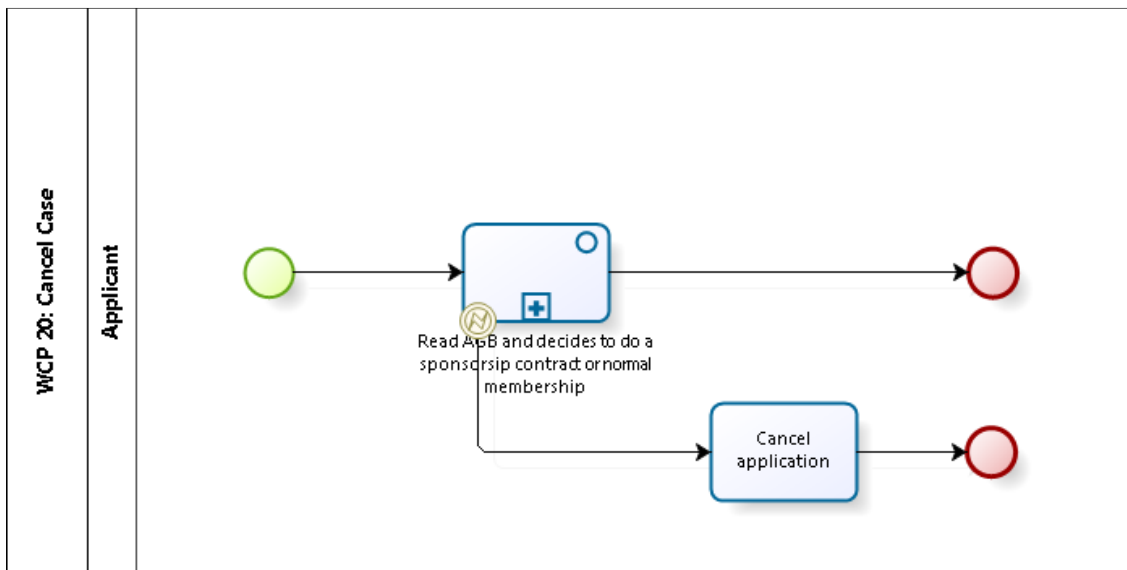


Illustration 27: Implementation of the "Cancel Case" pattern

4.5.3 Cancel Region (WCP 25)

General Description

This pattern describes the ability to disable a whole set of tasks in a process instance. If any of the nominated tasks is already executed or enabled then it is withdrawn. The set of tasks does not need to be a connected subset of the overall business process [Aalst09].

Use Case

This Use Case is similar to the one presented for the Cancel Case pattern (WCP 20, please refer to page 30). If the applicant changes his or her mind he or she just wants to cancel the sponsor contract.

Implementation

This pattern is supported by enclosing the cancellation region in a sub process and adding a cancel type intermediate event trigger at the boundary of the sub process. This event will terminate all activities associated with a process instance. The only limitation is that the cancellation region must be a connected sub graph of the overall business process.

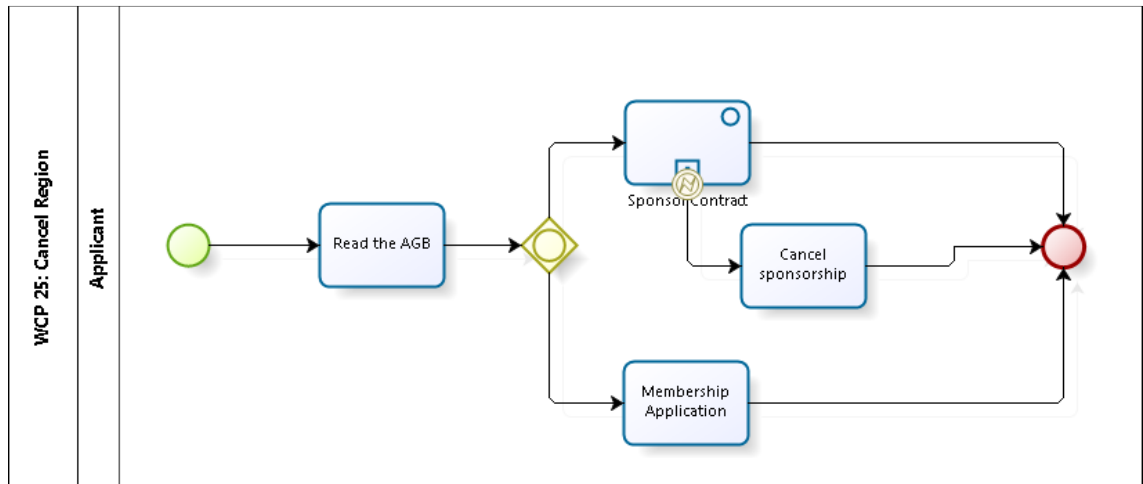


Illustration 28: Implementation of the "Cancel Region" pattern

4.5.4 Cancel Multiple Instance Activity (WCP 26)

General Description

The Cancel Multiple Instance Task pattern describes the ability of cancelling a whole Multiple Instance task by withdrawing all instances which have not yet been completed. All others - already completed - are not affected [Aalst09].

Use Case

This Use Case is based on the Use Case described for WCP 36 (please refer to page 26). It describes that the management decides to cancel the whole project while several developers are working on a solution.

Implementation

This pattern is realizable in BizAgi by using a Multiple Instance activity with an error type intermediate event trigger at the boundary. As soon as the MI activity is to be withdrawn, a cancel event is triggered to terminate any remaining MI activities.

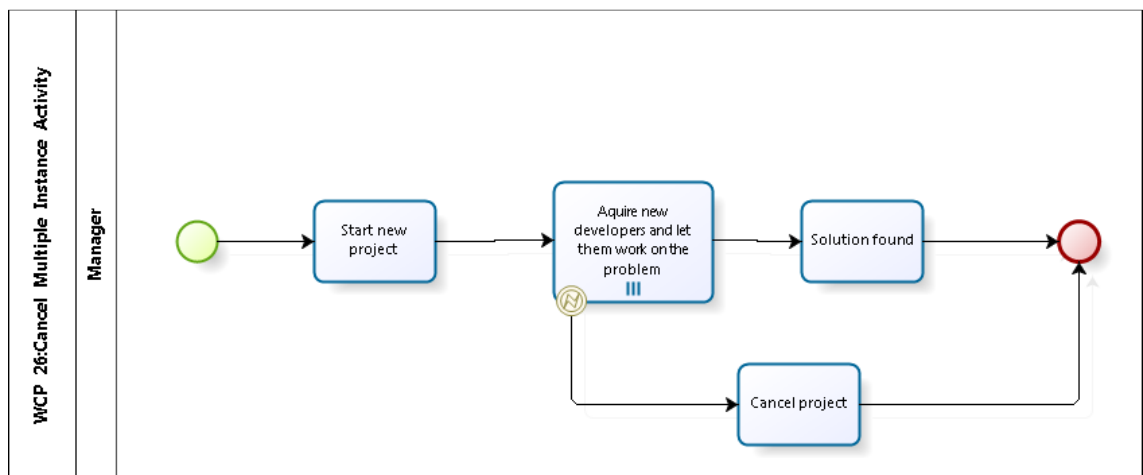


Illustration 29: Implementation of the "Cancel Multiple Instance Activity" pattern

4.5.5 Complete Multiple Instance Activity (WCP 27)

General Description

The Complete Multiple Instance Task pattern describes the possibility of enforcing the completion of remaining instances within a Multiple Instance activity by withdrawing any remaining instances and passing the thread of control to the subsequent task [Aalst09].

Use Case

This Use Case is based on the Use Case described mentioned for WCP 14 (please refer to page 23). The Use Case describes that the employees can continue directly with the event planning with no regard on the number of received answers.

Implementation

This pattern is realizable in BizAgi by using a Multiple Instance activity with an error type intermediate event trigger at the boundary. As soon as the MI activity is to be withdrawn, a cancel event is triggered to terminate any remaining MI activities.

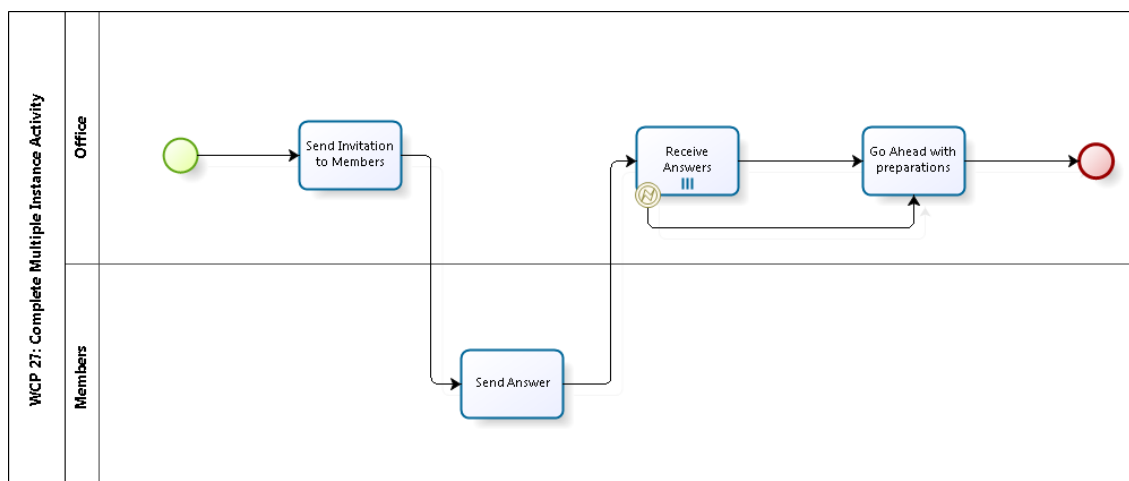


Illustration 30: Implementation of the "Complete MI Activity" pattern

4.6 Iteration Patterns

The patterns within the group Iterations patterns deal with capturing repetitive behavior in a business process.

4.6.1 Arbitrary Cycles (WCP 10)

General Description

The Arbitrary Cycles pattern describes a distinct point in a business process in which one or more activities are repeated several times. Special about these loops is that they have more than one entry or exit point. It must be possible for individual entry and exit points to be associated with distinct branches [Aalst09].

Use Case

An employee has a meeting with a customer. If the requirements of the customer are not feasible start the meeting again and try to find other requirements. If there is a possibility to implement the requirements the employee starts implementing

them and tests them. If the tests are successful the business process is completed. If the tests are not successful the employee tries to find out if there is another way of implementing the requirements. If yes he tries the new way and tests them. If not the employee has to set up a new appointment with the customer trying to change the requirements.

Implementation

There is no particular BPMN element within BizAgi for Arbitrary Cycles but you can easily adopt this pattern by using basic splits, merges and control expressions.

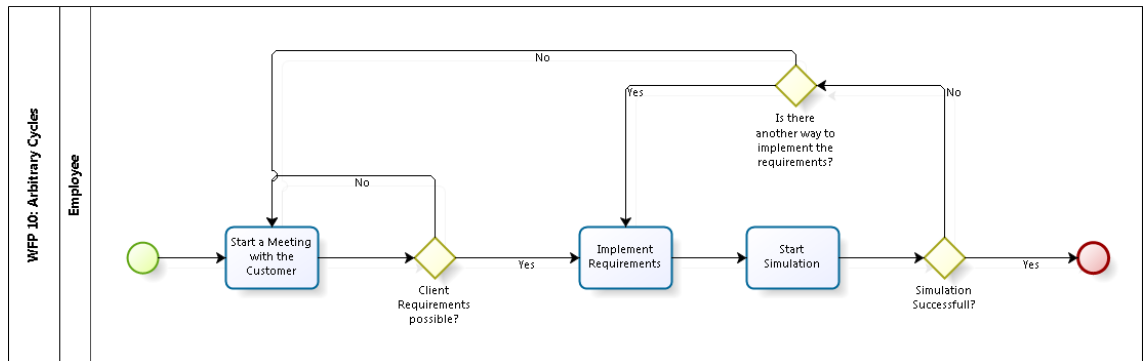


Illustration 31: Implementation of the "Arbitrary Cycles" pattern

4.6.2 Structured Loop (WCP 21)

General Description

The Structured Loop pattern describes the possibility of executing an activity or sub-process repeatedly. This loop has either a pre-test or a post-test condition which means that the condition is either evaluated at the beginning or the end of a loop. The loop itself has a single entry point and a single exit point. The pattern itself consists of three possibilities: the normal one, the pre-test one and the post-test one [Aalst09].

Since the pattern consists of three parts there are three different Use Cases and Implementations.

4.6.2.1 Normal

Use Case

The Office begins with the Education phase. Therefore it sends different mappings to a company and the company replies saying yes or no to this mapping. If the answer is "OK" the process is completed. If the answer is "No interest" the office has to change to application and start again with the Education phase.

Implementation

You can easily realize this Use Case by using basic activities in BizAgi.

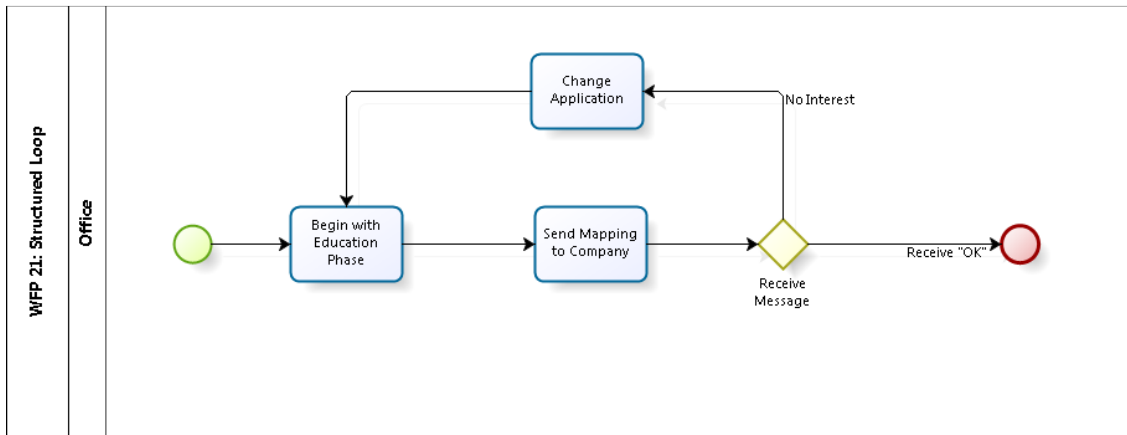


Illustration 32: Implementation of the "Structured Loop" pattern

4.6.2.2 Post

Use Case

The Office receives a task from a customer. They not start working and implementing the task. If the customer is satisfied the process is completed. If not the office has to work and implement again.

Implementation

In BizAgi you can realize Structured Loops with a Post Evaluation of the Condition easily by transforming a basic activity into a Standard Loop.

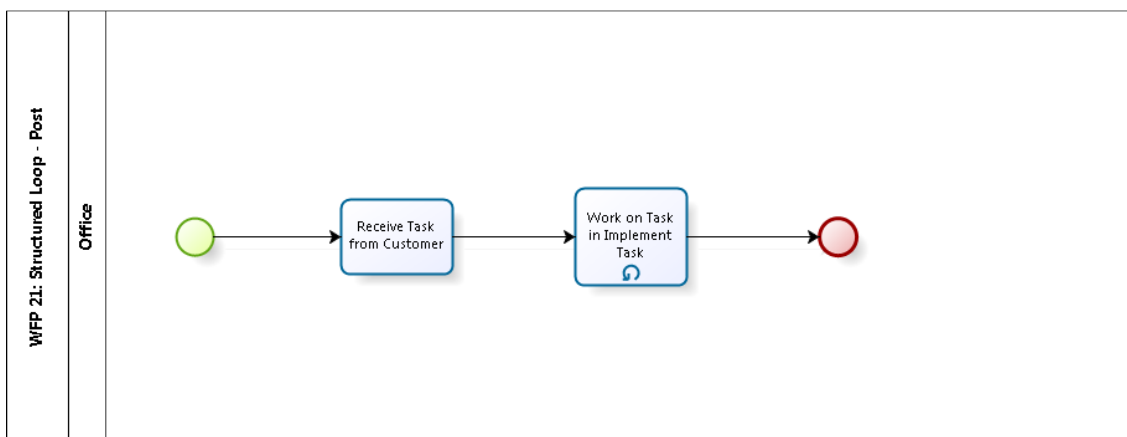


Illustration 33: Implementation of the "Structured Loop" pattern with Post Evaluation

Additionally you have to set the Test Time Value (under Properties, Advanced) to After.

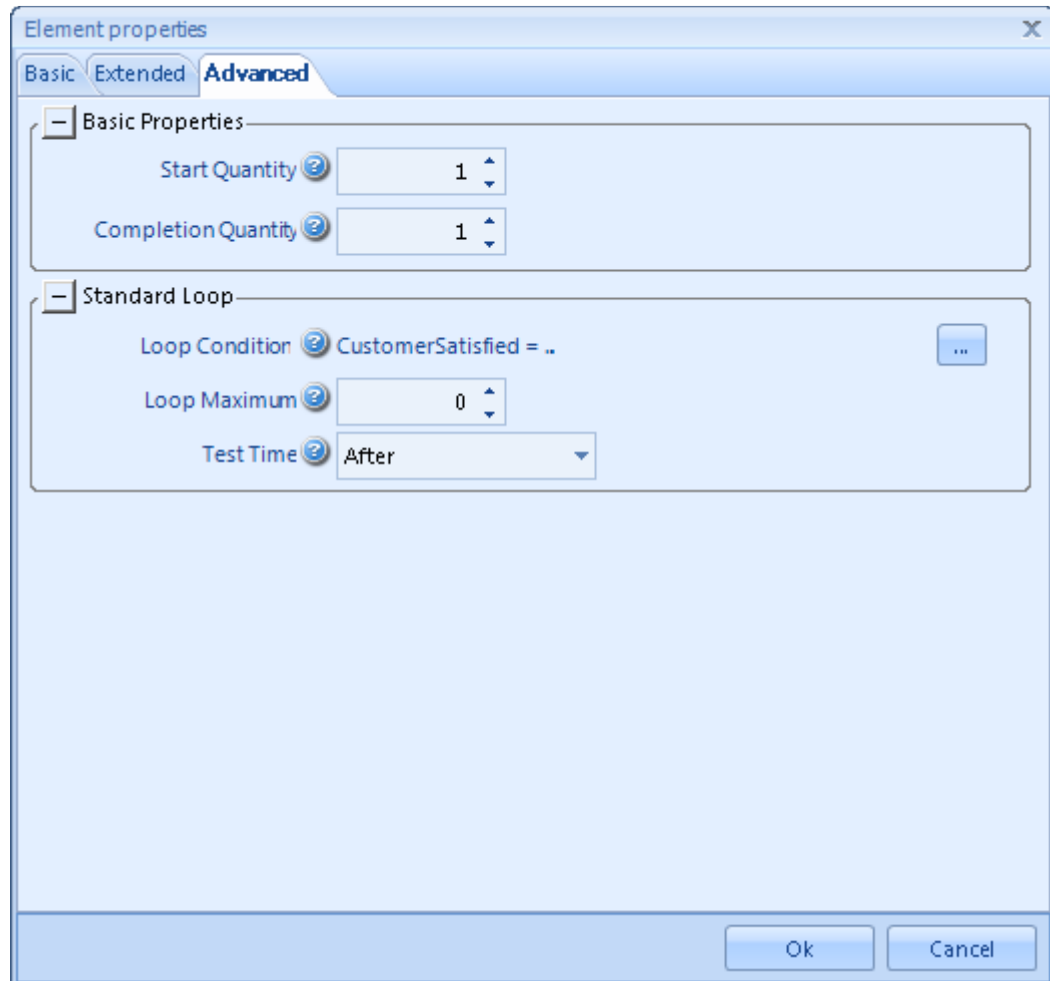


Illustration 34: Required properties for the realization of the "Structured Loop" pattern with Post Evaluation

Alternatively you can also model this process by using basic activities like splits and merges.

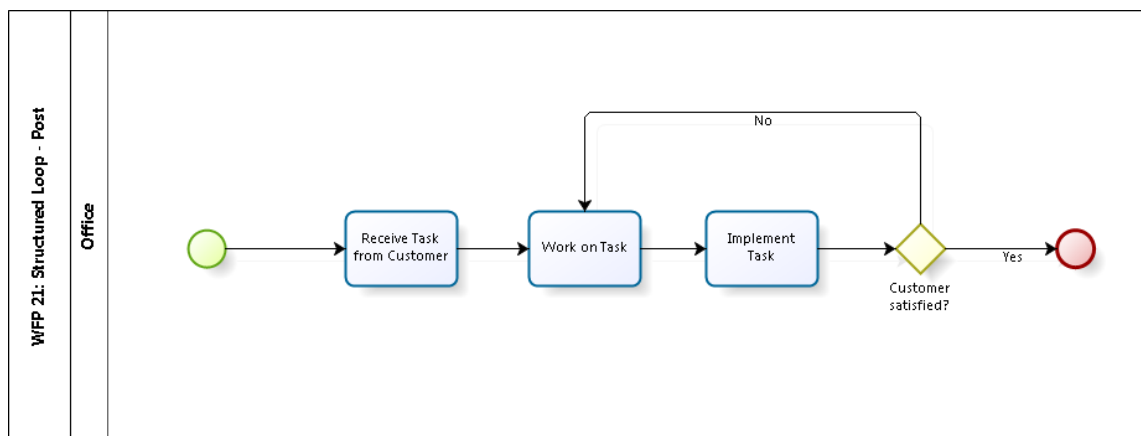


Illustration 35: Alternative Implementation of the "Structured Loop" pattern with Post Evaluation

4.6.2.3 Pre Use Case

The Office starts a new project. First of all they have to check if there is enough memory available on the server for this project. If yes the process is completed. If

not they have to order additional memory, install the memory and determine again if the available memory is enough.

Implementation

In BizAgi you can realize Structured Loops with a Pre Evaluation of the Condition easily by transforming a basic activity into a Standard Loop.

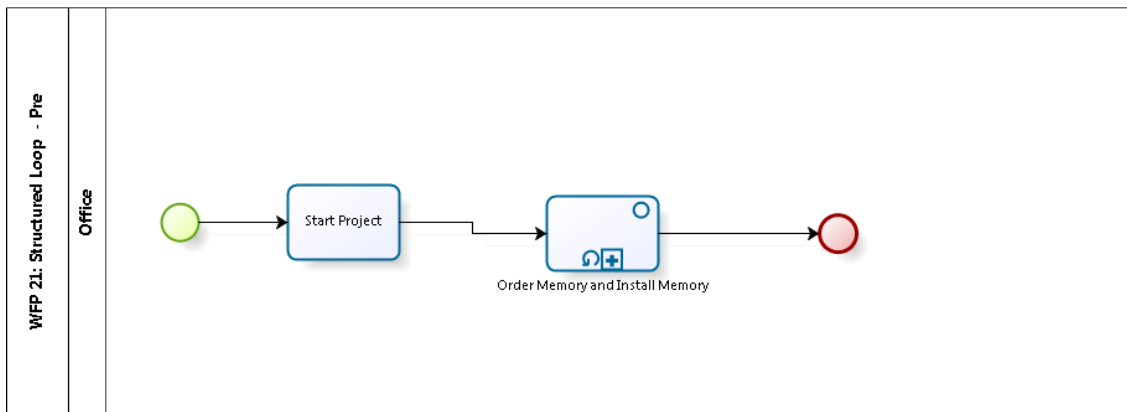


Illustration 36: Implementation of the "Structured Loop" pattern with Pre Evaluation

Additionally you have to set the Test Time Value (under Properties, Advanced) to Before. In this example the Loop is additionally an embedded sub process because you have to loop two activities at the same time.

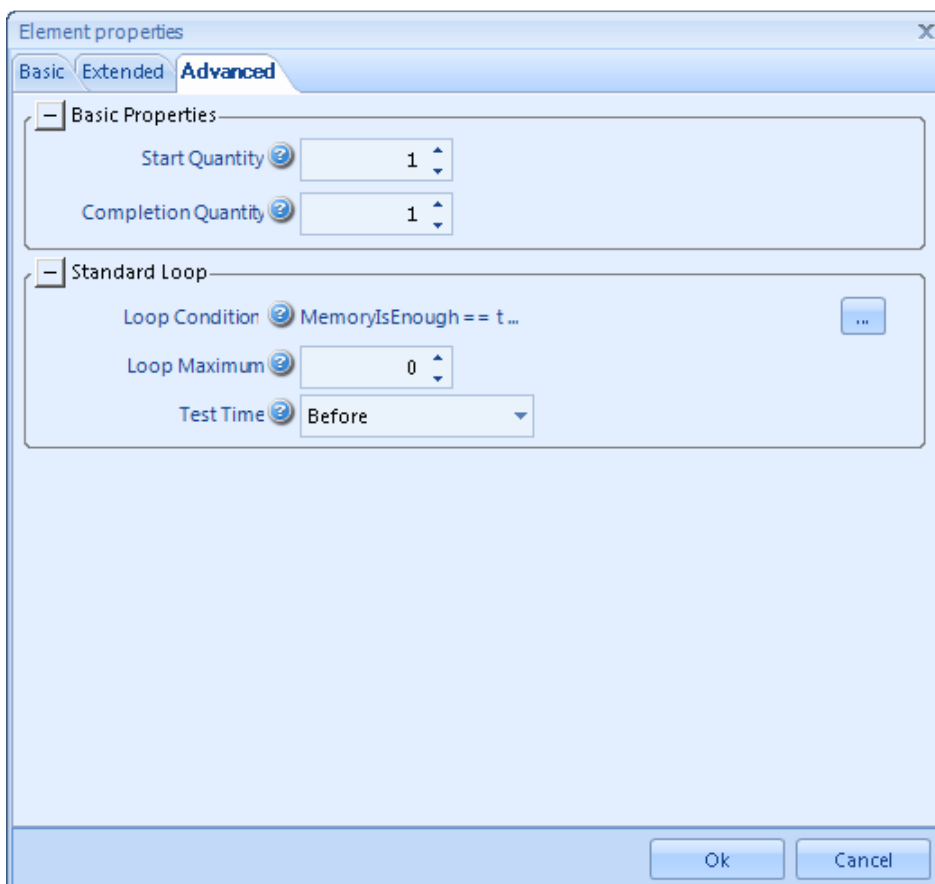


Illustration 37: Required properties for the realization of the "Structured Loop" pattern with Pre Evaluation

Alternatively you can also model this process by using basic activities like splits and merges.

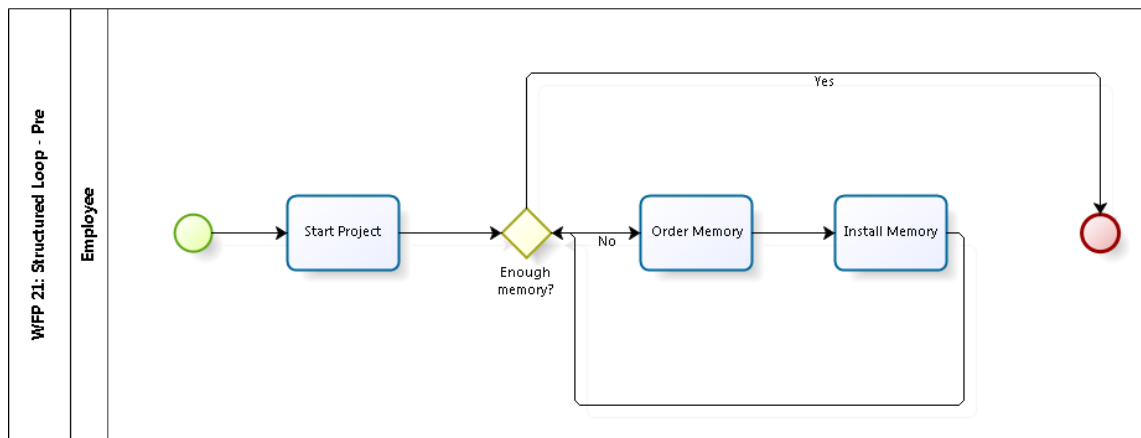


Illustration 38: Alternative Implementation of the "Structured Loop" pattern with Pre Evaluation

4.6.3 Recursion (WCP 22)

General Description

The Recursion pattern describes the ability of a task of invoking itself in terms of the overall decomposition structure with which it is associated [Aalst09].

Use Case

An employee of HR is doing dossiers of incoming application. The employee continues doing this until he or she has twenty dossiers or until the management sets a deadline for the application. However after the process exits the loop (it doesn't matter if this happens regularly or because of a deadline) the HR employee plans an appointment and sends this appointment for each dossier to the corresponding applicant.

Implementation

This pattern is not directly supported within BizAgi because there is no way of specifying recursive composition with a business process model. However the describes Use Case can be modeled via a work-around using several splits, merges and loops.

4.7 Termination Patterns

All patterns belonging to the Termination Patterns area dealing with the issue under which circumstances a business process can be considered as completed.

4.7.1 Implicit Termination (WCP 11)

General Description

The Implicit Termination pattern describes that a business process should terminate as soon as there are no more remaining items than can be processed and the process itself is not deadlocked [Aalst09].

Use Case

For this Use case we take a look at the hardware ordering process. The inventory department receives a request for a new hardware. An employee of this department then sends the new hardware to the computing centre and a bill to the accounting. The process is finished or terminated as soon as the new hardware has been installed and the bill has been booked.

Implementation

This pattern can be realized in BizAgi by simply adding two or more End Events to the process with several incoming branches and no outgoing branches. The BizAgi Suite will consider the whole business process as finished as soon as all End Events are enabled.

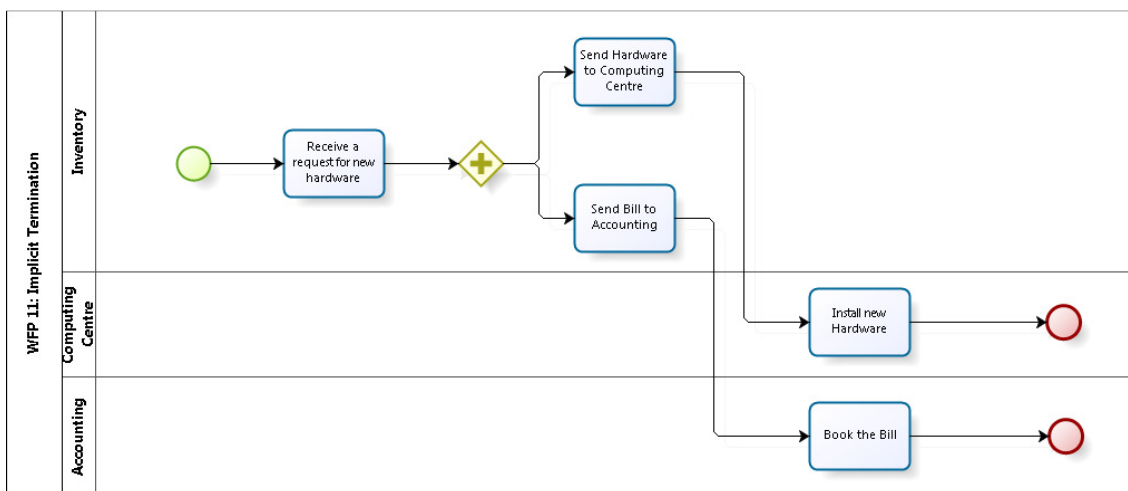


Illustration 39: Implementation of the "Implicit Termination" pattern

4.7.2 Explicit Termination (WCP 43)

General Description

The Explicit Termination pattern defines that a process should terminate as soon as it reaches a nominated state, typically denoted by a specific end node. When this end node is reached by any active process path the whole process is regarded as completed. Any remaining work is canceled [Aalst09].

Use Case

Let's assume a server is not working properly. The computing centre requests support from the Engineering team and the development team which both do analyze the error. As soon as one team have found and solved the error, the whole business process is completed regardless of the progress of the other team.

Implementation

This pattern can be realized in BizAgi by using an End Event in the business process with several incoming branches and no outgoing branches. The BizAgi Suite will consider the whole business process as finished as soon as this End Event is enabled.

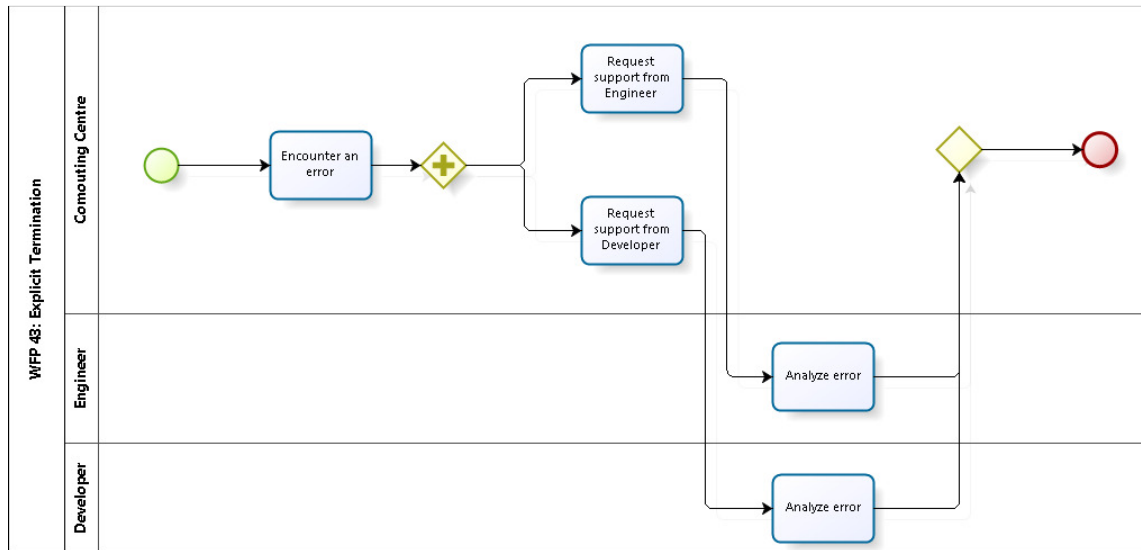


Illustration 40: Implementation of the "Explicit Termination" pattern

4.8 Trigger Patterns

The last group Trigger Patterns deals with external signals that may be required to start certain tasks.

4.8.1 Transient Trigger (WCP 23)

General Description

The transient trigger pattern describes that a task can be triggered by a signal from other parts of the process or an external event. These triggers are transient in nature and are lost if not acted on immediately by the receiving task. A trigger can only be utilized if there is a task instance waiting for it at the time it is received [Aalst09].

Use Case

In a company the secretary catches the manager's mail two times a day. When the secretary is passing by the manager's office and the manager has not yet completed his reports, the secretary simply passes by. If the manager has completed the reports the secretary will them with her.

Implementation

There is no distinct symbol or way in BizAgi to implement this pattern. However it is possible to model the above mentioned Use Case by using basic activities and basic splits and merges.

4.8.2 Persistent Trigger (WCP 24)

General Description

The persistent trigger pattern describes that a task can be triggered by a signal from other parts of the process or an external event. These triggers are persistent in form and are retained by the process until they can be acted on by the receiving task.

Use Case

A Developer from the Development Department starts a new project. He therefore requests the required server capacity and develops the software. During or after the development he or she has to wait for a message (event) that the server capacity is available. After receiving this event the developer can do the project test and product rollout.

Implementation

In BizAgi this pattern is realized via a message event connected to a preceding and proceeding activity. The business process is then halted until the message is received.

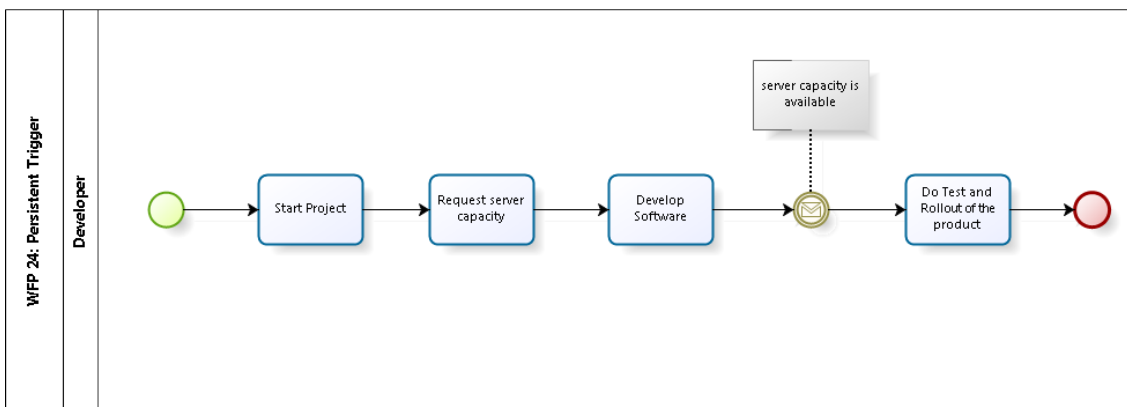


Illustration 41: Implementation of the "Persistent Trigger" pattern

5 Conclusion

In this chapter you will get a short overview of the results of my assignment including a wrap up and a further forecast.

5.1 Overview about the results

In this table you can get a short overview about the possible implementation of the different 43 patterns. The symbol + (green) represents direct support, the symbol +/- (yellow) represents an indirect support via a work around and the symbol - (red) represents that the pattern is not supported.

5.1.1 Basic Control Flow Patterns

Pattern	Result	Refer to...
Sequence	+	page 4
Parallel Split	+	page 5
Synchronization	+	page 6
Exclusive Choice	+	page 6
Simple Merge	+	page 7

5.1.2 Advanced Branching and Synchronization Patterns

Pattern	Result	Refer to...
Multi-Choice	+	page 8
Structured Synchronizing Merge	+	page 9
Multi Merge	+	page 10
Structured Discriminator	+/-	page 11
Blocking Discriminator	+/-	page 11
Cancelling Discriminator	+	page 12
Structured Partial Join	+/-	page 12
Blocking Partial Join	+	page 13
Cancelling Partial Join	+	page 13
Generalized AND Join	+	page 14
Local Synchronizing Merge	+	page 15
General Synchronizing Merge	+/-	page 16
Thread Merge	+	page 16
Thread Split	+	page 18

5.1.3 Multiple Instance Patterns

Pattern	Result	Refer to...
MI without Synchronization	+	page 20
MI with a priori Design Time Knowledge	+	page 22
MI with a priori Runtime Knowledge	+	page 23
MI without a priori Runtime Knowledge	-	page 25
Static Partial Join for MI	-	page 25
Cancelling Partial Join for MI	-	page 25
Dynamic Partial Join for MI	+/-	page 26

5.1.4 State-based Patterns

Pattern	Result	Refer to...
Deferred Choice	+	page 27
Interleaved Parallel Routing	+/-	page 27
Milestone	-	page 28
Critical Section	+/-	page 28
Interleaved Routing	+/-	page 29

5.1.5 Cancellation and Force Completion Patterns

Pattern	Result	Refer to...
Cancel Task	+	page 30
Cancel Case	+	page 30
Cancel Region	+/-	page 31
Cancel MI activities	+	page 32
Complete MI activities	+	page 33

5.1.6 Iteration Patterns

Pattern	Result	Refer to...
Arbitrary Cycles	+	page 33
Structured Loop	+	page 34
Recursion	+/-	page 38

5.1.7 Termination Patterns

Pattern	Result	Refer to...
Implicit Termination	+	page 38
Explicit Termination	+	page 39

5.1.8 Trigger Patterns

Pattern	Result	Refer to...
Transit Triggers	+/-	page 40
Persistent Triggers	+	page 41

5.2 Wrap Up

As seen the BizAgi Suite is a real powerful software application. The product is able to map 28 patterns directly, other 11 are map able with a small work around and only four patterns are not included. Seeing this result it is obvious and also comprehensible why BizAgi is so much accepted on the market and the product has proven that it is rightly called as one of the best BPM tools in the market.

Further conclusion of this assignment is that – in my opinion – there is some overhead in the patterns. Some of them (let's compare for example pattern 8 Synchronization and pattern 33 Generalized AND-Join) are nearly the same and only in theory different. In the real work and day-to-day business there is no perceivable difference between them. Additionally this assignment has shown that you can describe nearly all patterns with basic instructions and a workaround.

5.3 Forecast

In future there is no need for further workflow patterns in my opinion. The existing 43 are totally enough – it might also be reasonable to reduce the number of patterns.

However the BizAgi tool seems also to be primed for the next generation of BPM – the subject oriented approach. How this could look like is presented from the HPI (Hasso Plattner Institute Berlin) [Weske08] in their BPMN 1.1 definition overview.

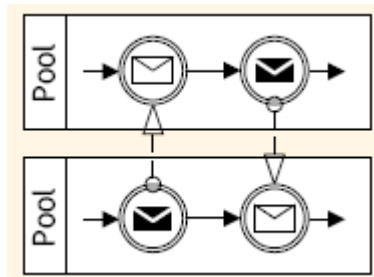


Illustration 42: BPMN 1.1 and the subject oriented approach

It is theoretically possible to model a subject oriented business process in the BizAgi Suite (as seen in illustration 42).

For more information on this topic please refer to [Graef09]

6 List of Literature

- [Weske08] Prof. Dr. Matthias Weske, Gero Decker, Alexander Grosskopf, Sven Wagner-Boysen, *BPMN 1.1*, HPI Berlin, http://bpt.hpi.uni-potsdam.de/pub/Public/BPMNCorner/BPMN1_1_Poster_EN.pdf , 2008
- [Uslar04] Mathias Uslar: *Workflow Patterns – ein Überblick und Beispiele*. Grin, 2004
- [Becker09] Jörg Becker, Christoph Mathas, Axel Winkelmann: *Geschäftsprozessmanagement*. Springer, 2009
- [Ehlers06] Stefan Ehlers: *BPM – Business Prozessmanagement in Praxis und Anwendung*. Books on demand, 2006
- [Aalst09] W.M.P van der Aalst: *Control-Flow Patterns*. <http://www.workflowpatterns.com/patterns/control/>, 2009
- [Graef09] Norbert Graef, Nils Tölle: *Evaluation, Mapping und quantitative Reduktion von Workflow Pattern (Control-Flow)*. 2009
- [BizAgi09] BizAgi: *BizAgi*. <http://www.bizagi.com> . 2009

7 List of Illustrations

Illustration 1: Implementation of the "Sequence" pattern.....	5
Illustration 2: Implementation of the "Parallel Split" pattern.....	5
Illustration 3: Implementation of the "Synchronization" pattern.....	6
Illustration 4: Implementation of the "Exclusive Choice" pattern.....	7
Illustration 5: Implementation of the "Simple Merge" pattern.....	8
Illustration 6: Implementation of the "Multi-Choice" pattern.....	9
Illustration 7: Implementation of the "Structured Synchronizing Merge" pattern.	10
Illustration 8: Implementation of the "Multi Merge" pattern	10
Illustration 9: Implementation of the "Cancelling Discriminator" pattern.....	12
Illustration 10: Implementation of the "Generalized AND-Join" pattern.....	15
Illustration 11: Implementation of the "Local Synchronizing Merge" pattern	16
Illustration 12: Implementation of the "Thread Merge" pattern.....	17
Illustration 13: Required properties settings for the "Thread Merge" pattern	18
Illustration 14: Implementation of the "Thread Split" pattern using a basic activity and the "Completion Quantity" value.....	19
Illustration 15: Required properties settings for WCP 42	19
Illustration 16: Implementation of the "Thread Split" pattern using a MI activity.	20
Illustration 17: Implementation of the "Multiple Instances without Synchronization" pattern.....	21
Illustration 18: Menu for the right click on an activity.....	21
Illustration 19: Properties window for an activity with the required settings.....	22
Illustration 20: Implementation of the "Multiple Instances with a Priori Design-Time Knowledge" pattern.....	23
Illustration 21: Required properties settings for WCP 13	23
Illustration 22: Implementation of the "Multiple Instances with a Priori Run-Time	

Knowledge" pattern.....	24
Illustration 23: Required properties settings for WCP 14.....	25
Illustration 24: Implementation of the "Deferred Choice" pattern.....	28
Illustration 25: Implementation of the "Interleaved Routing" pattern using an Ad-Hoc activity	30
Illustration 26: Implementation of the "Cancel Task" pattern.....	31
Illustration 27: Implementation of the "Cancel Case" pattern.....	32
Illustration 28: Implementation of the "Cancel Region" pattern	33
Illustration 29: Implementation of the "Cancel Multiple Instance Activity" pattern	33
Illustration 30: Implementation of the "Complete MI Activity" pattern.....	34
Illustration 31: Implementation of the "Arbitrary Cycles" pattern.....	35
Illustration 32: Implementation of the "Structured Loop" pattern	36
Illustration 33: Implementation of the "Structured Loop" pattern with Post Evaluation.....	36
Illustration 34: Required properties for the realization of the "Structured Loop" pattern with Post Evaluation	37
Illustration 35: Alternative Implementation of the "Structured Loop" pattern with Post Evaluation.....	37
Illustration 36: Implementation of the "Structured Loop" pattern with Pre Evaluation.....	38
Illustration 37: Required properties for the realization of the "Structured Loop" pattern with Pre Evaluation	38
Illustration 38: Alternative Implementation of the "Structured Loop" pattern with Pre Evaluation.....	39
Illustration 39: Implementation of the "Implicit Termination" pattern	40
Illustration 40: Implementation of the "Explicit Termination" pattern.....	41
Illustration 41: Implementation of the "Persistent Trigger" pattern.....	42
Illustration 42: BPMN 1.1 and the subject oriented approach	45