

Albert Fleischmann
Stefan Raß
Robert Singer

S-BPM Illustrated

A Storybook about Business Process Modeling and Execution

Springer Open

www.dbooks.org

S-BPM Illustrated

Albert Fleischmann
Stefan Raß
Robert Singer

S-BPM Illustrated

A Storybook about Business Process Modeling and Execution

Springer

www.dbooks.org

Albert Fleischmann
Pfaffenhofen, Germany

Robert Singer
Graz, Austria

Stefan Raß
Graz, Austria

ISBN 978-3-642-36903-2 ISBN 978-3-642-36904-9 (eBook)
DOI 10.1007/978-3-642-36904-9

Library of Congress Control Number: 2013937512

ACM Computing Classification (1998): J.1, H.4, K.6

Springer
Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.
© The Editor(s) (if applicable) and the Author(s) 2013.
The book is published with open access at SpringerLink.com
All commercial rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher’s location, in its current version, and permission for commercial use must always be obtained from Springer. Permissions for commercial use may be obtained through Rightslink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Printed on acid-free paper.

Springer is part of Springer Science+Business Media
www.springer.com

S-BPM stands for "subject-oriented business process management" and focuses on subjects. Subjects represent the entities (people or programs etc.) who are actively engaged in processes.

S-BPM has become one of the most widely discussed approaches for process professionals. Its potential particularly lies in the integration of advanced information technology with organizational and managerial methods to foster and leverage business innovation, operational excellence and intra- and inter-organizational collaboration. Thus S-BPM can also be understood as stakeholder-oriented and social business process management.

S-BPM as a discipline is characterized by a straightforward approach towards the analysis, modeling, implementation, execution and management of interaction patterns with an explicit stakeholder focus.

Institute of Innovative Process Management, www.i2pm.net, 2013.

www.dbooks.org

Preface

This book shows how subject-oriented business process management (S-BPM) and its tools can be used in order to solve communication and synchronization problems of humans and/or machines in an organization.

This is a hands-on book. All the activities which are necessary to implement a business process are shown step by step. We start with analyzing the problem, continue with modeling and validating the corresponding process, and finish off by embedding the process into the organization. The final result is a workflow which executes the process without requiring any programming. In the first step a very simple process is implemented. This process is extended and improved in "adaption projects", because additional problems have to be solved. This reflects reality where processes must always be changed and adapted to new requirements.

If you want to execute all the steps by yourself you can download the tool suite from the www.i2pm.net website. If you want to get more background information about S-BPM you can find it in the book "Subject-Oriented Business Process Management" which is available as a Springer Open Book. You can download it from Springers website[1] for free.

There are many people in the background who helped in the production of this book. In particular, the authors wish to thank Metasonic AG for allowing use of their BPM suite, Udo Kannengiesser for proofreading the manuscript (all remaining errors are the authors), and last but not least Ralf Gerstner of Springer Verlag for his support and cooperation.

Graz, March 2013

Albert Fleischmann
Stefan Raß
Robert Singer

[1] http://link.springer.com/book/10.1007/978-3-642-32392-8/page/1

Contents

1 **Introduction** 1
1.1 **Business Processes and Business Process Management** 1
1.2 **Taylorism, Fordism, and Post-Fordism** 2
1.3 **Communication instead of Central Control** 4

2 **The Problem – Part I** 7
2.1 **Do you Know this?** 7
2.1.1 About Communication … and other Troubles 7
2.1.2 Daily Quarrel in the Factory 8
2.1.3 The Solution? 10
2.2 **What is S-BPM?** 11
2.3 **The Workshop** 14

3 **The Solution – Part I** 21
3.1 **Summary of the Problem** 21
3.2 **Solution – Step by Step | Scenario I** 21
3.2.1 Creating a Project 21
3.2.2 Creating a Process 22
3.2.3 Modeling the Process – Subject Communication 24
3.2.4 Modeling the Process – Internal Behavior 30
3.2.5 Enacting and Executing the Process 49
3.3 **Accomplishments** 65
3.4 **Lessons Learned** 65

4 **Transition – Part I** 67

5 **The Problem – Part II** 69

6 **The Solution – Part II** 77
6.1 **Summary of the Problem** 77
6.2 **Solution (Step by Step)** 77
6.2.1 Copying the Process 77
6.2.2 Altering the Process 80
6.2.3 Altering the Internal Behavior 83
6.2.4 Executing the Process 92

www.dbooks.org

6.3 **Accomplishments** ... 97
6.4 **Lessons Learned** ... 98

7 **Transition – Part II** ... 99

8 **The Problem – Part III** ... 101

9 **The Solution – Part III** ... 105
9.1 **Summary of the Problem** ... 105
9.2 **Solution (Step by Step)** ... 105
9.2.1 Copying the Process ... 105
9.2.2 Altering the Process ... 106
9.2.3 Altering the Internal Behavior ... 109
9.2.4 Customizing the Process ... 118
9.2.5 Executing the Process ... 118
9.3 **Accomplishments** ... 124
9.4 **Lessons Learned** ... 124

10 **Transition – Part III** ... 127

11 **The Problem, the Solution and the End – Final Part** ... 129

12 **Troubleshooting** ... 139

The Institute of Innovative Process Management ... 141

S-BPM ONE Conference Series ... 143

Introduction

Subject-oriented Business Process Management (S-BPM) is different from current BPM approaches. In this chapter we want to explain what processes and Business Process Management are about and on which hidden paradigms current BPM approaches are based. Then we show how S-BPM is different to most of these approaches.

1.1 Business Processes and Business Process Management

In modern days, no successful company without processes exists. Large companies may even have hundreds of different processes. These processes can be remarkably simple with only one or two participants or highly complex with a dozen or even hundreds of participants. Processes use the company's resources to produce a desired output that is of value for the company or its stakeholders (i. e., customers). This output, for instance, can be a service or a product (technical or otherwise). It is very important for companies to keep their processes as effective and efficient as possible; this is ensured through the use of Business Process Management (BPM). BPM uses many different methods and tools to identify, control, and improve a company's processes.

A process is a structure consisting of logically connected tasks, operators, material expenses, and information. This includes a chronological, geographical, and quantitative definition. A process has a defined launch event (input) and result (output) with the goal of producing something of value for customers. The sum of all processes is the process organization.[1]

Processes must be continuously adapted to changing business environments. This should be done in a structured and well-defined way. This activity is called business process management which is, according to Fischermanns[2] and Roger T. Burlton[3], a process in itself. This process has to be managed and controlled, to ensure continuous improvement of the organization's performance (and therefore success). In Business Process Management the following activity bundles have to be executed:

- Analyze a process
- Model a process
- Validate a process
- Optimize a process
- Embed a process into the organizational structure
- Embed existing IT-Solutions into a process
- Run and monitor instances of a process

[1] Dr. G. Fischermanns: Praxishandbuch Prozessmanagement, 6. Auflage, Gießen: Verlag Dr. Götz Schmidt 2006, p.12

[2] Dr. G. Fischermanns: Praxishandbuch Prozessmanagement, 6. Auflage, Gießen: Verlag Dr. Götz Schmidt 2006, p.26f.

[3] Roger T. Burlton: Business Process Management, Profiting from Processes, USA: Sams Publishing 2001

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_1, © The Author(s) 2013

www.dbooks.org

Normally these activities are not strictly executed in that order. If deficiencies are discovered in a process model you can go back either to *analyzing a process* or *modeling a process*.

Current BPM approaches are still heavily influenced by Scientific Management proposed by F. W. Taylor[4] and Fordism developed by the Ford Motor Company[5]. In the following sections we want to show that Taylorism and Fordism are still the unspoken paradigms underlying "modern" business process management.

1.2 Taylorism, Fordism, and Post-Fordism

Taylor began by analyzing work systematically. He wanted to replace the "rules of thumb" used for organizing work with a systematic scientific approach. The major aspects of Taylor's Scientific Management are described in his article "The Principles of Scientific Management" (see footnote 4):

> Under the old type of management success depends almost entirely upon getting the "initiative" of the workmen, and it is indeed a rare case in which this initiative is really attained. Under scientific management the "initiative" of the workmen (that is, their hard work, their good-will, and their ingenuity) is obtained with absolute uniformity and to a greater extent than is possible under the old system; and in addition to this improvement on the part of the men, the managers assume new burdens, new duties, and responsibilities never dreamed of in the past. The managers assume, for instance, the burden of gathering together all of the traditional knowledge which in the past has been possessed by the workmen and then of classifying, tabulating, and reducing this knowledge to rules, laws, and formulae which are immensely helpful to the workmen in doing their daily work. In addition to developing a science in this way, the management take on three other types of duties which involve new and heavy burdens for themselves. These new duties are grouped under four heads:
>
> First. They develop a science for each element of a man's work, which replaces the old rule-of-thumb method.
>
> Second. They scientifically select and then train, teach, and develop the workman, whereas in the past he chose his own work and trained himself as best he could.
>
> Third. They heartily cooperate with the men so as to ensure all of the work being done in accordance with the principles of the science which has been developed.
>
> Fourth. There is an almost equal division of the work and the responsibility between the management and the workmen. The management take over all work for which they are better fitted than the workmen, while in the past almost all of the work and the greater part of the responsibility were thrown upon the men.

[4] Taylor, Frederick Winslow (1911), The Principles of Scientific Management, New York, NY, USA and London, UK: Harper and Brothers, LCCN 11010339, OCLC 233134. Also available from Project Gutenberg

[5] An overview and references can be found at http://en.wikipedia.org/wiki/Fordism last access January 2013

Taylor's scientific management is a business process management system which means he is describing a way to identify effective and efficient production steps or sequences of production processes. These work plans are developed and described by management (white-collar workers) and executed by blue-collar workers. Taylor does not elucidate how the sequences of actions are described or how their execution is supported generally.

Independently of Taylor, the Ford Motor Company focused on the aspect of how succeeding work steps executed by different blue-collar workers could be organized in an effective and efficient way. For this purpose Henry Ford introduced assembly lines.[6]

> The first step forward in assembly came when we began taking the work to the men instead of the men to the work. We now have two general principles in all operations–that a man shall never have to take more than one step, if possibly it can be avoided, and that no man need ever stoop over. The principles of assembly are these:
>
> 1. Place the tools and the men in the sequence of the operation so that each component part shall travel the least possible distance while in the process of finishing.
> 2. Use work slides or some other form of carrier so that when a workman completes his operation, he drops the part always in the same place–which place must always be the most convenient place to his hand–and if possible have gravity carry the part to the next workman for his operation.
> 3. Use sliding assembling lines by which the parts to be assembled are delivered at convenient distances.

In the 1970s several market changes occurred. A general saturation of consumer markets had major impacts on mass production. Increased competition from new markets (especially Southeast Asia) due to globalization, made the old system of mass producing identical, cheap goods through division of labor uncompetitive. Additionally, more individual and specialized products were required by consumers. The development of information and communication technology allowed work to be organized in a totally new way. This period of time is called Post-Fordism[7]. According to S. Hall Post-Fordism is characterized by the following attributes:

- new information and communication technologies
- more flexible, decentralized forms of labor process and work organization
- decline of the old manufacturing base and the growth of the "sunrise" computer and communication industry
- the contracting out of functions and services
- more specialized products
- emphasis on types of consumers in contrast to previous emphasis on social class
- the rise of the service and the white-collar worker, and a declining need for unskilled workers
- the feminization of the work force

[6] Ford Henry, My Life and Work, available from Gutenberg Project, http://www.gutenberg.org/cache/epub/7213/pg7213.txt

[7] S. Hall, Brave new World, Marxism today, October 1988

www.dbooks.org

Mass marketing was replaced by flexible specialization, and organizations began to emphasize communication rather than command.

1.3 Communication instead of Central Control

The principles developed for production systems were transferred into the world of administration and became the paradigm for BPM. Business processes define how an organization reacts to business events like a customer order, customer complaints, supply chain events etc.

Today most BPM approaches are still based on Taylor's and Ford's principles. In BPM there are specialists mainly from consulting companies who evaluate the current processes and define "better" ones (white collars). These processes are evaluated by the people who have to execute these processes (blue collars).

Most process specifications are based on control flow diagrams enhanced with swim lanes, events, connectors (and, or) etc. Control flow diagrams are like abstractions of assembly lines. The activities in a control flow diagram correspond to workplaces in an assembly line. The transportation activity of an assembly line is like the execution of a control flow diagram. This is mainly done by computers. The software used for this is called a workflow system.

This paradigm does not fit with the properties of post-Fordism. In today's service industry, people executing activities in knowledge-intensive service processes are highly qualified. Normally they know best how they should do their job. Service processes must be executed very flexibly and therefore a lot of communication is necessary between the people. Because they are highly qualified people want to define their work by themselves, and this self-empowerment is essential for their motivation. They do not accept a strong central control. Because of division of work different people in different organizations must work together. In such situations there is no institution that controls the required cooperation.

The parties involved in a process communicate in human-centered workflows. This is where Subject-oriented Business Process Management (S-BPM) comes into play. It marks a paradigm shift from the flow-oriented execution of activities to a communication-based view of subjects interacting as active parties in a process. S-BPM directly involves participants in the design of their processes. Because of an easy-to-understand graphical notation based on natural language (subject, predicate, object) the domain experts can model their processes by themselves. They describe their individual view of their task by specifying three activities: receiving information from others, sending information to others, and perform functions. As the resulting models are based on a clear and unique formal, and thus executable, logic[8] the process participants can evaluate and modify them on the fly. These properties of the S-BPM approach allow the decentralized, self-organized design of work patterns as it fits to Post-Fordism and modern organizational theory. Nevertheless, the subject-oriented approach to BPM also supports the traditional flow-oriented way of designing processes if necessary. This is possible, because central control is just a special case of communication, where interactions are kind of "hard-wired." This means, contrary to

[8] page 315 in A. Fleischmann, W. Schmidt, C. Stary, S. Obermeier, E. Börger; Subject Oriented Business Process Management, Springer 2012

traditional concepts, S-BPM covers both, communication-oriented and flow-oriented processes. In the following chapters we elaborate the concept and its features using a real-world example. In that example not all concepts of S-BPM are applied, we focused on the practical use of the most important aspects. All the concepts are described in the already-mentioned book (see footnote 8).

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

www.dbooks.org

The Problem – Part I

The soul never thinks without a picture. *Archimedes*

This chapter tells a short story about typical situations in production companies. You can skip this chapter at any time and go to the next chapter to directly work through the examples. Nevertheless, the story defines the context of the step-by-step examples.

2.1 Do you Know this?

2.1.1 About Communication … and other Troubles

John Doe, Operations Manager

It was just another morning of a typical working day for John Doe, who was one of the Teaching Factory's operations managers. John took a look at today's production schedule. Even though one of his workers, responsible for quality inspection, was on vacation, today's goal for the production schedule would easily be met.

Or, so he thought. Suddenly he received an e-mail from one of his workers, saying that he was not able to come to work that day, due to illness, but assuring he would come back tomorrow. Upon reading the name of the worker who called in sick, it struck John like a lightning bolt: it was another quality inspection worker. After closely examining that day's production schedule, John quickly realized that for today's production order the absence of his quality inspection workers would lead to a serious bottleneck in quality assurance, which would delay the order by at least one day and also lead to a lot of semifinished items being stuck at the quality inspection workbench. John was getting nervous. If the production of today's order would take two days instead of one, the production schedule for the whole week was screwed. This would anger his superiors because he knew of at least one customer deadline, that would be violated by this delay.

He anxiously looked over the production schedule to find a solution for this situation. After examining the schedule for the next day, he realized that the solution for this problem was relatively easy. The items to be produced the next day had a much less complicated quality check than today's items, and thus could be done in a shorter time. If he just could swap the production schedule from today with the one for the next day, all problems would be solved because the missing worker would be back – therefore no time would be lost, and the production schedule for the week would also be met. John took a look at the clock on his office wall. He still had one hour left to switch production schedules.

The only thing he had to do to switch production schedules was to tell the logistics department – whose task was to deliver the raw material for production – to deliver the parts for tomorrow's production schedule and not for today's. This was essential because the items to be produced today and tomorrow required different components. John immediately sent an e-mail to all managers of the logistics department to notify

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_2, © The Author(s) 2013

them about the situation and of his proposed solution. Because of the importance of the matter he asked for immediate confirmation.

We have here an example of so-called "unstructured communication."

But after 15 minutes, which felt like 15 hours, still nobody had answered his e-mail. There were only 45 minutes left and so he decided to make a phone call. He called each manager from the logistics department, but nobody picked up the phone.

With only 30 minutes left he ran over to the logistics department, which was in a different building, only to find out that there was no manager present and also none of the employees knew where they went.

Exhausted, angry and defeated, John walked back to his office and, from his office window, watched the impending disaster, which he was unable to prevent, unfold. For a moment he wondered how high the agreed penalty for the delayed delivery would be and was angry, because he now had to take the blame for something upon which he had absolutely no influence. It was not his fault that one of the orders was calculated so scarcely and it also was definitely not his fault that none of the responsible persons in the logistics department could be reached. It was always the same with those logistics people. If you needed them, they let you down. John had to learn that from experience. Logistics was in no means flexible or reliable, he thought. And now he had to take the blame for that. And he hates not getting a response to his mails – sometimes he had the feeling mails were disappearing in a sort of digital black hole; they never come back. Or, in the best case you get an answer too late (see illustration in Fig. 2.1).

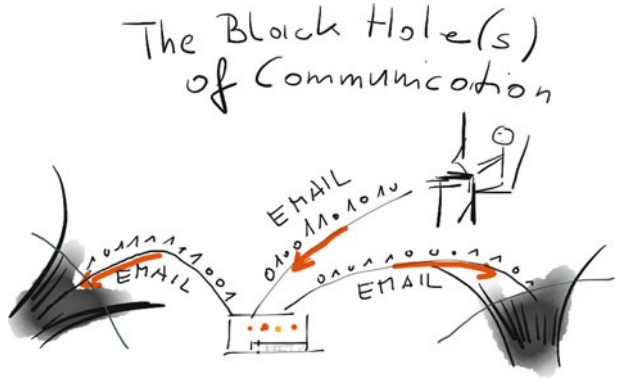


Fig. 2.1 E-mails are a very flexible and convenient way to communicate, but you do not know what will happen with them. The drawback of this flexibility is, that it is an unstructured form of communication. For example, there is no defined time to answer, and therefore no defined throughput time for the communication

2.1.2 Daily Quarrel in the Factory

Later that day, at noon, John decided to have lunch in the company's canteen. After clearing his head – while walking there – he took his meal from the self-service counter and looked for a place to sit. Soon he noticed Norma Roe, one of the logistics department's managers, sitting alone at a table, having lunch. Now upset again, he walked over and took a seat to have a serious talk with her.

Norma Roe, Logistics Manager

"Seriously, Norma," he said, trying to calm his voice. "What on earth is wrong with you?"

Surprised, she looked up.

"I sent you an email, I called you three times and I even ran over to your department, only to see nobody was there! We had real trouble today and I really would have appreciated your support!"

www.dbooks.org

"Sorry, John" she answered. "But we had an important strategy meeting today, which each executive in our department had to attend. I just finished that meeting to have lunch … I did not even have time to check my phone or my mails."

"But I really needed someone from your department today! It's always the same. Every time I need anything from your department you either say that you can't help me or don't react at all; then I am made responsible for your actions!"

"Calm down, John. I know that our department has communication issues, but there is no need to make me responsible for everything …"

Jerry Smith, CEO

The conversation went on like this for several minutes until the CEO, Jerry – who was having lunch at a table nearby – stood up and approached them. He sat down at their table and said "You two really need to make peace with each other. This can't go on like this, with your two departments always fighting over something. So tell me your problem."

Jerry took a napkin from the table and took out a pen from his jacket. As John and Norma told him their problem, Jerry started to draw on the napkin. After a few minutes all three stared at the sketch. As many people, Jerry saw himself as a bad illustrator, but he knew that visualization was an important tool for solving problems.

"So, I don't want to hear anymore excuses." Jerry said after some minutes of silence. "You will need to get down to the root of the problem, which, as far as I can tell, is related to the way you communicate, when I see my simple illustration. Listen, this year I was in Deggendorf, a small city in the middle of nowhere in Bavaria, at a convention called S-BPM One …"

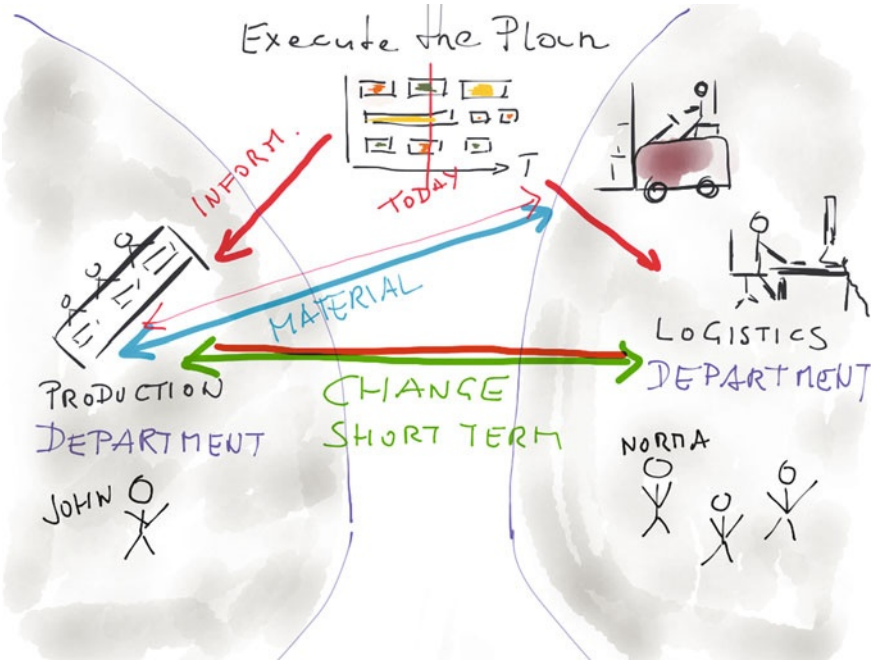


Fig. 2.2 Jerry's napkin illustration

2.1.3 The Solution?

Jerry spent the rest of the lunch break talking about this convention in Vienna and about an interesting concept he had heard there: *subject-oriented business process management (S-BPM)*. He told them about several talks and workshops where this method was explained and discussed.

"You know, we have all had our troubles with business process management. We define the processes, we forget them – that is the reality. And we needed a lot of time to define them. But as you can see, without well-defined *and* enacted processes, you are always in trouble – firefighting and finger-pointing all day!"

S-BPM is a methodology to enable structured communication, i. e., to model and enact business processes.

S-BPM as a method seems to focus on the actors – the subjects – who take part in a process which was an interesting new concept for Jerry. He also explained how the focus of S-BPM seems to be on the communication between all the actors – a pattern he could also see when looking at the problems of John and Norma. John also found this new methodology very interesting and listened closely, while Norma said that in her opinion this looked just like a marketing strategy to sell something. It was easy to promise heaven on earth by just claiming to solve all communication issues, but this was something also a lot of other consulting companies did.

John, however, was very interested and wanted to learn more about this method. "You know, Norma," he said, "In the best-case scenario this could really help to solve our problems and improve our life at work. In the worst-case scenario I just waste a few hours of my work time."

Jerry agreed to send John all the information he had via e-mail and the three of them went back to work.

After John worked through some tasks for the day, he decided it was finally time to start a project with this S-BPM thingy. He decided to informally call it "Project X" – this sounded cool and mysterious and he could rename it if it actually started to look promising. After having received Jerry's e-mail he created a new folder called "Project X" on his work PC and started to gather some information about the matter.

After researching for several hours, John began to understand that S-BPM seemed to be more powerful than he would have thought. Because of the fact that his time was limited and also very valuable, he decided to shorten the process. While doing his research he found a company nearby which offered to do teaching and consulting for S-BPM. He decided to contact this company and to listen to what they had to offer; as an executive he didn't want to spend myriads of hours on research if he could spend some of his department's budget to pay somebody who told him what he wanted to know.

He decided to phone the external expert and the phone call was very promising. Instead of trying to explain the concept of what they did on the phone, the expert offered to come over with a colleague to actually show him what they did. John liked the idea, and after making an appointment with him, he also invited Norma to this event and insisted that she at least listened to what they had to say.

www.dbooks.org

Finally, the arranged day came and John, Norma, and the two business process management experts Bob and Al met in one of the meeting rooms. After everyone introduced themselves Bob and Al immediately started their presentation. To John's amazement they didn't start with a product presentation to sell him a product – what he had expected – but started with an explanation of the concept behind *subject-oriented process management*. The experts said that it made no sense to sell a solution if the customer doesn't understand the concept behind it enabling him to successfully use it.

2.2 What is S-BPM?

"So" said one of the experts, "let's see, what is the central idea behind the S-BPM concept?" "To work it out together: you said that you have defined, i. e., documented, business processes. Why do you think these processes do not support you in some situations?"

Norma didn't need to think about this question and replied immediately – with a somewhat cynical smile on her face – "they are far too simple to be helpful if something does not work as planned. I was part of the modeling team and when we designed the processes we could not include all thinkable situations and therefore we decided to keep it simple." She laughed and said "Somebody insisted we call it lean."

"Yes, I understand" one expert said, "we call this the happy path. How often do you think you execute processes according to the happy path?"

"Seldom," Norma and John replied at the same time.

Both experts had to laugh. "Really, we hear this often," Al replied. "But we are convinced that real customer satisfaction and profitability can only be gained, if business processes are designed to handle most cases. Nevertheless, as long as people are key elements in the processes they can handle any situation, but mostly with firefighting – a predominant culture in manufacturing companies. And … firefighting is not efficient, and it does not lead to a comfortable working environment at all."

"Another aspect of undefined processes is that the company has to rely on the personal experience of the people in charge. Defining some guideline for the main actors is a sort of applied knowledge management. The mistake most process designers make is that they try to define actions on an atomic level – this is not necessary in all cases. Sometimes we only need to define who has to communicate what with whom."

One of the experts, Al, went to the flip chart, flipped the last sheet over and started to draw (Fig. 2.3). "Look, in principle we have to concentrate on the people in the company."



Fig. 2.3 Al starts drawing on the flip chart

Then he drew two boxes connected by two arrows, one from left to right, one from right to left (Fig. 2.4). "Let's say, the boxes are the main actors, then they synchronize their work through communication, that means exchanging messages."

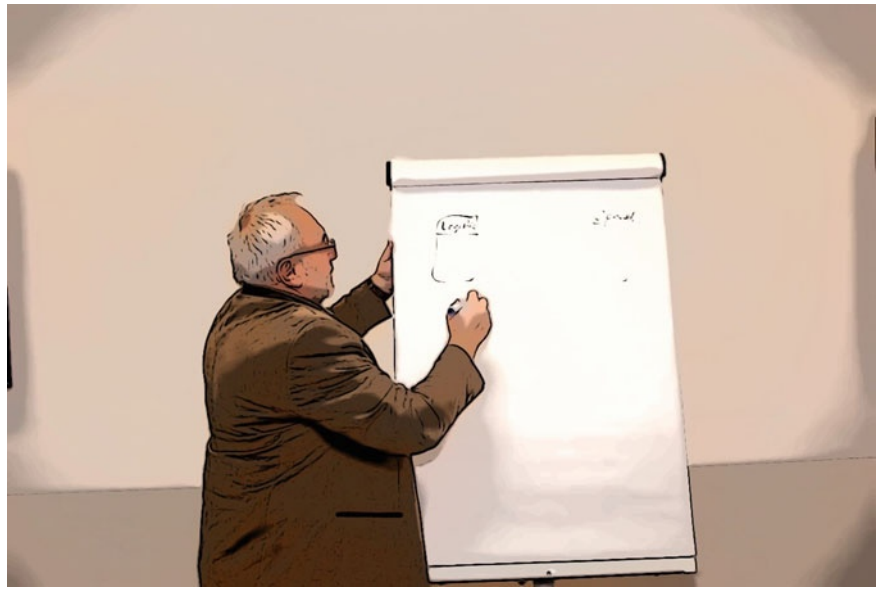


Fig. 2.4 The flip chart already shows the two subjects

Synchronize (define) work through the exchange of messages.

The other expert, Bob, stood up and went to the flip chart (Fig. 2.5). "And" he started, "This concept is even more general. The boxes symbolize the subjects in the process, but they are not necessarily people in the company. It can even be a software system or an external subject, such as a customer or a supplier. This concept is universally applicable – synchronize work through the exchange of messages."

www.dbooks.org



Fig. 2.5 Bob continues to explain the concept

"That's fine, but how do we know what a 'subject' has to do?" John replied with a somewhat skeptical tone (Fig. 2.6).



Fig. 2.6 John is sceptical

"That is the next level of detail," Al replied. "So we define the so-called internal behavior of the subjects. And now comes the surprise, this can be done with only three more symbols."

Every subject can be in one of three states: wait, send, do something.

Al started to draw on the flip chart. He drew the three symbols (Fig. 2.7). "We can describe the behavior of the subjects using one symbol for waiting for message, one symbol for sending a message, and one for doing something. That is the main concept. As always we can think about sugaring these symbols for ease of use, but this is not necessary."



Fig. 2.7 S-BPM uses only a few symbols for modeling

Formal models (as S-BPM models) can be enacted on software systems.

Finally, Bob stated: "The benefit now is that behind this concept of defining processes in such a human-centric way, we also have a formal mathematical theory. That means, we can easily model processes together with the people working in the process and the model can be directly executed on a software platform to support the communication and the management of documents or data."

"Now," Bob continued, "fill your cups with coffee and let's model your process."

2.3 The Workshop

John started to explain the whole issue once more: "As I said before, I am one of this company's operations managers. My task is to fulfill the production schedules for the week."

"I am one of our logistics managers and I am responsible for ensuring the availability of the raw materials," Norma continued.

John went on explaining the issue: "on a normal working day, I conduct the production schedules and try to fulfill them. This can only be done with the help of the logistics department; they deliver the production material directly to our department. Usually, this works pretty well. You know, our company is ISO-9000 certified and we have defined and documented processes, everything is principally well defined. Problems arise whenever situations pop up which are not defined in the processes."

"Just to give you the latest example: some time ago I had a very nice production schedule which we would have met easily – if not for one of my workers calling in sick

www.dbooks.org

shortly before production was due to start. This caused a bottleneck at quality inspection, with the consequence that the schedule could no longer be met. It would have been so easy to solve this issue – I could have just swapped schedules and there wouldn't have been a problem. But, to do that, I would have had to notify the logistics department, meaning Norma or one of her colleagues, to deliver the right production material in time. We need to agree on such issues, because as the production department we are unaware of any issues on the purchasing and logistics side that may affect the decision. It is just that I could not reach any of them – so the wrong material was delivered, the schedule was not met, my boss was very angry with me and I was very angry with the logistics department …"

Norma went on: "And this is just one example – such situations happen far too often, and if they happen, they cause a lot of frustration. I remember the situation John just mentioned – we had a real quarrel over that one."

The experts looked at each other and one of them said "Okay, that was a very interesting description of your problem. I see that there are also a lot of emotions involved here."

The other expert then tried to clarify the situation: "So, to sum up, you seem to be having a communication issue here, did I understand the situation correctly? If you had been able to communicate with the logistics department in a more or less structured way, none of that would have happened."

"Well, yes, that's true I guess …and it should be supported by technology," John answered. "So I assume, if somebody is not available, he or she can delegate certain process activities to somebody else. That is far better than having such production problems."

"Okay," one of the experts said, "the first step would be to identify the people who are part of this process – we call them subjects, as you know."

"Aren't you supposed to tell us what you think the subjects are?" Norma asked. "No," the expert replied. "We want to work out the solution together with you. That way you will learn more and in the end you will also be able to create the solutions on your own, instead of always being dependent on consultants like us. We don't know your business, we only know the methodology."

John, who was thinking about the expert's question, raised his voice and said: "I think I got it …I think there are two subjects interacting here: Me, as an operations manager, and Norma, as part of the logistics department."

As he finished talking, the other expert took out a flip-chart marker, painted two rectangles and replied: "Yes, that's a good start."

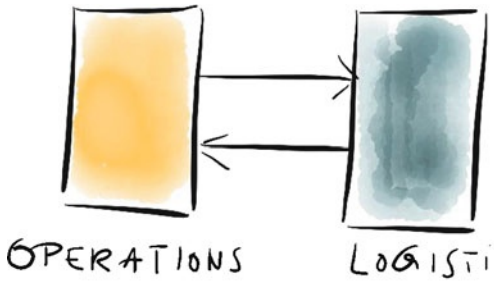


Fig. 2.8 Two communicating subjects. Basic set of messages: one question and one answer

Then it was the other expert's turn again: "And what do you think, what messages, what communications should the two subjects exchange in case such an exceptional

event, which is not defined in the standard processes, occurs? What would you call these messages and who sends them?"

After a short discussion, Norma and John agreed that the first message would be a "request for change," which would be sent from the operations manager to a manager of the logistics department.

They also agreed that there were several possible responses: either the logistics department would send back an approval, in which case everything was fine. Another possibility was that the logistics department would not answer at all – in which case an alternative should exist.

"You know, like, hiring extra workforce just for the day or something like that," John added. If no alternative existed, one should try to reach someone in the department by any means possible.

"I think you forgot one case," one of the experts added after the two had finished explaining. "What, if the logistics department declines the request for some reason?"

"You are right … we would have to consider that. What do you think, Norma? Would they just send back a declination?"

"Hmm. I think that would cause the process to become unnecessarily bloated. Wouldn't it be better if, in case of a declination, the logistics department would call the respective operations manager and work out a solution?"

"Hey, yes, you are right – that is a way better idea. Of course, after the phone call or whatever means of communication were used, the logistics department would have to send a confirmation that they agreed upon a change."

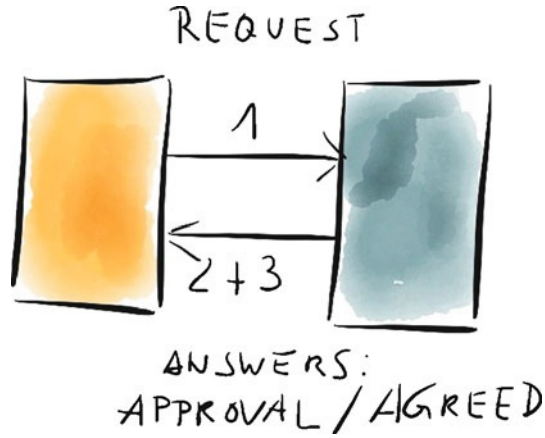


Fig. 2.9 Two communicating subjects. Complete set of messages

After that, one of the experts started to visualize what they were speaking about while the other one tried to sum up: "So, there are three messages: the 'request for change' message, which the operations manager sends, the 'approval' message from the logistics department, and an 'agreed upon a change' message, also from the logistics department."

After that, the four looked at the flip chart for a while, and found that this covered all situations they could think of.

The two experts looked at each other and one of them said "Okay … I think that now, after our discussion of the problem, we are finally ready to create a running solution."

"What?" John asked, "Here and now?"

www.dbooks.org

"Yes," the expert replied. We will show you what you can achieve with a product called *Metasonic Suite* and how you could transform this theoretical discussion and these flip charts into a real and working solution which you could use in your production environment."

"I don't know," Norma said sceptically. "This just sounds too easy to be true!"

The experts replied: "Just wait and see …" The following pictures give an impression of the discussions in the modeling workshop.



Fig. 2.10 Al, Bob, and John prepare to start modeling



Fig. 2.11 John's first shock because of the capabilities of S-BPM and the Metasonic Suite



Fig. 2.12 Norma joins the modeling team while Bob explains important stuff



Fig. 2.13 Norma and John watch Bob and Al modeling the process they previously discussed

www.dbooks.org



Fig. 2.14 John finally begins to understand the concept and the application

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

The Solution – Part I

Speed is the essence of war. *Sun Tzu*

3.1 Summary of the Problem

This chapter sums up the story presented in the last chapter, so it is possible to skip the last chapter if you wish.

The first scenario depicts a problem between an operations manager and a person from the logistics department; this means that in this case there are two actors. The operations manager is responsible for his team meeting certain production schedules and coping with any problems that may occur.

The logistics department is responsible for delivering production material in time so the production schedule can be met.

A problem arises when there is an exceptional process situation that may endanger on-schedule performance of production operations. In this case the operations manager has to reschedule production and inform the logistics department so production material is delivered when needed.

The challenge, which will be addressed in this scenario, is the communication between the operations manager and the logistics department, because there is a lot of room for error in communication, which could lead to massive production problems.

After reading this scenario, it should be clear how to create a new process group with the *Metasonic Suite* which is able to support solutions for the problems described.

3.2 Solution – Step by Step | Scenario I

3.2.1 Creating a Project

Open the *Metasonic Suite* by double-clicking on the desktop symbol.

After starting the application, the dialog window *Workspace Launcher* eventually pops up. There you can define the location in which your project files will be stored. Activate the check box *Use this as the default and do not ask again*, if you do not want to see this dialog again. Press the button *OK*, when finished. If you start the application for the first time a welcome window is shown. Close this window; if you wish to see this window again simply choose *Welcome* in the *Help* menu.

To start a new project, choose the menu entry *File/New/Process Group*. In the opened dialog window *Metasonic Build Process group* enter a name in the field *Process group name*. For our purposes please enter "Teaching Factory goes S-BPM" and click on the button *Finish*. By default the standard work space location – as defined during application start-up – of the *Metasonic Suite* is used. After having created a process group, it is displayed on the left-hand side in the *Navigator* window (see Fig. 3.1).

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_3, © The Author(s) 2013

www.dbooks.org

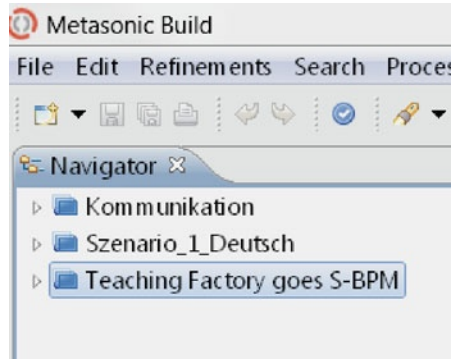


Fig. 3.1 Viewing the new *Process Group*. Depending on the chosen work space, different process groups are shown

3.2.2 Creating a Process

Of course you could also specify another name.

Now, that a process group has been defined, it is necessary to create a new process within this group. A process group is able to store several processes. Right-click on src in the *Navigator* window on the corresponding process group "Teaching Factory goes S-BPM" and select *New/Process* from the context menu, as shown in Fig. 3.2.

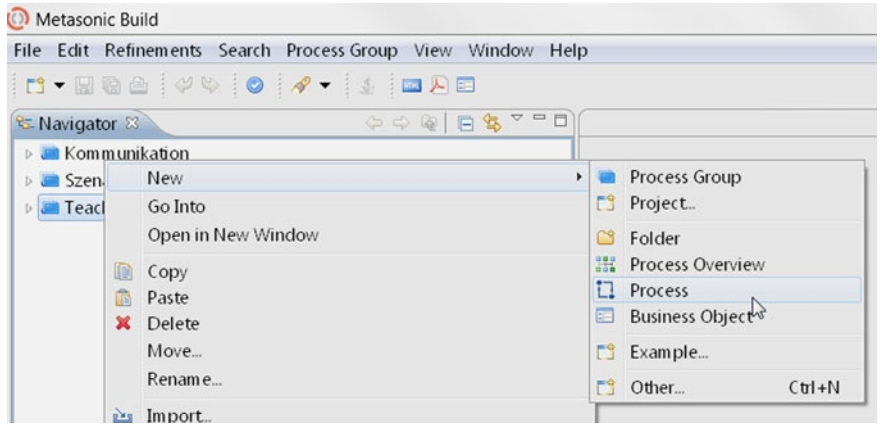


Fig. 3.2 Creating a new process by using the menu

The chosen name is automatically converted into a machine readable format.

You should always give the processes meaningful names. In this case the name is "Scenario 1 – Operations Manager and Logistics", as shown in Fig. 3.3. Press the button *Finish*, when done.

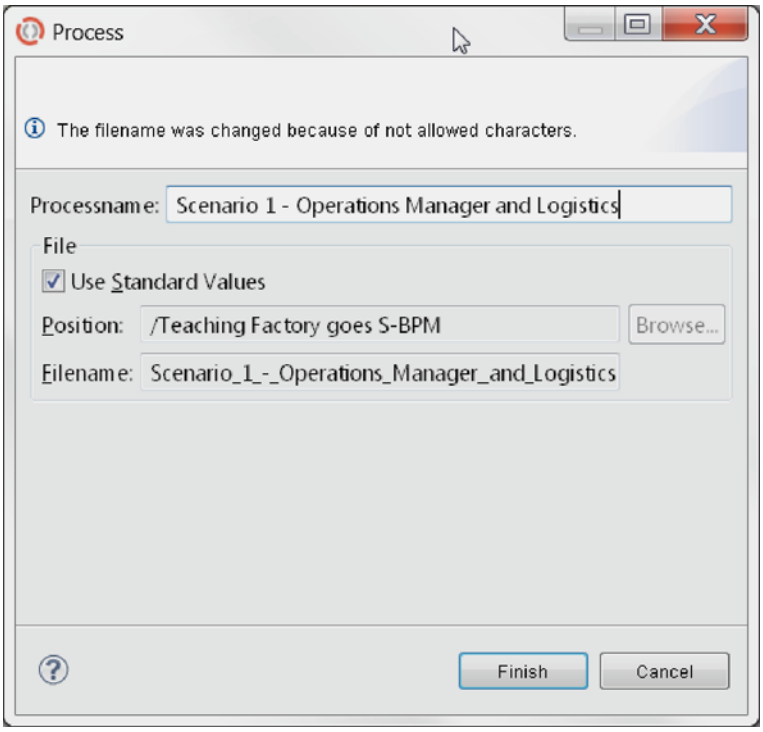


Fig. 3.3 Dialog window *Process*: create and name a new process. Leave the check box *Use Standard Values* checked

The application window automatically displays the modeling space for the newly created process. All processes of a process group can be chosen in the *Navigator* window (see Fig. 3.4) for modeling. Mark a process group and click on the small triangle to the left of the group to expand or collapse it. Double-click on a process to start working with it.

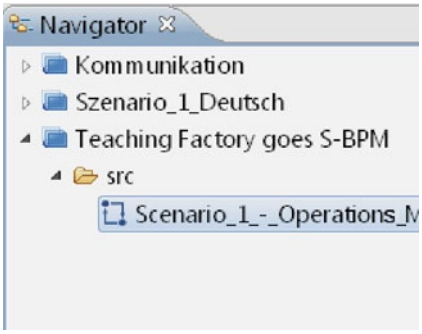


Fig. 3.4 Viewing the new process

www.dbooks.org

3.2.3 Modeling the Process – Subject Communication

Now that a process group and a process exist, we can start to model the process.

Subjects

Exactly speaking, each subject is a process on its own. They synchronize action through communication.

The first question you should answer is the number of subjects in the process. In this case there are two subjects: the operations manager and the logistics department. So the first step is to create these two subjects in the process. This is simply done by dragging and dropping the object named *Internal Subject* from the *Palette* (on the right-hand side by default) onto the opened process modeling space, as shown in Fig. 3.5. In the dialog window *Create a new subject* enter a reasonable name for the created subject. Leave the check box *Multiple subject* unchecked.

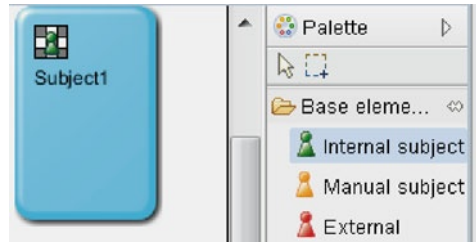


Fig. 3.5 Creating a new subject from the palette

The subject's name is either "Operations Manager" or "Logistics". After having added the first subject, repeat this step for the second subject. After adding both, the *Operations Manager* and the *Logistics* subject, the process should look like that shown in Fig. 3.6. Feel free to move the subjects around on the working space – or canvas, if you prefer to think of it from a design point of view.



Fig. 3.6 The newly created subjects: *Operations Manager* and *Logistics*

Communication

After having defined all necessary subjects, the next step is to identify the communication between them.

Remember, we want to model the communication behavior as discussed in the first chapter. If you have not already read that chapter you may wish to do so now.

The next step includes some brain work: the goal is to identify the communication behavior between the two subjects. Analyzing the problem, the question should be: what kind of messages do the subjects exchange and what are the contents?

In the example, there are four possibilities to structure the communication:

Starting point: The operations manager sends a message to the logistics department to inform them about an irregular process state: an order cannot be completed because a specified problem occurred which forces the production date of that item to be postponed. The operations manager also specifies the production order number, a response

time by which a response from logistics is needed and an alternative, which is executed when there is no timely solution.

Possibility 1: The logistics department receives, reads, and accepts the message. As a matter of acknowledgment a confirmation is sent back, with an optional comment.

Possibility 2: The logistics department does not respond in time, but an alternative exists – the alternative is then executed.

Possibility 3: The logistics department does not respond in time and no alternative exists. In that case the operations manager tries to reach an agreement with the logistics department to solve the problem. This can be done via different communication channels, for example calling the logistics department, a personal chat with a responsible member of the logistics department, escalating the matter to a superior etc. After both sides have agreed on a solution, the logistics department sends a message back to the operations manager in which the agreed changes are specified (and documented).

Possibility 4: The logistics department receives, reads but does not accept the message. An informal solution is then agreed on via an arbitrary communication channel, as specified above. After both sides have agreed on a solution, the logistics department sends a message back to the operations manager in which the agreed changes are specified.

When looking at the four different possibilities it is clear that there are three different types of messages which are exchanged between the subjects:

- A "Request for Change" (RFC), sent by the operations manager to the logistics department which includes the following fields:
  - Production order number
  - Old starting date and time
  - New starting date and time
  - Explanation of reason for change
  - Answer needed by (date and time)
  - Alternative, if not accepted
- An "Approval", sent by the logistics department to the operations manager if the RFC is accepted including the additional field:
  - Comment
- An "Agreed upon Change" message, which is sent after both sides have agreed on a solution with the following fields:
  - Production order number
  - New starting date and time
  - Comment

Some of these communication issues could be handled by using an ERP-system, but typically not all of them.

After you have identified all necessary information for further modeling, the next step is to implement the communication flow in the *Metasonic Suite*.

The first action is to model the communication flow from the operations manager to the logistics department subject. To do so, you left-click on the previously created subject *Operations Manager* and click on the envelope symbol labeled *Create new message*, as shown in Fig. 3.7.

www.dbooks.org

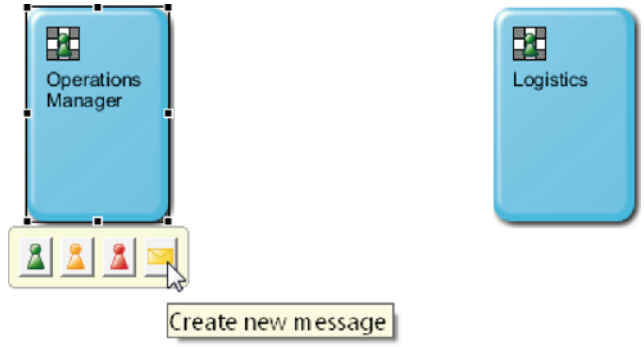


Fig. 3.7 Subjects with context menu, which has the same options as in the *Palette* window on the right-hand side of the application window

Now move the mouse pointer to the target subject *Logistics* (an arrow representing the message flow appears) and left-click on the target subject. A dialog window labeled *Create new message* appears. In that window click on the link *Create a new message type*, as highlighted with a red box in Fig. 3.8.

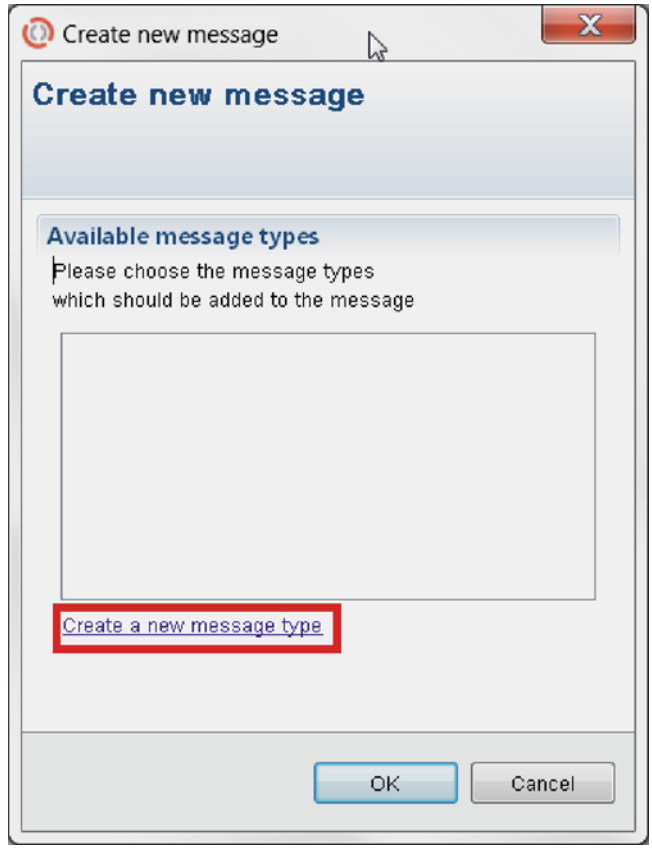


Fig. 3.8 *Create new message* window. Select the *Create new message type* link, which is highlighted with a red square

In the *Name* field enter the text "Request for Change", which is the first message to be created (see Fig. 3.9).

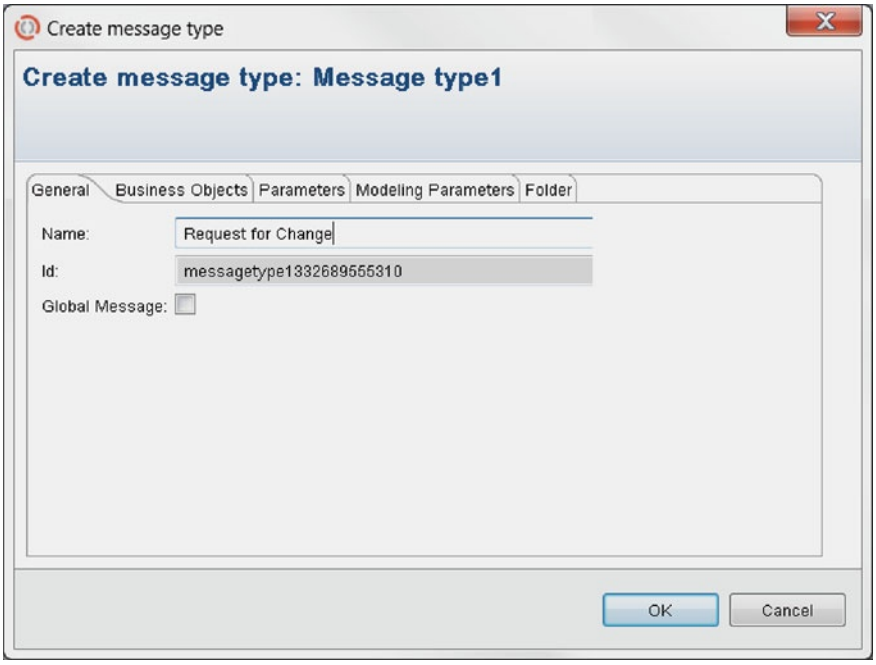


Fig. 3.9 *Create message type* window. The field *Id* is automatically created by the application and will show a different text in your case

After you have named the message, the parameters of that message have to be specified. This is done in the tab *Parameters* by simply clicking *New* and adding the name of the parameter to be created. The parameters correspond to the different message types mentioned in the text above. Figure 3.10 shows an example of the parameter *Production Order Number*.

The parameters of any defined message can also be modified from the main window via the *Palette*. If you accidentally closed the dialog window *Create message type*, then do it that way.

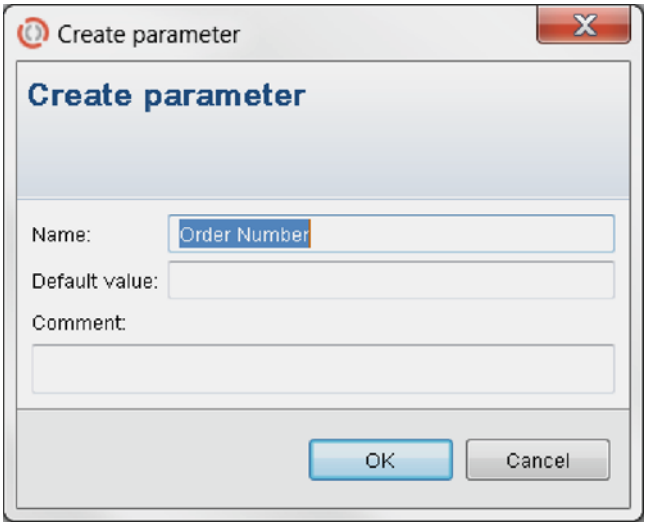


Fig. 3.10 *Create parameter* window. In addition to the name, you could also specify a default value, but in our example this is not necessary

The parameters are added one by one by entering the name in the dialogue *Create parameter* and clicking the *OK* button afterwards.

www.dbooks.org

After you have added all necessary parameters, the window should look like that shown in Fig. 3.11.

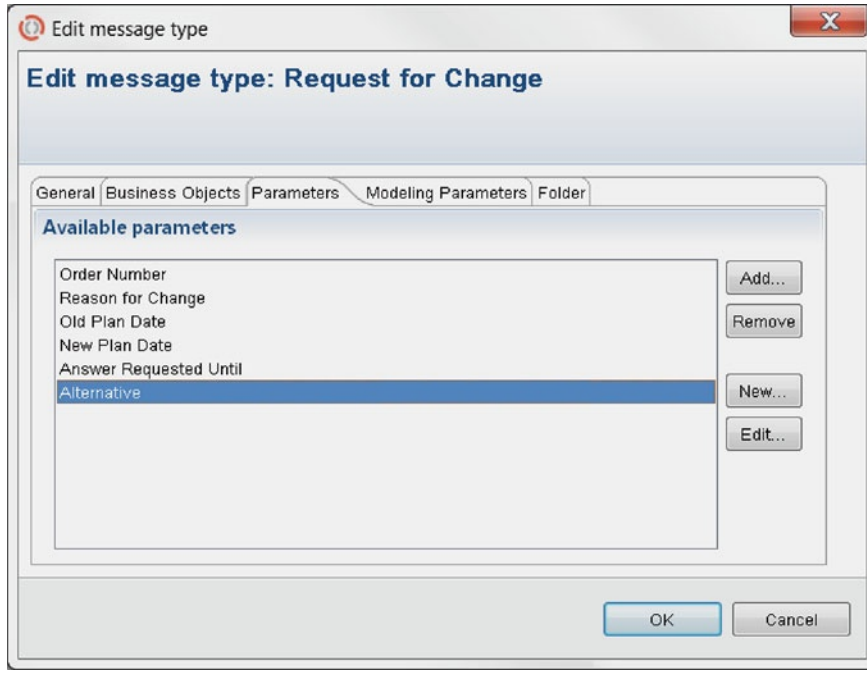


Fig. 3.11 Dialog window *Edit message type – Request for Change*. Here you can see an overview of all parameters that are assigned to this message type

After you hit the *OK* button, the message is created and again the *Message Overview* window, which now shows all available messages, is shown, with *Request for Change (local)* being the sole available message. Now we can use all the created message parameters during the modeling of the process.

After you click the *OK* button again, the message created beforehand is linked from the *Operations Manager* subject to the *Logistics* subject and the whole process model should look as shown in Fig. 3.12.

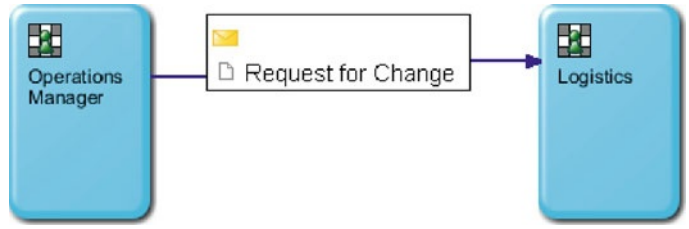


Fig. 3.12 Process overview 1: The two subjects, *Operations Manager* and *Logistics*, connected by the message *Request for Change*

Then the other two messages, which are directed from the *Logistics* subject to the *Operations Manager* subject, are created. Use the same procedure as before: click the *Logistics* subject and use the envelope icon to connect the message arrow with the *Operations Manager* subject. In the opened dialog window now create a new message type by clicking on the link *Create a new Message Type*.

Name this message "Approval" with the only parameter "Comment". After confirming to close the dialogue window, the window *Create new message* now displays

two messages – the previously created "Request for Change" and the newly created "Approval".

If one or both messages are not shown, create the missing one.

Persevere and add the third message by clicking the link *Create a new message type* again. This message is named "Agreed upon Change". Here all needed parameters already exist because you already created them previously. So you can add them by clicking the *Add* button and checking the needed parameters: "New Plan Date", "Comment", and "Order Number". After confirming the selection, the window *Create message type* should look as shown in Fig. 3.13.

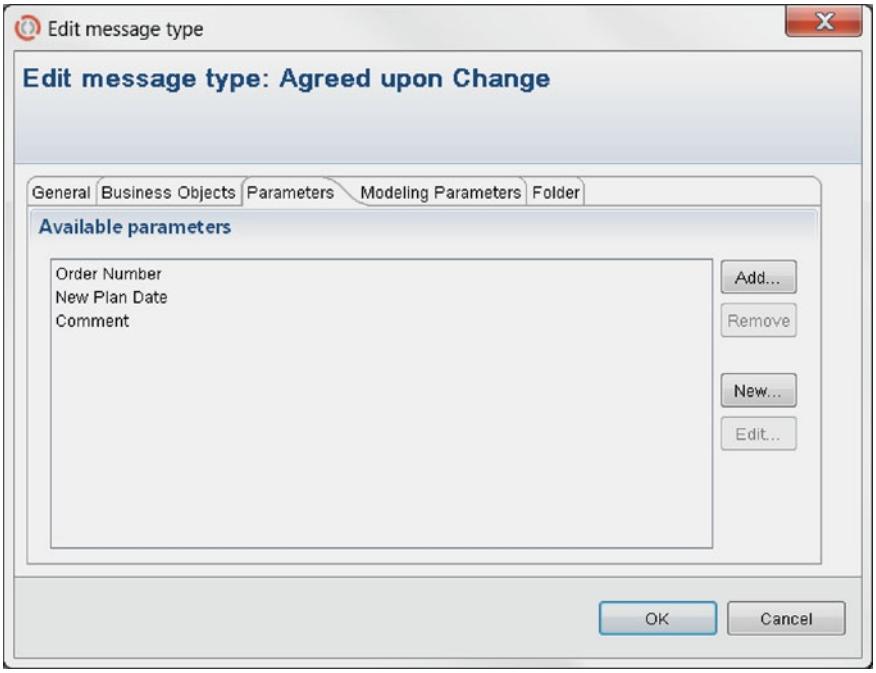


Fig. 3.13 Dialog window *Edit message type – Agreed upon Change* with the assigned parameters

Now, after you have confirmed closing the current and the following window, create the communication link between the logistics department and the operations manager subject as you did before. The two subjects should now look as displayed in Fig. 3.14.



Fig. 3.14 Process overview 2: In addition to the first overview, also the *Approval* and *Agreed upon Change* messages are implemented

The overall communication between the two subjects is now defined.

www.dbooks.org

3.2.4 Modeling the Process – Internal Behavior

Now we have to define the internal behavior of the two subjects; this means that we now want to define how the subjects will deal with the previously defined messages. To be exact, each subject represents a process, which is defined as the internal behavior of the subject; these processes are synchronized, i. e., coordinate their execution, via the exchange of messages.

Behavior of the Operations Manager

First, we have to define which of our subjects in the model initiates the process. In our example this is the operations manager, which means that the first step is to model the internal behavior of that subject. The modeling of the internal behavior is initiated by double-clicking the related subject, which in this case is the *Operations Manager*.

This opens another canvas labeled with the name of the subject, in our case *Operations Manager*. The communication structure was discussed above; this structure now has to be modeled in the S-BPM notation. The first element needed is the start element – this element is unique within a process and starts the whole process choreography.

To start modeling, select a *Function state* element from the *Palette* on the right-hand side of the application window and drag it onto the canvas *Operations Manager*, as shown in Fig. 3.15.

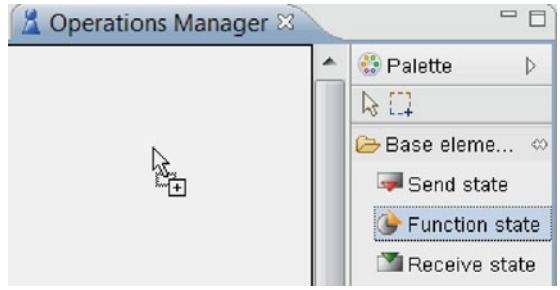


Fig. 3.15 A new function state is created by dragging and dropping from the *Palette*

For a better view, it is suggested that you place it on the top center of the actual canvas. You have to assign a name to the element to indicate its function. In this example, label the new state "Problem description" (see Fig. 3.16). The chosen name and other configuration possibilities are shown in the *Properties* tab below the canvas. There you can also see the text *Is start state*. The first created function state will be assumed to be a start state.

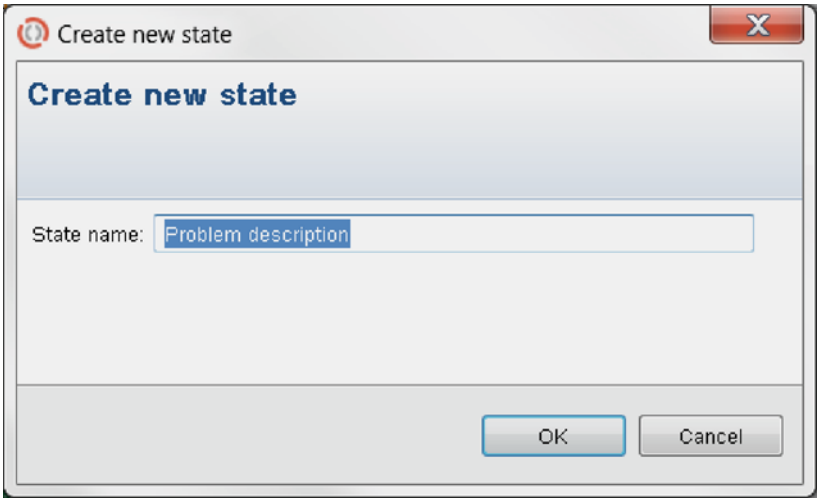


Fig. 3.16 *Create new state* window – here the name for the new state is chosen

The "play" icon in the top right corner of the created *Problem description* function state visually indicates that this is the starting state of the subject (see Fig. 3.17). This means that the first action the subject should do, when the process starts, is to describe the problem which occurred; in our case this is a problem in the execution of the planned production schedule.

Any process starts because of an event; in this case it is the need to reschedule production.

After you have successfully created the first function state, it is necessary to add the previously created parameters. To do so, select the function state *Problem description* and then, at the bottom of the screen, select the menu entry *Parameters* in the tab *Properties* (see Fig. 3.18). Practically, we make the messages and parameters available to the model. The idea behind this is the so-called paradigm of information hiding. We should always carefully select, who should be able to see what information. So here we can decide whether a specific piece of information is only visible (read only) to a subject, or if it can be edited by that subject (read and write).

Information hiding paradigm

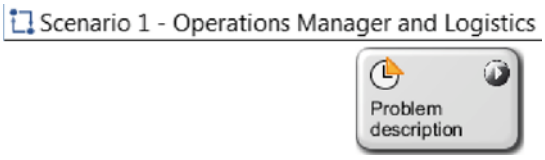


Fig. 3.17 Internal behavior – 1: Here you can see the starting point of the process, the *Problem description* function state

Since the *Operations Manager* is the one who sends a notification about a problem, the parameters need to be editable. All existing parameters are added as editable by clicking the *Add…* button.

The *Select all* option is helpful to select all the parameters at once.

After you have added all parameters, the window should look as shown in Fig. 3.18. The parameters are inherited throughout the process model, which means that subsequently created elements also have access to the specified parameters.

www.dbooks.org



Fig. 3.18 The *Parameters* tab of the function state *Problem description* which shows the editable parameters

After formulating the problem, the operations manager sends the *Request for Change* message to the logistics department. To model this, choose the element *Send state* from the *Palette*.

The simplest way to do so, is to select the *Problem description* state and then use the leftmost symbol in the context menu, to directly create a new send state. Move the mouse pointer to the position at which you want to place the new send state and release with a mouse click – the new state is created and the *Create transition* window pops up; enter "Send Change Request" in the text field *Send state name* to name it. The action that occurs between the two states is already defined as "Problem description done" by the application, which in this case exactly describes what should happen (Fig. 3.19). After that, the internal behavior should look similar to that shown in Fig. 3.20.

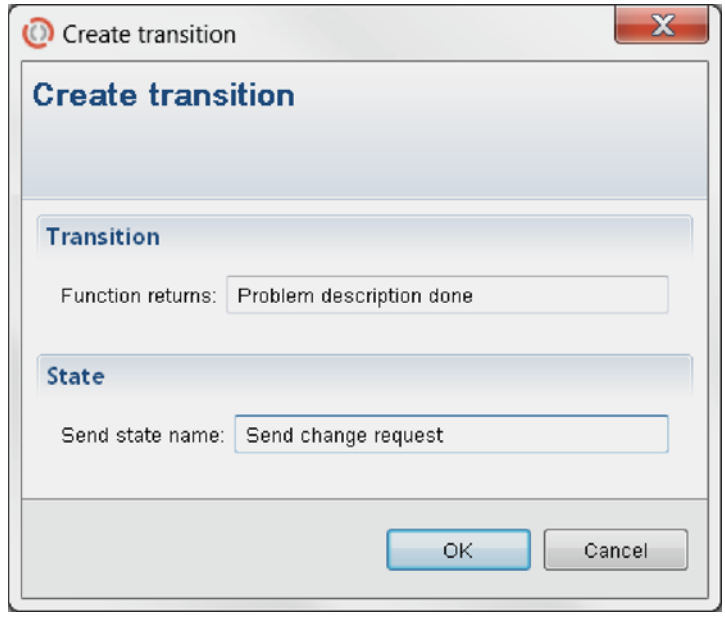


Fig. 3.19 *Create transition* window – the program always automatically suggests a text for the *Send state name* text box

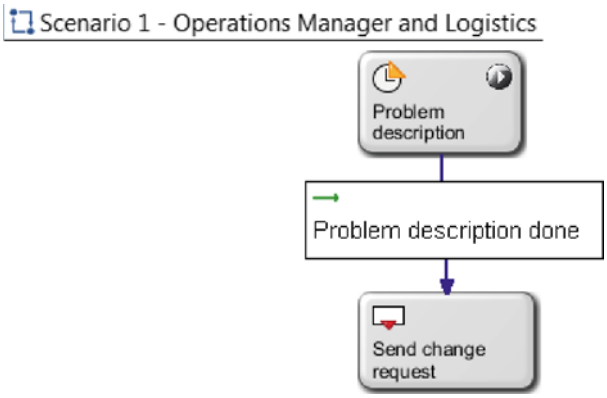


Fig. 3.20 Internal behavior – 2: The *Send change request* state is added to the starting state

After having sent the change request, the operations manager is waiting for an answer. To model this, a so-called receive state is used. This state also specifies from whom we expect which message. The approach is similar to the one described before, except that the icon used in the *Palette* is the second one.

Now, create a new *Receive state* below the send state. A dialog window, labeled *Create transition*, pops up; there we have to define the *Transition*, i. e., the *Receiving Subject* and the *Messagetype*. In this case the *Receiving Subject* is *Logistics* and the *Messagetype* is *Request for Change (local)* – use the drop-down menus for proper selection. Before closing the dialog window, name the created *Receive state* "Waiting for answer" and press *OK*. After that, the internal behavior should look as displayed in Fig. 3.21.

You can rearrange all symbols with your mouse; or, you can use the menu command *View/Align* for automatic alignment.

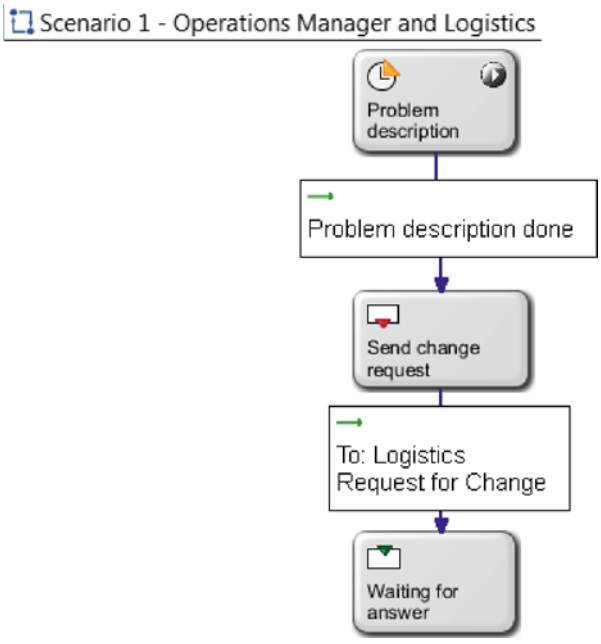


Fig. 3.21 Internal behavior – 3: After sending the change request, the subject is now waiting for an answer

From the point of view of the operations manager, there are now three possible cases: either the *Approval*, the *Agreed upon a Change*, or no message at all (a timeout) is received. We suggest to first model the "positive" or "happy" path, where an *Approval* is

www.dbooks.org

received, which is also the desired output of the process from the operations manager's point of view.

To do so, create another function state below the *Waiting for answer* state with the approach already used twice. The function state is the second symbol from the left in the small pop-up menu. Creating this state will make the *Create transition* window appear. Here you have to specify which message is to be expected and from whom. In this example the sender is the logistics department and the received message is the *Approval*, as shown in Fig. 3.22. Name the new function state "End", because after having received the approval, there is nothing more to do for the operations manager and the process ends.

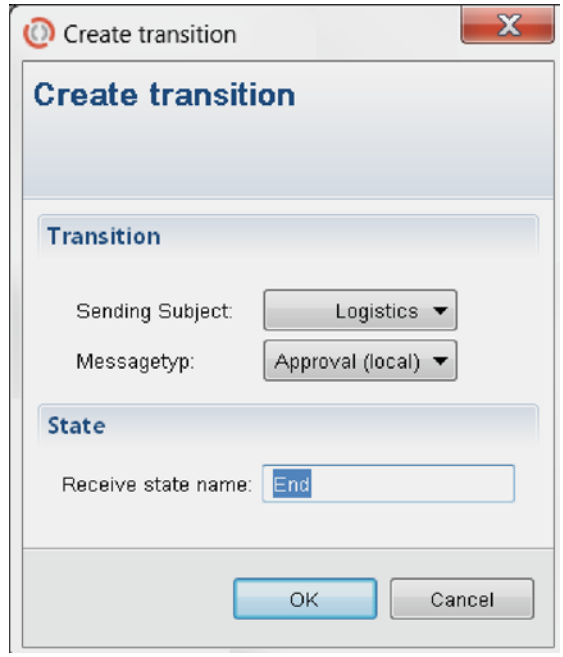


Fig. 3.22 *Create transition* window, which creates the transition between a receive state and a function state called "End"

After the function state *End* has been created, it has to be marked as an *end state* so the process can terminate after the state is reached. To do so, click the state and check the *Is end state* check box in the *Properties* window at the bottom of the screen. If you can't see the check box, make sure that the *General* tab is selected. The end state is visualized by a stop icon in the top right corner of the function state. The whole modeled internal behavior should now look as in Fig. 3.23.

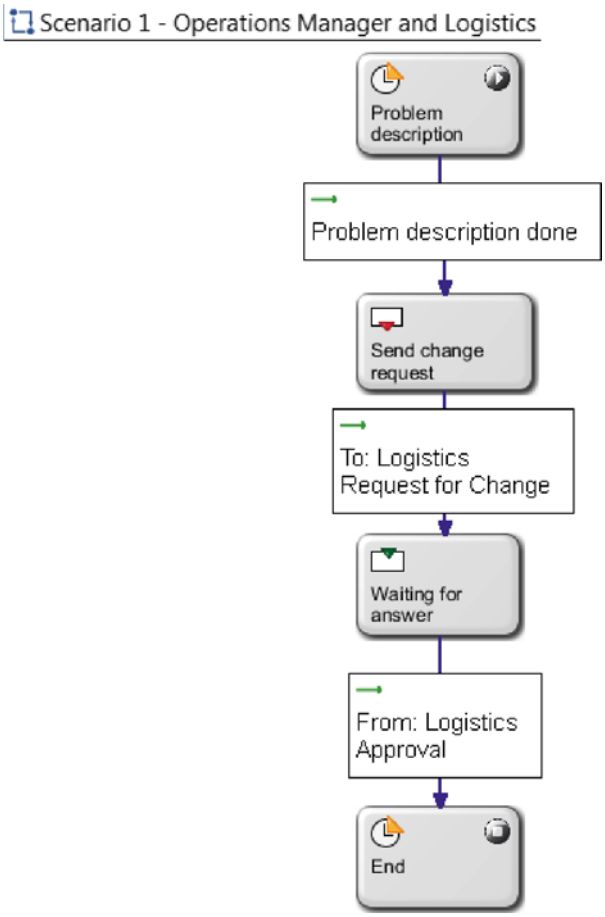


Fig. 3.23 Internal behavior – 4: The subject now also has an *End* state (recognized by the square symbol in the top right corner of the function state)

The next possibility which has to be modeled is the one where the request for change is not acceptable for the *Logistics* subject, but where the two subjects (representing the involved actors) agree on an informal solution. The informal solution is then sent from the logistics department to the operations manager as a documented confirmation. This situation represents scenarios where communication occurs beyond the process management system; this could be via personal negotiations on the phone, by email, or face-to-face. But, at the end we want the process instance to reflect the real path and to terminate with a predefined state.

An instance is the notion of an executed process, a concrete case.

Of course we could modify the process in any thinkable direction; for example the operations manager sends a confirmation to reflect a mutual agreement. This means that in this case the operations manager receives an *Agreed upon a change* message. After that message, the operations manager only needs to check the agreed message parameters (representing, e. g., the new time schedule) and after that the process is finished.

www.dbooks.org

So, this is another possibility of what can happen after the *Waiting for answer* state. Another function state, *Realize approved changes*, has to be created and connected to the function state *Waiting for answer* using a transition labeled *Agreed upon a change*.

Now, select the *Waiting for answer* state and create a new function state (use the small context menu) at the right-hand side of the *Waiting for answer* state. Again, in the dialog window *Create transition*, select from the drop-down menu *Sending Subject* the entry *Logistics* and from the drop-down menu *Messagetype* the entry *Agreed upon Change (local)*. Name the newly created state "Realize approved changes" (Fig. 3.24).

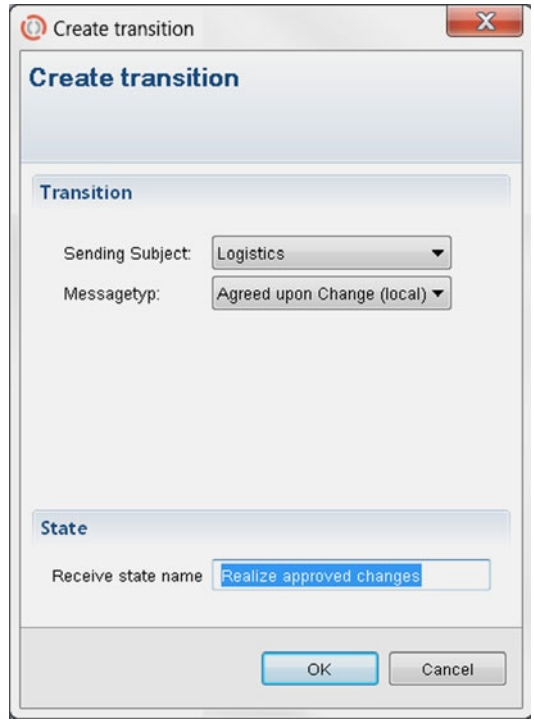


Fig. 3.24 *Create transition* window between a send state and a function state

After that, the internal behavior should look as shown in Fig. 3.25.

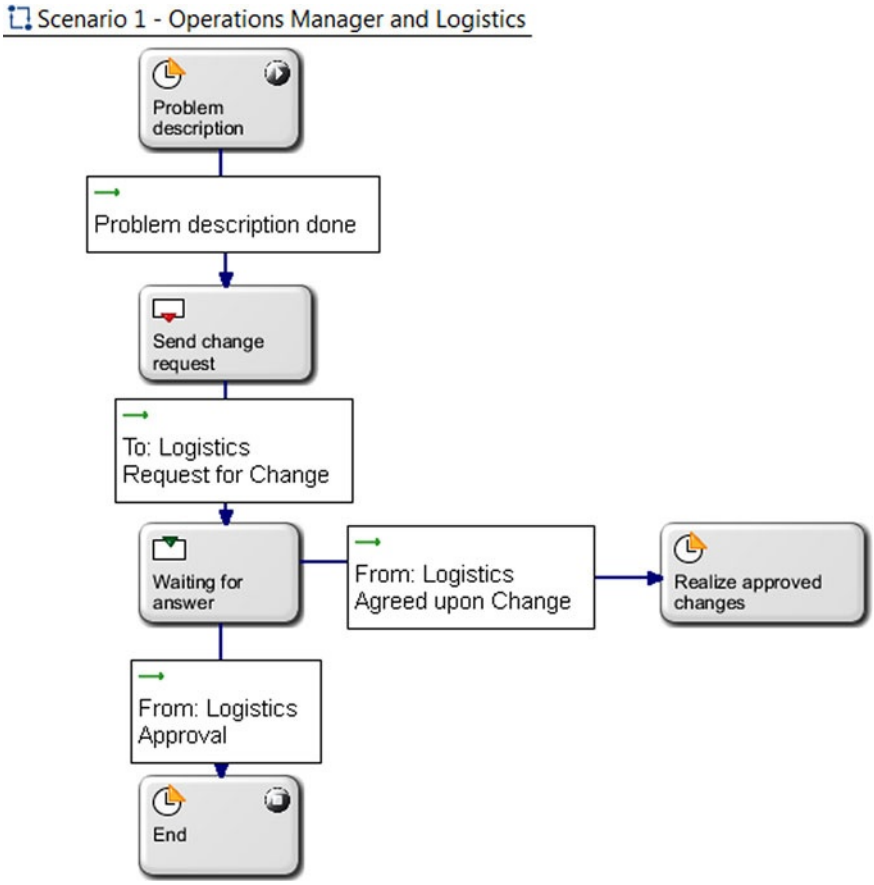


Fig. 3.25 Internal behavior – 5: A fork is added to capture the case where the *Logistics* subject does not approve

You can change the appearance of connectors with your mouse; click on the connector to see the connection points with states (red) and the "moving" points (black). So move it around as you like. The state transition description box can also be moved to a different position; if you move it away from the transition, a thin line will still show the connection to the corresponding transition.

As the activity *Realize approved changes* is the last one in this process path, the process should then end. So we have to connect this state with the previously created *End* state. Select the *Realize approved changes* state and then choose the first icon in the context menu on the right, which is a connector, to connect it to the *End* state. The default name for the transition ("Realize approved changes done") can either be left as is or changed to something like "Realized approved changes". This should make the internal behavior look as shown in Fig. 3.26.

www.dbooks.org

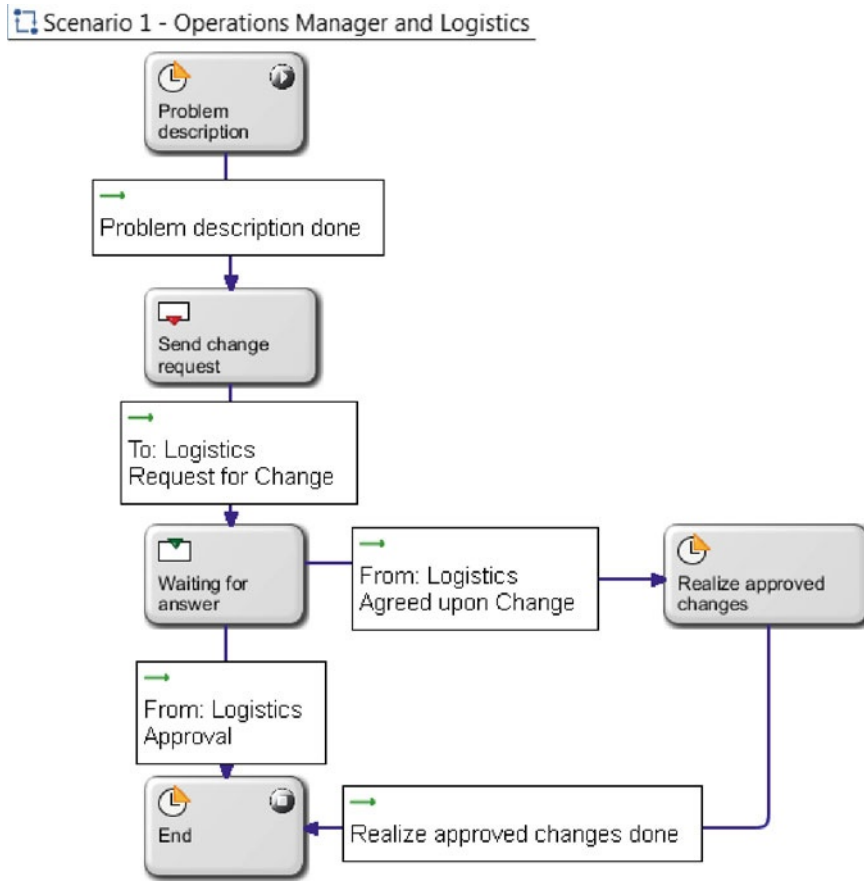


Fig. 3.26 Internal behavior – 6: The new branch of the fork is now connected to the *End* state. Every process must have an end state

The next step to model is the possibility that there will be no answer on time. This case will be modeled in a slightly different way, because *function states* created via the small pop-up menu when selecting a state do not cover the use of timeouts.

The first thing to do, when there is no answer on time, is to check whether an alternative path exists or not. So drag a new *Function state* from the *Palette* on the right to the left-hand side of the *Waiting for answer* state. Label this state "Alternative exists?" (Fig. 3.27).

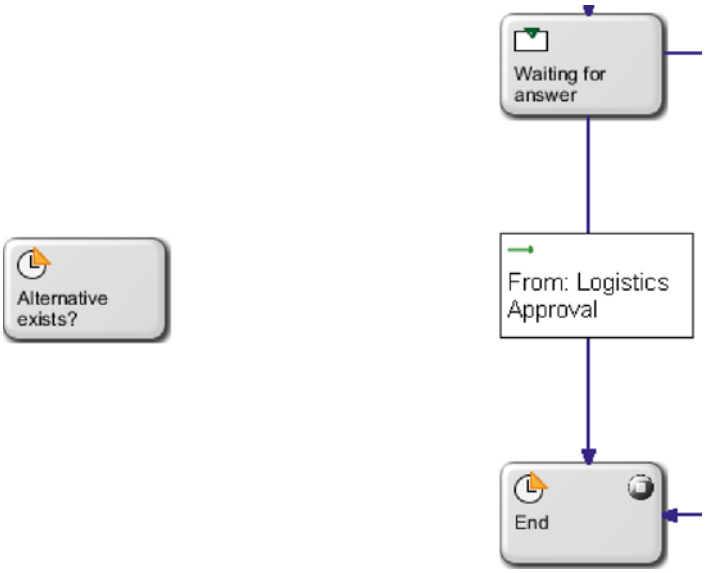


Fig. 3.27 Internal behavior – 7: A third branch is added, to capture the case where there is no answer on time

Contrary to the previous approach, the transition between these two states has to be created manually. To do so, select the *Manual timeout* from the *Palette* with a left click. Click on the *Waiting for answer* state and then on the *Alternative exists?* state (in exactly this order) to create a transition from the former to the latter state. Label the transition "No answer on time" (Fig. 3.28).

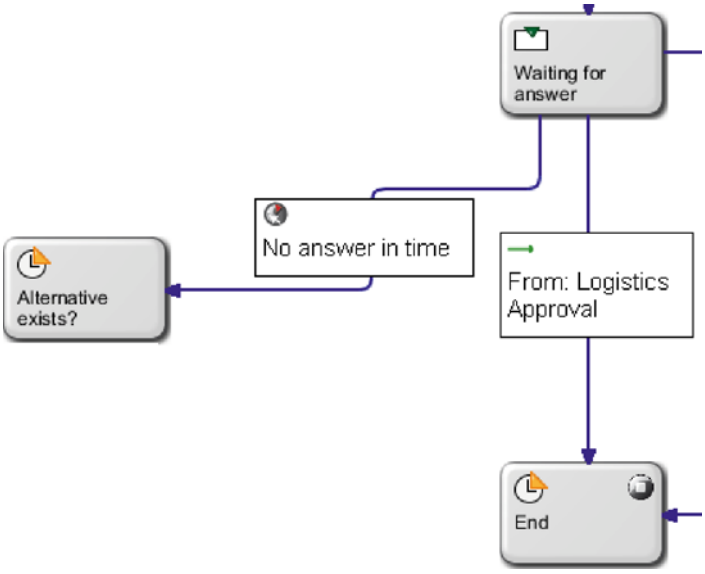


Fig. 3.28 Internal behavior – 8: A manual timeout is used for the third branch

Now, again, there are two possibilities. Either there exists an alternative, in which case the alternative is executed and the process ends, or there is no alternative, which means that the two *subjects* have to agree on an informal solution.

www.dbooks.org

To model the case of an existing alternative, create a new *Function state* labeled "Realize alternative" below the *Alternative exists?* state, again with the help of the small pop-up menu. Label the transition "Alternative exists" instead of "Alternative exists? done". After you have created the new function state, the only step left is to connect it to the *End* state in exactly the same way as described before when connecting the *Realize approved changes* to the *End* state. Again, the transition can be relabeled; otherwise, the default label can be used.

A view of the bottom of the resulting internal behavior model is shown in Fig. 3.29.

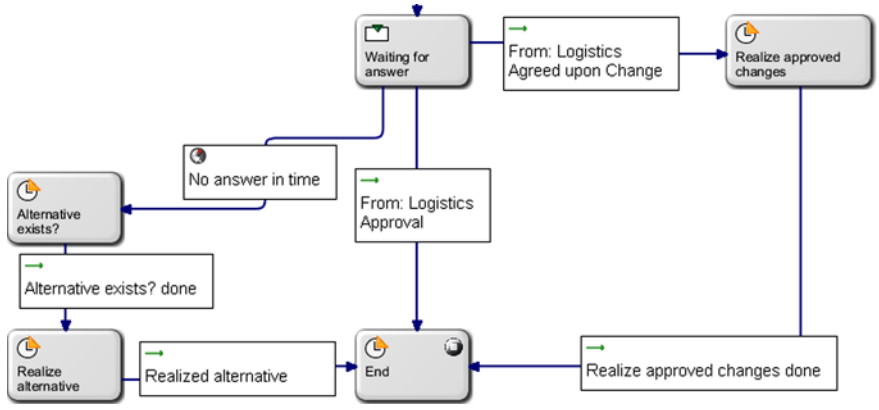


Fig. 3.29 Internal behavior – 9: All branches of the fork need to lead to an *End* state

The last step in modeling the internal behavior of the operations manager is to model the case in which no alternative exists and there is no answer on time. Following this case, there has to be an informal solution that the two subjects agree upon – using any possible communication channel.

So, create a function state labeled "Informal solution". Create this function state above the *Alternative exists?* state and name the transition "No alternative exists".

If there is an informal solution, the logistics department sends an *Agreed upon Change* message, which means that after agreeing on an informal solution, the operations manager has to go back into the *Waiting for answer* state.

To make this happen, connect the newly created *Informal solution* state to the *Waiting for answer* state with the transition *Agreed on informal solution* – and the modeling of the internal behavior of that subject is now finished. Now all possibilities have been taken into account and modeled as internal behavior.

Figure 3.30 shows the newly created parts of the internal behavior. Figure 3.31 provides an overview of the complete internal behavior of the operations manager.

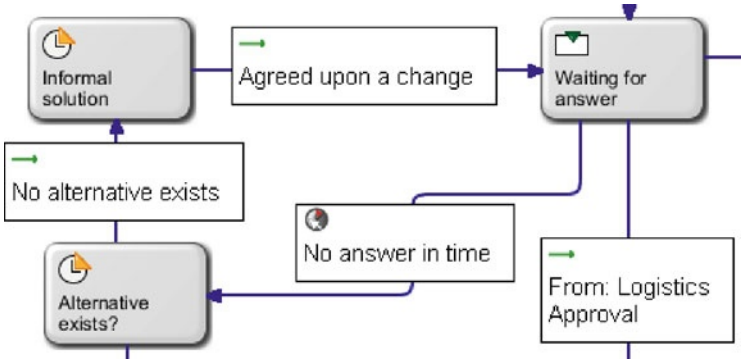


Fig. 3.30 Internal behavior – 10: The last possibility (there is no answer on time and no alternative exists) is added

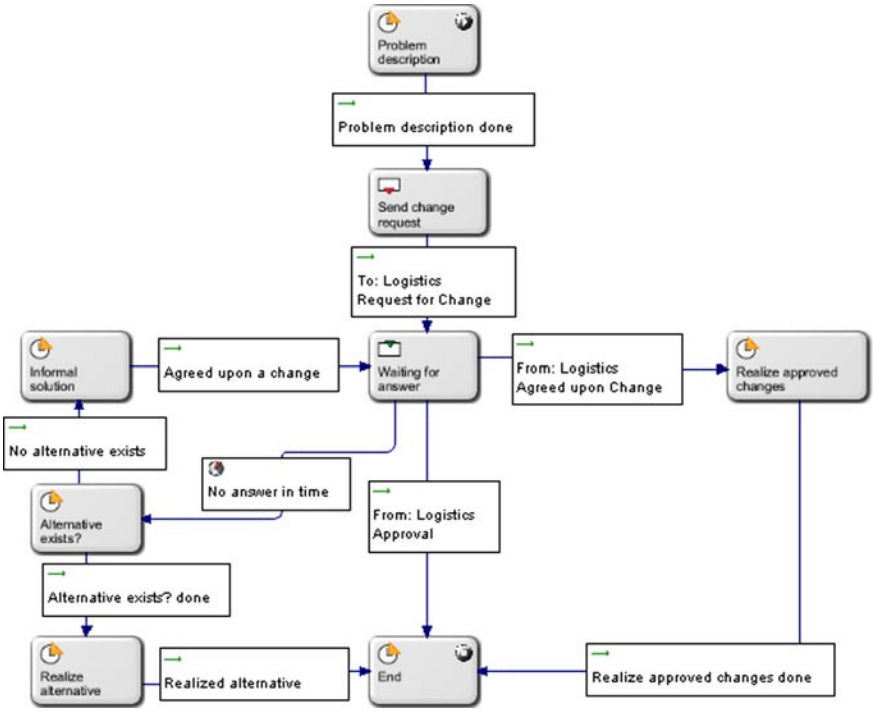


Fig. 3.31 Internal behavior – overview. The internal behavior of the *Operations Manager* subject is now finished

Behavior of the Logistics Department

We will now model the internal behavior of the *Logistics* subject.

The possible behavior from the logistics' point of view is quite simple: whenever a change request is received, it is either accepted, which results in sending back an *Approval* message, it is not accepted, which leads to the *Agreed upon change* message – or it is simply ignored.

In all cases the first step is to receive the *Request for change* message. To start modeling, double-click the *Logistics* subject in the overview window. Then drag a *Receive state* element from the *Palette* onto the canvas (Fig. 3.32) and label it "Receive change request".

www.dbooks.org

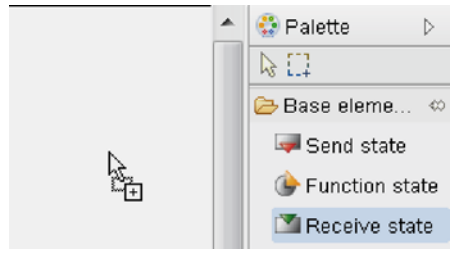


Fig. 3.32 Creating a new receive state from the *Palette*

Placed in the top center of the canvas, it now marks the starting point of the internal behavior of the *Logistics* subject, which is again visualized by a small "play" icon within the element, as shown in Fig. 3.33.



Fig. 3.33 Internal behavior – 1: The start state of the *Logistics* subject

The next step in the behavior, after having received the change request, is to evaluate whether it is acceptable or not. This means, that the next state will be a function state, which also creates the transition between the two elements. To create it, select the already created *Receive change request* state and then click *Create new function state* from the pop-up menu as described previously.

Create this state beneath and label it "Change request acceptable?". The transition consists of *Operations Manager* as the *Sending Subject* and *Request for Change (local)* as the *Messagetype* (Fig. 3.34).



Fig. 3.34 *Create transition* window – once again

Afterwards, the behavior of the logistics subject should look similar to the one shown in Fig. 3.35.

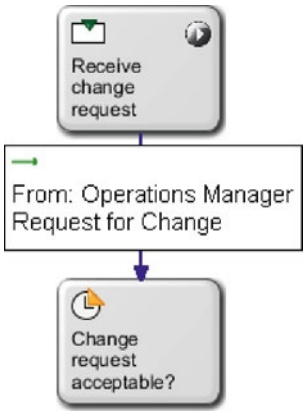


Fig. 3.35 Internal behavior – 2: After a message is received, the subject must check whether the request is acceptable or not

After that, similar to the operations manager's internal behavior, the parameters have to be specified so the received message can be read correctly. To do this, select the *Change request acceptable?* state and choose the parameters tab. This time the parameters are not editable but for display only. Therefore, on the right-hand side, in the *Parameters for display only* section, add all parameters as display parameters by using the *Add…* button followed by the *Select all* button (Fig. 3.36).

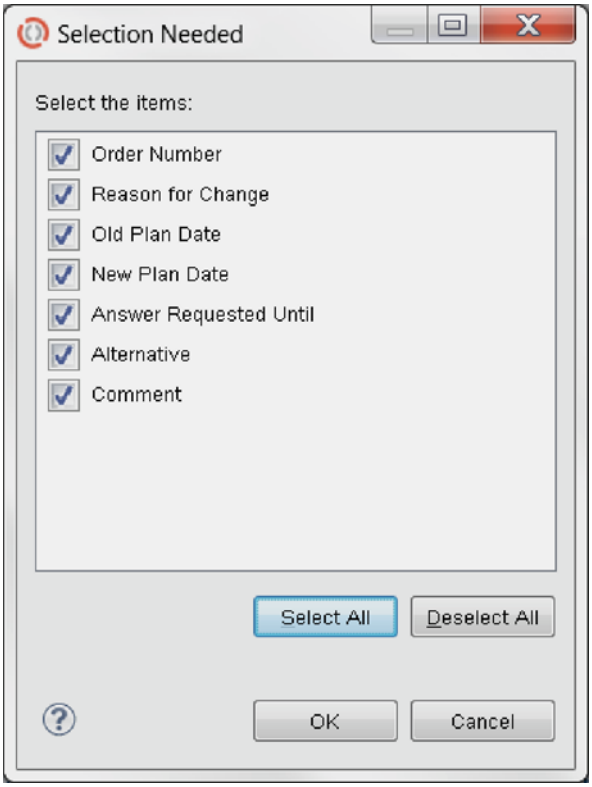


Fig. 3.36 *Selection needed* window – use the *Select all* button to check all check boxes

www.dbooks.org

If the *Parameters for display only* section looks as shown in Fig. 3.37, the step was performed correctly.

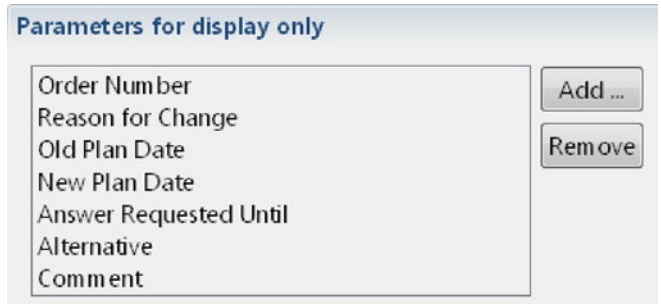


Fig. 3.37 *Parameter* window – the previously selected parameters are now displayed here

Again, the simplest path is modeled first. The "happy" result of this process is, that the logistics department immediately accepts the operations manager's request for change.

After coming to the conclusion that the *Request for change* is acceptable, the logistics department sends back the *Approval* message, and after that, the process ends.

So the next step is to model the sending of the *Approval* message by creating a send state, using the already known pop-up menu approach, below the *Change request acceptable* state, which also creates the transition between the two elements. The send state is the first element on the left in the pop-up menu.

Name the transition *Change request acceptable* and the newly created send state "Agree on change request". The internal behavior of the *Logistics* subject will then look like that shown in Fig. 3.38.

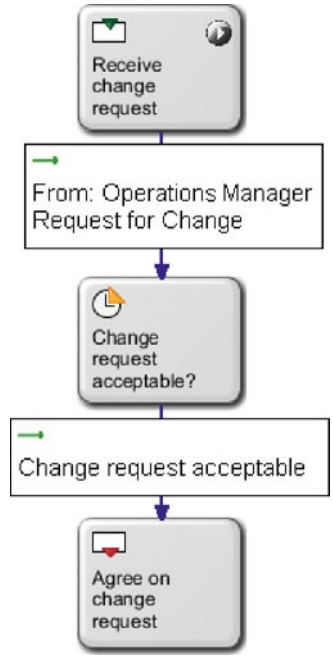


Fig. 3.38 Internal behavior – 3: A send state is added to the internal behavior

If you want the logistics department to be able to add a comment to the *Approval* message, you will have to add the *Comment* parameter to the editable parameters section of the *Change request acceptable?* state. To do so, simply right-click the *Change request acceptable?* state and select *Properties* from the context menu. Afterwards click the *Parameters* tab. Here, click the *Comment* parameter in the *Parameters for display only* section and click the *Remove* button. After that, click the *Add…* button in the *Editable parameters* section. In the following window, select the only available parameter (which should be *Comment*) and add it by checking the check box and clicking the *OK* button.

The only thing left to do for the logistics subject, after having agreed on the change request, is to end the process, which leads to the transition which actually sends the message.

To finalize the so-called happy path (sometimes also called sunshine path), simply add a function state labeled "End" beneath the *Agree on change request* state using the pop-up menu. For the transition, the *Receiving Subject* is of course *Operations Manager* and the *Messagetype* is *Approval (local)*. To formally specify that the process terminates at the *End* state, check the *Is end state* check box in the *General* tab when selecting the element. Figure 3.39 shows an overview of the internal behavior of the logistics subject modeled so far.

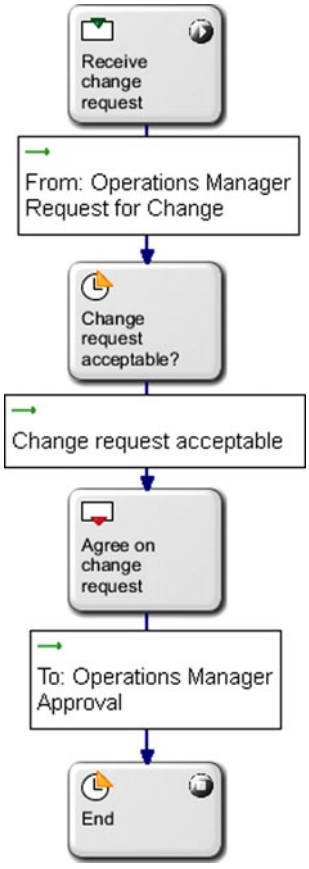


Fig. 3.39 Internal behavior – 4: This subject also needs an *End* state

www.dbooks.org

Informal communication and natural language are the essence of any human collaboration; but tracking the result, we can even analyze how often and how long such actions were needed.

The next step is to model the possibility that the change request is not acceptable. If that is the case, the logistics subject is required to simply find a solution by using a freely chosen communication channel. The outcome should be that a solution is negotiated with the operations manager. After a solution has been agreed upon – via informal communication and natural language –, the agreed solution is sent from the logistics subject to the operations manager subject as a confirmation. After that the process ends.

Create a function state called "Find solution" on the right-hand side of the *Change request acceptable?* state via the pop-up menu. Label the transition "Change request not acceptable", which makes the internal behavior look similar to that shown in Fig. 3.40.

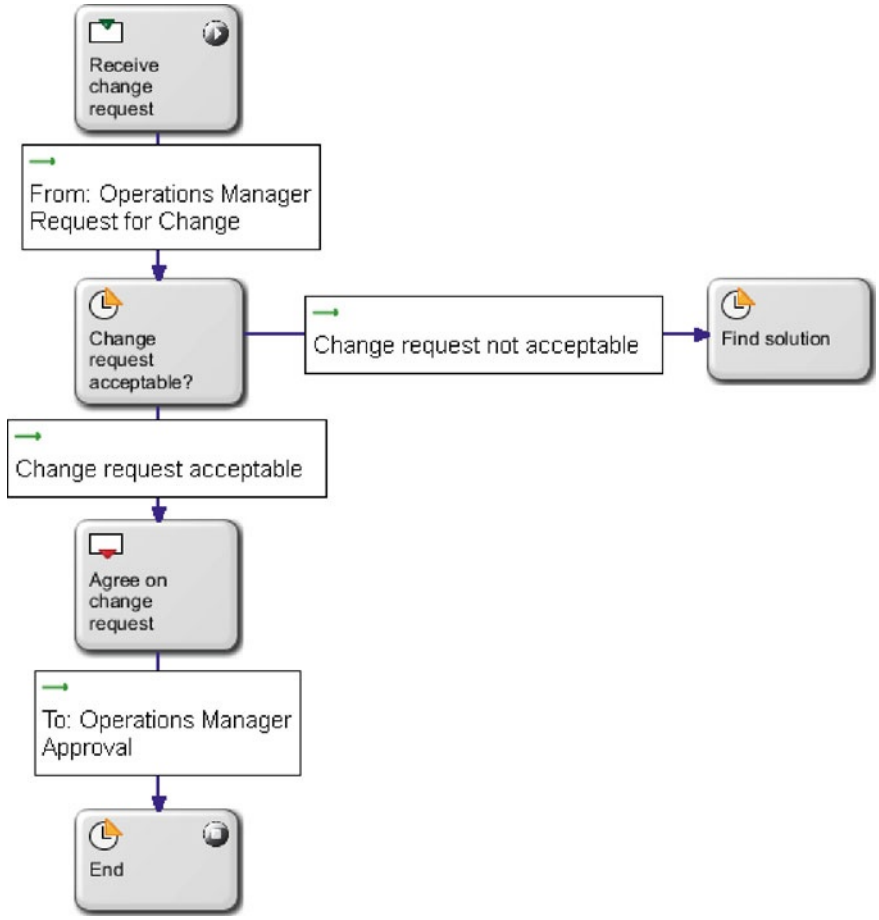


Fig. 3.40 Internal behavior – 5: The first fork is added, to capture the case where a change request is not acceptable

The next step is to send an *Agreed upon change* message to the operations manager. There are only two editable parameters which the logistics subject may have to change: the *New Plan Date* parameter and the *Comment* parameter.

After selecting the *Parameters* tab of the *Find solution* state, add these two parameters to the *Editable parameters* section by using the *Add…* button. Afterwards add all remaining parameters to the *Parameters for display only* section by again using the *Add…* button and the *Select All* button. This should lead the *Parameters* tab of the *Find solution* state to look similar to that shown in Fig. 3.41.



Fig. 3.41 *Parameter* window – there should be two editable and five noneditable parameters

As said before, the next step after a solution has been found is to send the agreed solution to the operations manager as a confirmation. Therefore, create a *Send* state below the *Find solution* state using the pop-up menu. Label the transition "Found solution" and the new send state *Send agreement*. Afterwards the *Change request not acceptable* branch should look similar to the one shown in Fig. 3.42.

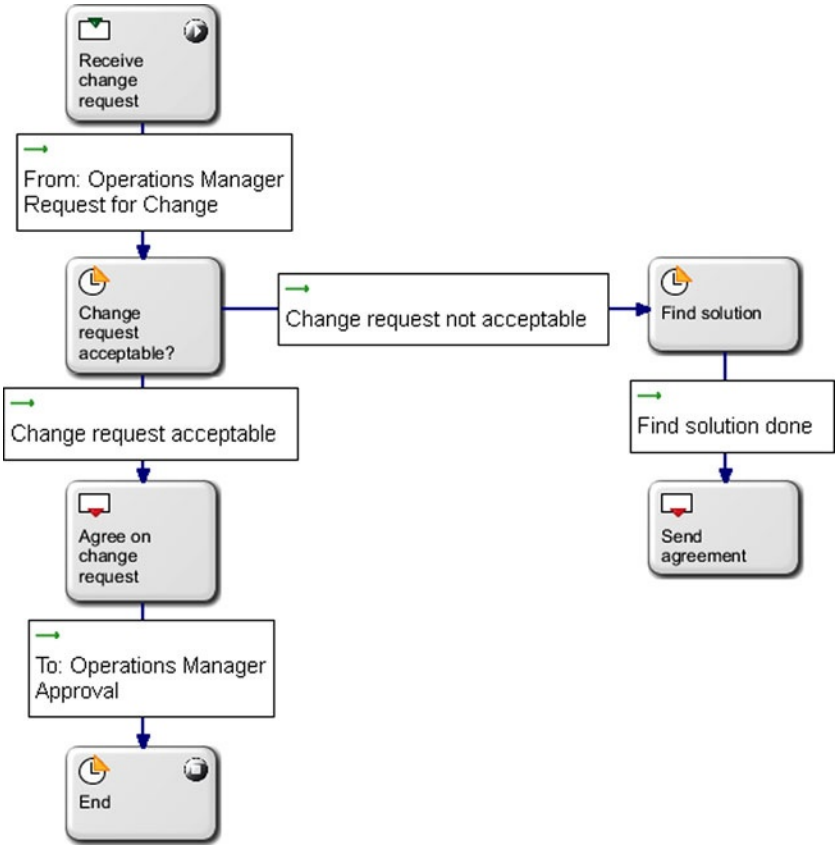


Fig. 3.42 Internal behavior – 6: The branch is continued

www.dbooks.org

The last step is to simply connect the *Send agreement* state to the *End* state and thus end the process. You can also do this via the pop-up menu using the last element on the right. The transition requires you to set the *Operations Manager* as the *Receiving subject* and *Agreed upon change (local)* as the *Messagetype*. The whole internal behavior, which now includes two possible branches, should look similar to that shown in Fig. 3.43.

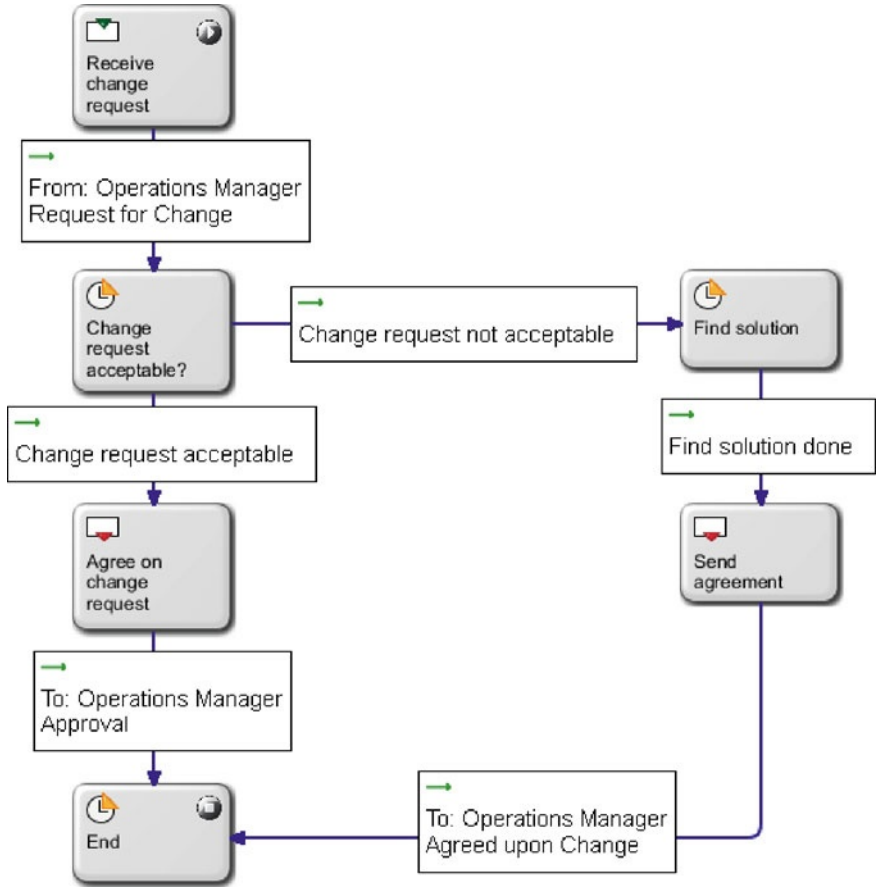


Fig. 3.43 Internal behavior – 7: The new branch is connected to the *End* state

The last possibility is the case where the *Logistics* subject either does not read the received message (which causes the operations manager to execute the alternative) or the *Logistics* subject is processing the received message while the timer runs out. If the timer runs out and the operations manager executes an alternative, the process must also be terminated for the logistics subject – no matter what state it is in.

The simplest way to achieve this is to turn every single state within the logistics subject into an *End* state. This means that the process may terminate within that state but isn't necessarily doing so. In this case the process is only terminated if it reaches the *End* state within the operations manager's internal behavior.

In practice, this means selecting each created state and checking the *Is end state* check box, even of the *Receive change request* state – it is then a start and end state at the same time.

After you turned all six states into end states, the complete behavior of the logistics subject should look similar to Fig. 3.44.

Note the icon, which indicates that every state is now an *End* state.

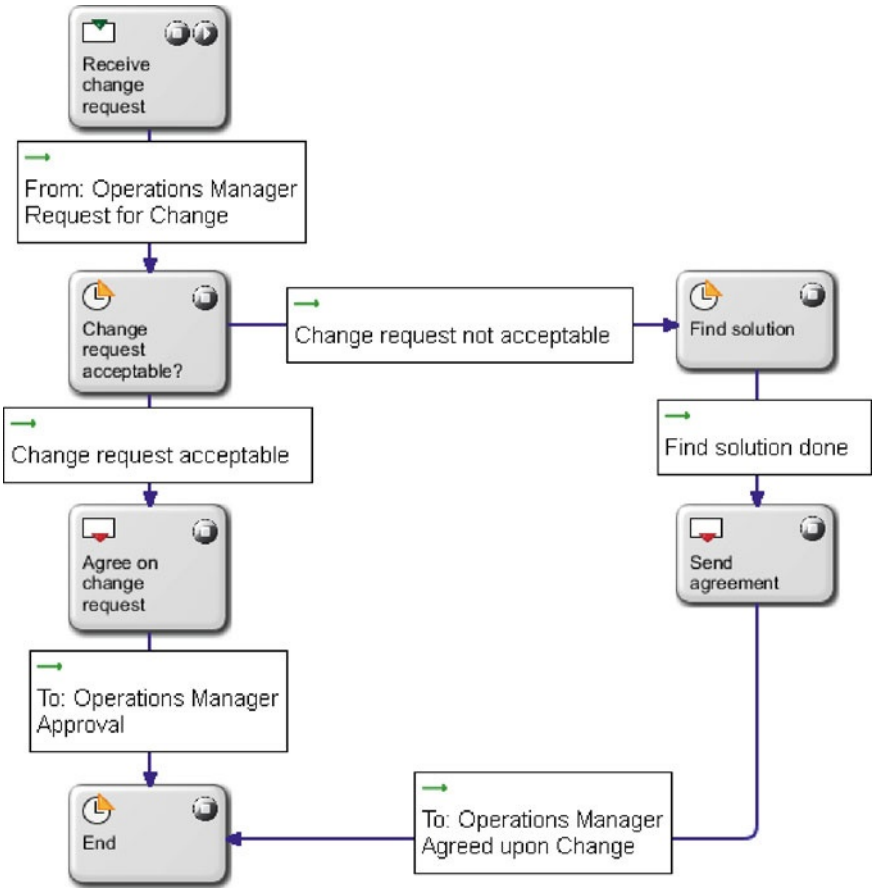


Fig. 3.44 Internal behavior – overview: The internal behavior of the *Logistics* subject is now also finished

3.2.5 Enacting and Executing the Process

After you have created the whole process model consisting of two subjects, their communication and internal behavior, the next step is to enact and then execute this model. An executed model has the advantage that all existing possibilities can be validated. It is also possible to check whether there exist possibilities which were not taken into account during the design phase.

Validation Environment

The previously created process can be executed directly within the *Metasonic Suite*. It can also be played through (step by step) using actually existing user accounts, which are assigned to a subject.

www.dbooks.org

The accounts can be mapped to an existing *active directory* (user accounts already defined in your IT infrastructure).

Prior to executing the process, these accounts have to be created by mapping the business organization onto the process. They are stored in a separate database within the validation environment provided by the *Metasonic Suite*.

The first step is to start the validation environment by choosing *File/Start Validation Environment* from the menu (Fig. 3.45).

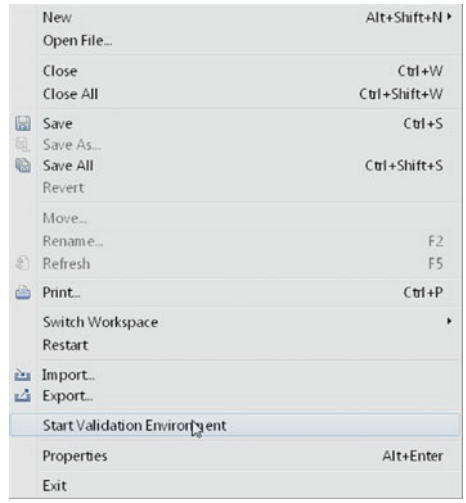


Fig. 3.45 Starting the validation environment from the menu

After you clicked on this menu item, the validation environment is being started which can take a while, depending on your hardware. After the environment is started, the whole window should look exactly as shown in Fig. 3.46.

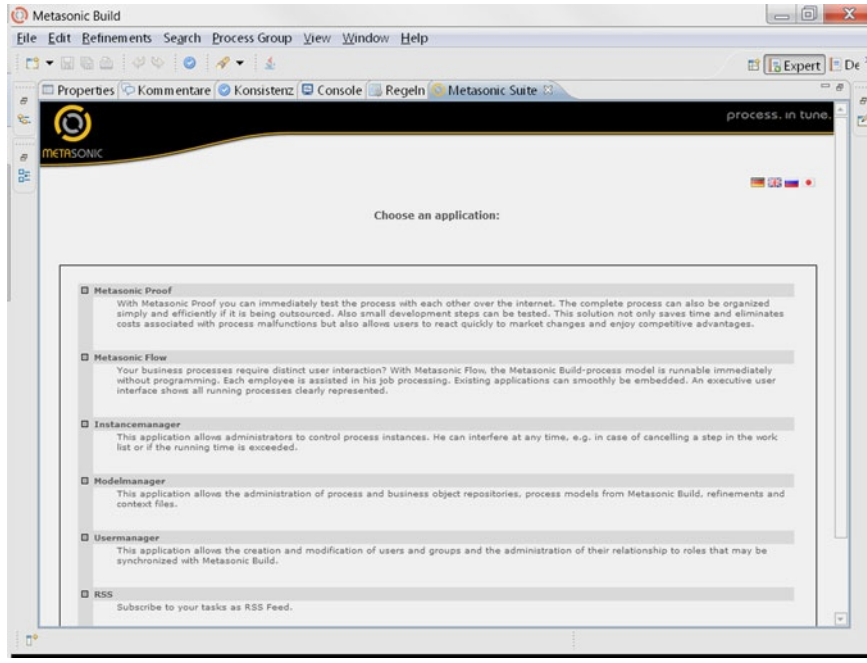


Fig. 3.46 Validation environment window overview

This window will be used for validation/execution purposes.

Users, Roles, and Groups

In this section we have to define who is doing what. We have defined subjects including their internal behavior. Now we map the organization onto these subjects (or processes, if you like). This includes the definition of roles as an abstract representation of responsibilities within an organization. At the end, we then map groups and real users onto those roles, which again are linked with subjects. This is the most flexible way to design the enactment of business processes.

The *Metasonic Suite* uses a typical three-level scheme to organize user accounts and to connect them with the subjects created in the process model (Fig. 3.47).

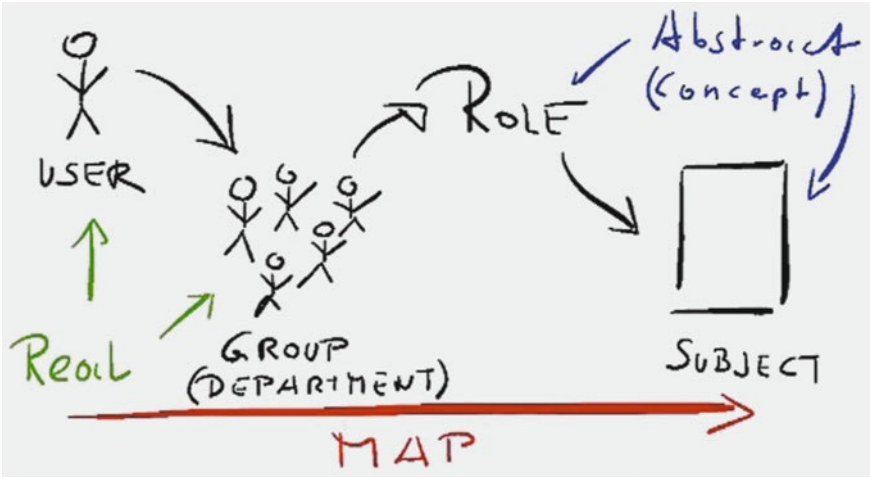


Fig. 3.47 Mapping users with subjects

On the first level are the user accounts, which can be assigned real names and which represent actual humans in the real world. The second level consists of user groups, which users can be assigned to and which can represent, for example, departments. The third level consists of roles which link the user groups to the created subjects.

For our company, the following organizational structure exists (for a visual aid, see Fig. 3.48):

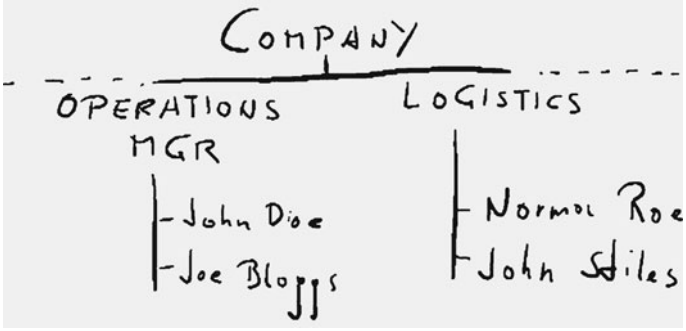


Fig. 3.48 The company's structure

www.dbooks.org

Within the operations management department there are two employees:

- John Doe
- Joe Bloggs

Within the logistics department there are also two employees:

- Norma Roe
- John Stiles

This structure has to be created within the *Metasonic Suite* so John Doe and his colleagues can login and start working with the *Metasonic Suite*.

The first step is to create the necessary roles. To do so, open the *Usermanager* from the (previously started) validation environment window with a left-click. The *Usermanager* is the second link from the bottom on the *Metasonic Suite's* validation environment window (in full-screen mode) as shown in Fig. 3.46.

The *Usermanager* opens a new page in a web browser and offers a selection of three functionalities: User-, Group-, and Role administration. The easiest approach is to start creating the groups because this results in the least amount of clicks needed for this scenario.

To create groups, select the menu item *Group administration* from the menu on the left-hand side. The groups in our scenario represent the departments in which the users work; therefore, in this case, they are named "Operations Manager" and "Logistics".

Create a new group by selecting the *Create new group* link after having selected *Group administration* from the menu on the left-hand side as shown in Fig. 3.49.

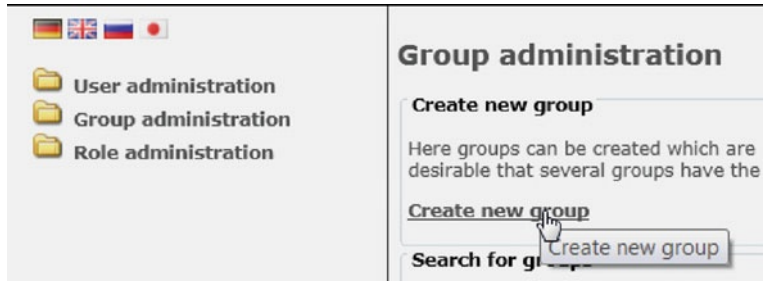


Fig. 3.49 *Group administration* browser window – create a new group by clicking the *Create new group* link

As shown in Fig. 3.50, the first group you create is named "Operations Manager". All user accounts which represent operations managers will later be added to this group.

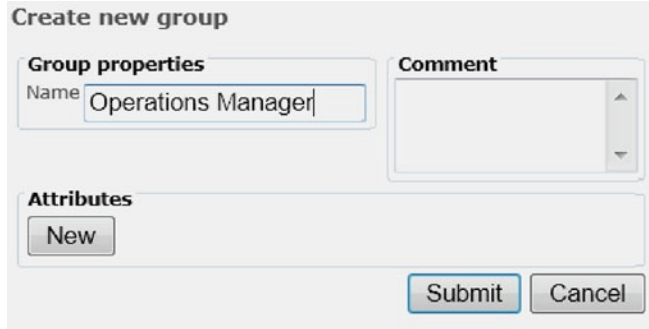


Fig. 3.50 *Create new group* browser window – the new group is named "Operations Manager"

Enter the name of the group and hit the submit button. Use the same approach for the second group named "Logistics".

The creation of both groups can further be validated by using the *Search* button with an empty input field – this displays all existing groups. Within all displayed groups there should also be the two groups you just created.

The next step is the creation of the four user accounts, which is done via the *User administration* link from the menu. The *User administration* browser window looks very similar to the *Group administration* browser window. Create a new user account by clicking the *Create new user* link.

Create the first user called John Doe with the login name of "jdoe" and the password "topsecret". To link the account with the group "Operations Manager", click the *Search* button within the user creation form and add the element "Operations Manager" to the *Associated* tab with the [→] button. After you have filled out all necessary information, the *Create new user* form should look like the one in Fig. 3.51.

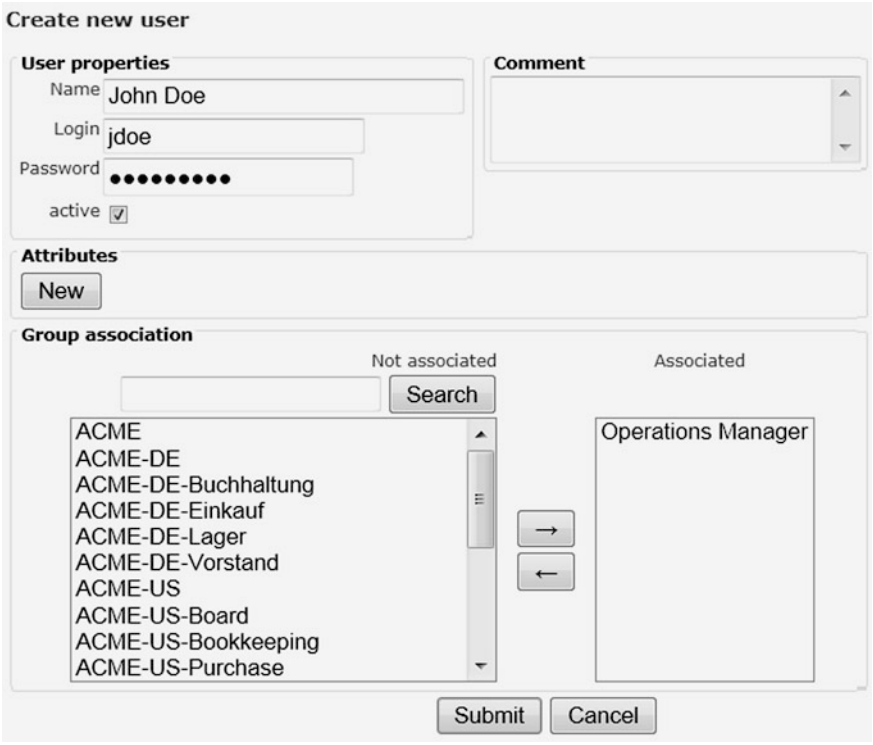


Fig. 3.51 *Create new user* browser window. Fill out the text boxes accordingly

Repeat this step for the user account of "Joe Bloggs" with the username "jbloggs", the password "topsecret" and the group "Operations Manager".

Use the same approach to create "Norma Roe", "nroe", "topsecret", group "Logistics" and "John Stiles", "jstiles", "topsecret", group "Logistics".

After the creation of all users, they should all be listed when using the *Search* button on an empty form in the *User administration* section as shown in Fig. 3.52.

www.dbooks.org

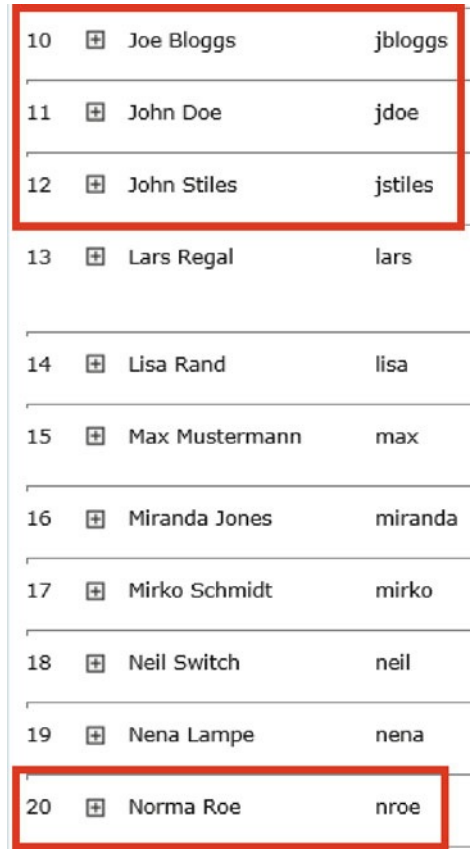


Fig. 3.52 *User overview* browser window – make sure you find your previously created users there!

The last step is to create the necessary roles. To do so, select the *Role administration* item from the menu.

As previously mentioned, roles link the groups to the subjects in the process model. Therefore, it makes sense to name the roles according to the subjects.

The first role to create is the "Operations Manager". Do so by selecting *Create new role*, as previously described.

Name the first role "Operations Manager" and link it to the group *Operations Manager*. The group can be selected by using the *Search* button with an empty input field and choosing the previously created *Operations Manager* group. Add the group to the *Associated* tab by using the → button. The resulting form should look like the one in Fig. 3.53.

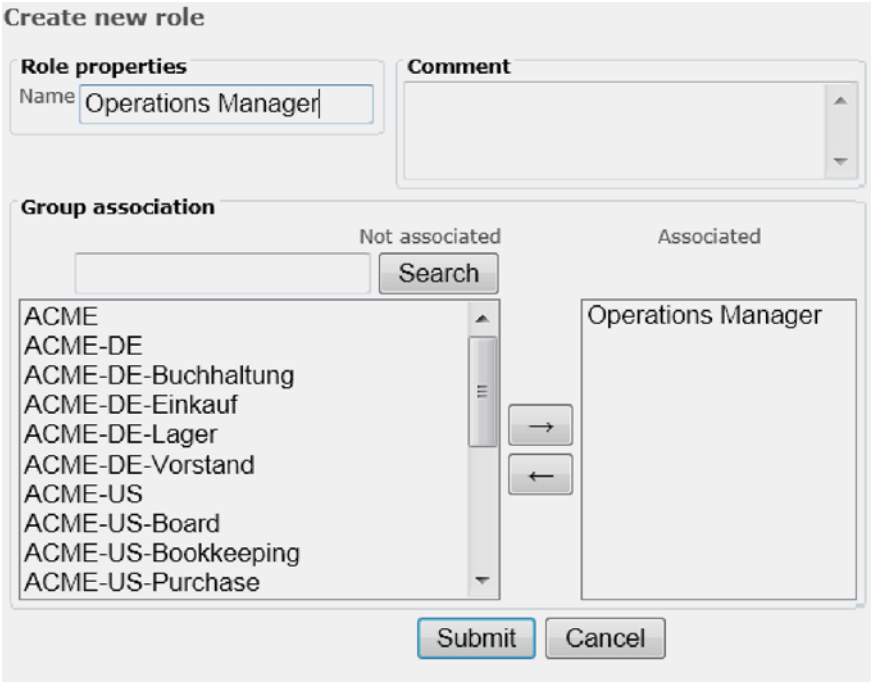


Fig. 3.53 *Create new role* browser window – Make sure that the new role is associated with the corresponding group!

Submit this form and use the same approach for the role *Logistics*, i. e., you create a role with the name "Logistics" and the *group association* "Logistics".

Afterwards, the two created roles should be displayed when using the *Search* button with the empty input field in the *Role administration* browser window, as shown in Fig. 3.54.



Fig. 3.54 *Role overview* browser window – Check the successful creation of the two roles

After you created all necessary groups, roles, and users connect the roles with the subjects in the process model. Close the browser window and open the *Metasonic Suite* again. A double-click on the *Metasonic Suite* tab minimizes the validation environment and brings back the process overview.

The first step is to bring up the *Roles for process group* window. To do so, right-click the process group named "Teaching factory goes S-BPM". Then select *Metasonic Build – process group specific settings/Roles*.

This step opens the *Role repository* window (Fig. 3.55).

www.dbooks.org

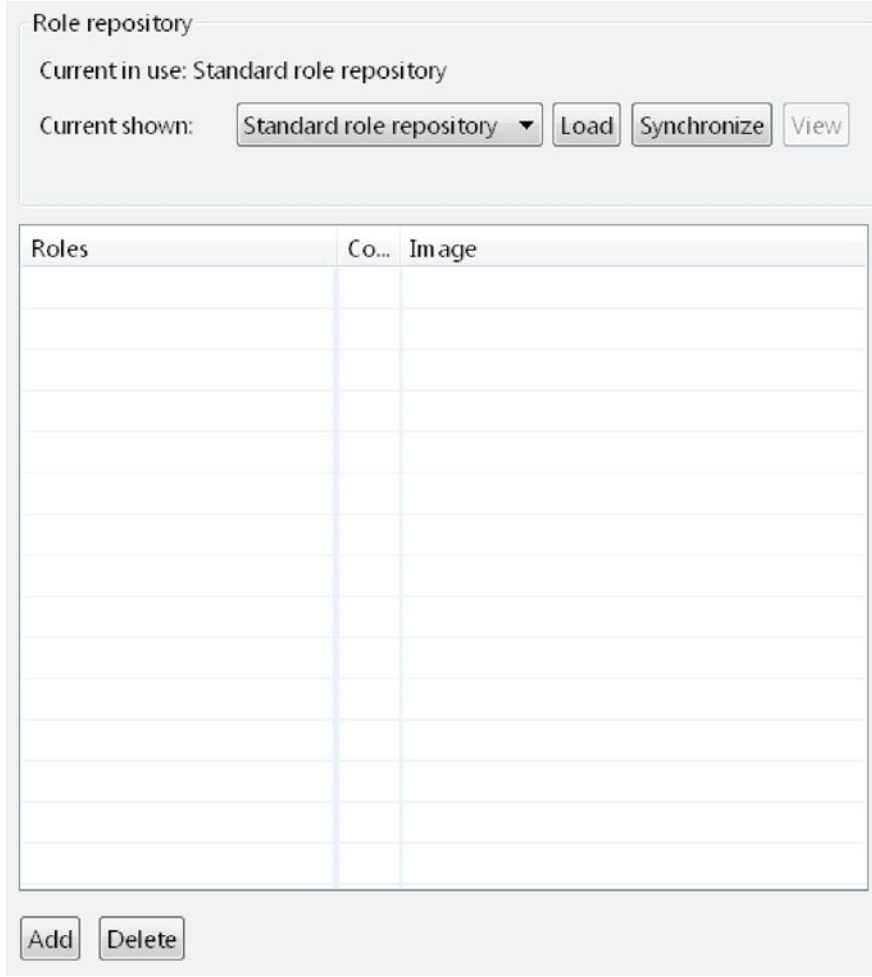


Fig. 3.55 *Role repository* window – here all synchronized roles are displayed. It is also fine if it is empty

As the next step click *load* after selecting *Standard role repository* from the drop down menu in the center of the screen. Select *Usermanager* in the following *Roles match* window as shown in Fig. 3.56.

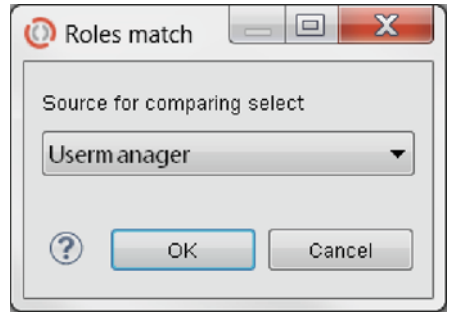


Fig. 3.56 *Roles match* dialogue – be sure to select *Usermanager*

After you clicked the *OK* button, all existing roles should be displayed within the *Role repository* window, which includes the *Operations Manager* and the *Logistics* roles. Click the *Synchronize* button and again select the *Usermanager* in the *Roles match* window to synchronize the current process group with the previously created roles. After you saved the whole file (e. g., by using the shortcut Ctrl + S), the roles are ready to be linked with their respective subjects.

Return to the *process group overview* window which shows the two existing subjects and their communication (Fig. 3.14).

Here, select the "Operations Manager" with a right-click. In the following context menu, click *Properties*.

In the *General* tab of the *Properties* window there is a drop-down menu labeled *Role*. Set this option to "Operations Manager", which should be selectable (Fig. 3.57).

If it is not selectable, please go back to the start of this section and ensure that you followed all steps correctly.

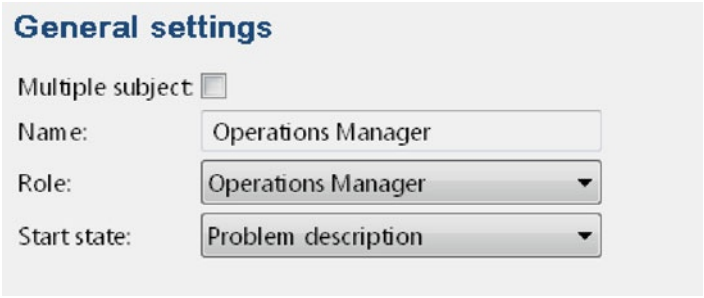


Fig. 3.57 *General settings* window of the *Operations Manager* subject. Make sure the corresponding role is selected correctly

Use the same approach to select the *Logistics* subject and set its role to *Logistics* (Fig. 3.58).

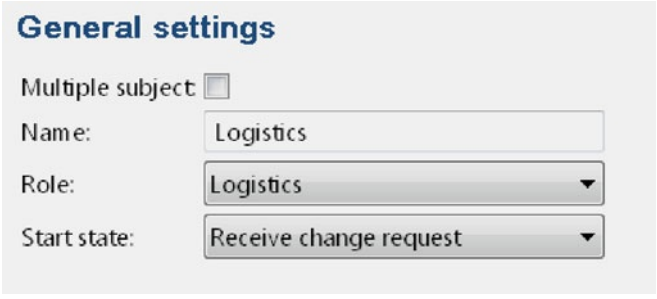


Fig. 3.58 *General settings* window of the *Logistics* subject. Same here

This finalizes the links between the process model and the users, roles, and groups. Save the file again, and now it is ready to be uploaded and executed.

Metasonic Flow

In this section we discuss how to upload a concrete S-BPM model (the one you created) into a repository where it can be executed by the process execution engine. In the repository you can store different versions of a process for example for documentation purposes. The upload step can also be seen as a formal release step, so that only completely designed processes are enacted.

www.dbooks.org

The *Modelmanager* is the third item from the bottom. If the validation environment is not already started, see Sect. 3.2.5.

Do so by clicking the *Modelmanager* link in the *Metasonic Suite's* validation environment overview window.

After the *Modelmanager* has opened in a web browser, the overview window is displayed. The next step is to create a new *Process Repository*. Create a new repository by entering "Teaching Factory goes S-BPM" in the only available text input field and clicking the *Create* button.

After you created the repository, create a folder within that repository which is called "Scenario 1". Create a folder by using the form available on the right-hand side of the screen after you selected the previously created *Process Repository* (Fig. 3.59).

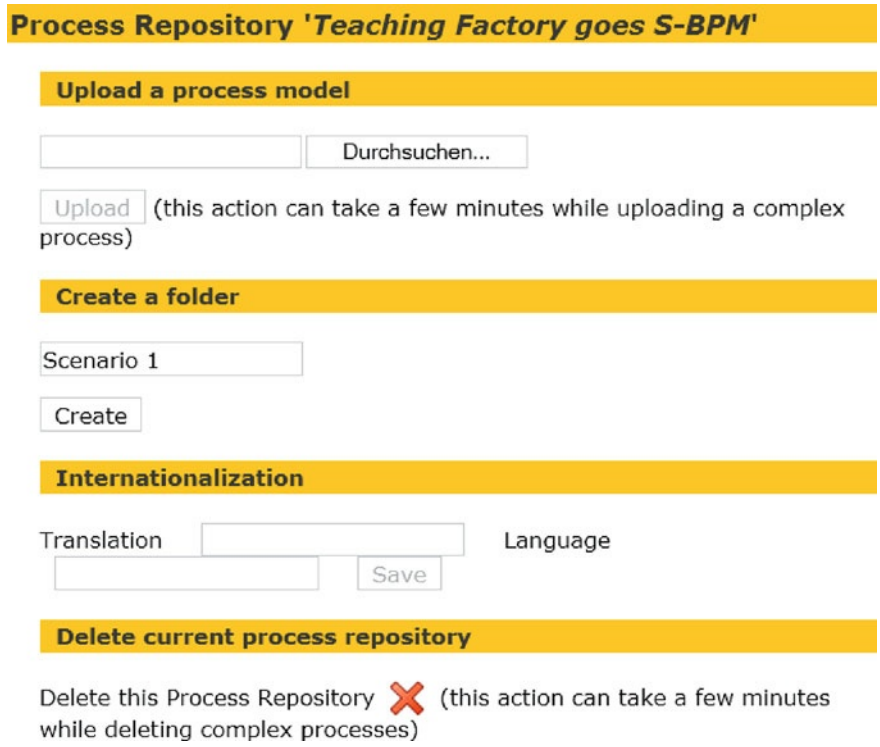


Fig. 3.59 *Process Repository* window – detailed view. Here it is possible to create new (sub-)folders or to upload processes

After you created the folder, the process model is ready to be uploaded. Select the newly created folder and click the *Search* button to search for the process file, which contains the modeled process, on your hard drive. In this case the file you are searching for is the file

"Scenario_1_-_Operations_Manager_and_Logistics.jpp", which should be located in a folder labeled *src*, typically placed within the project folder. The project folder in our case is called *Teaching Factory goes S-BPM*. This is also the work space location, which was selected at the very beginning when you first started the *Metasonic Suite*.

After you selected the desired process file, it can be uploaded by clicking the *Upload* button as shown in Fig. 3.60.

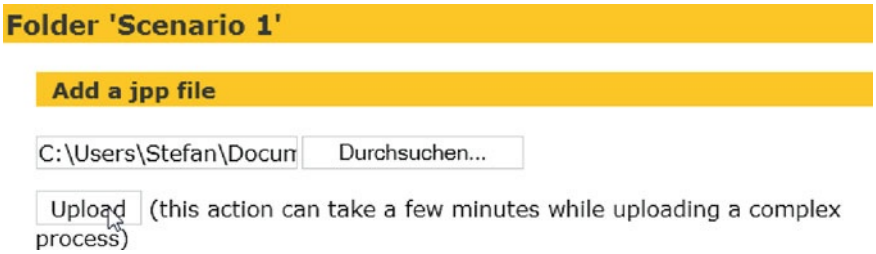


Fig. 3.60 Upload the process model from your hard drive

The uploaded process file is automatically selected after uploading. The only thing left to do before you can start with the execution is to check the *Metasonic Flow start* check box and, afterwards, click the *play* button next to it (Fig. 3.61). If you encounter an error, please check if all roles contain users and each subject has been assigned a role.

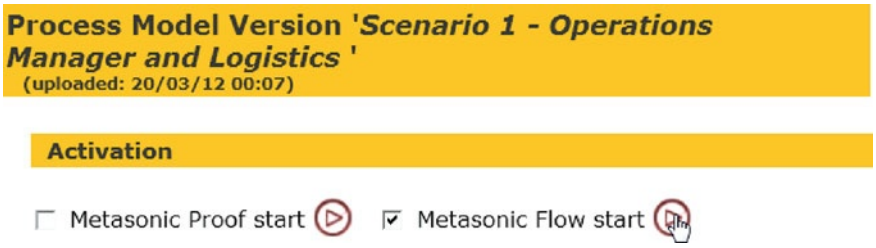


Fig. 3.61 Make the uploaded model ready for execution by checking the *Metasonic Flow start* check box

The process is now ready for execution, and a login window, which is shown in Fig. 3.62, pops up.



Fig. 3.62 *Login* screen – login with existing user credentials

It is now possible to login here with the previously created credentials and execute actions depending on their linked subject's behavior.

The *login window* is actually the start page of *Metasonic Flow*, which can also be opened by selecting the second link from the top at the *Metasonic Suite's* validation environment overview window. After uploading the process file and activating it for execution via the check box, there is no further need to use the *Modelmanager*. To execute this process, only the *Metasonic Flow* window is needed.

For more details, see Sect. 3.2.5.

In the next step all possible process variations can be played through for validation purposes (or just for fun). In this case you will only try out the sunshine (or happy) path: an error occurs in the manufacturing process, which causes the operations manager to send a *Request for Change* message to the logistics subject; a certain schedule cannot

www.dbooks.org

be met which is then accepted and confirmed via an *Approval* message by the logistics department.

A typical setup could be a server machine hosting the model repository and any number of clients as working machines for each user.

You will play this process through by only using one physical machine, therefore always logging out and in again with the corresponding subjects. In a real world environment, of course, every subject has their own machine.

So the first step is to login with an operations manager user account such as John Doe.

Log in with username "jdoe" and password "topsecret". Afterwards, the *Metasonic Flow* home screen is displayed (shown in Fig. 3.63.)

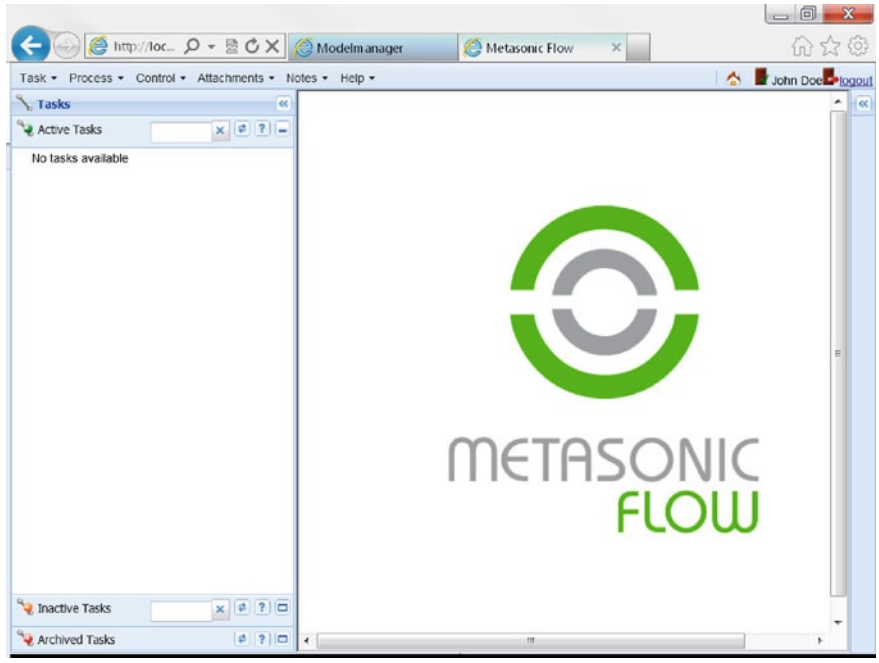


Fig. 3.63 *Metasonic Flow* overview – from here it is possible to start new tasks or to work on existing ones

The next step is to start a new process instance as John Doe, operations manager. Do so by selecting *Task* from the menu bar and then *New Task* from the drop-down menu located at the top left of the screen.

The next window asks which process to start. Here the only possible choice is the previously uploaded process. The window should at least include the folder structure shown in Fig. 3.64.

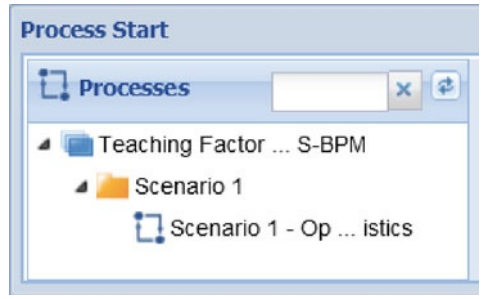


Fig. 3.64 *Process Start* window – select the right process to start. If you did everything right, there should be only one to select

After selecting the "Scenario 1" process a form appears where several settings can be adjusted. It is recommended to change the *Title* of the process instance to explicitly describe the purpose of that instance. In this case, rename the *Title* to something like the current date and time followed by "– Scenario 1 – Sunshine Path" (example shown in Fig. 3.65).

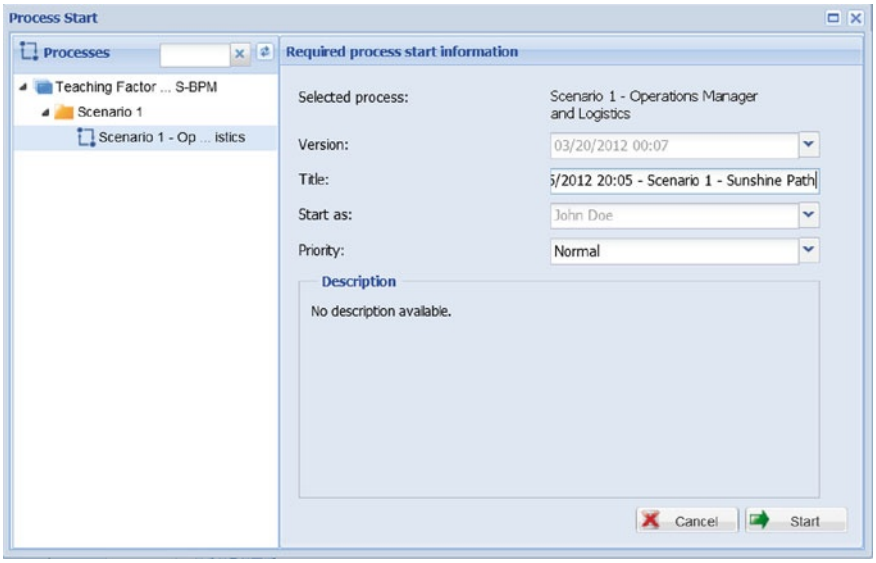


Fig. 3.65 *Process start* window. Make sure you give your process instance a meaningful name

After that, click the *Start* button – a new process instance for this process is started and the whole browser window should look like Fig. 3.66.

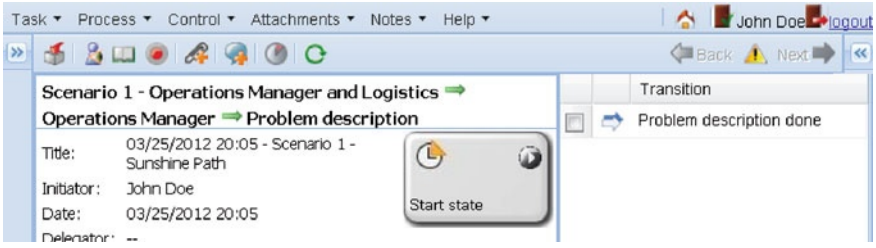


Fig. 3.66 Problem description – this is what you should do

At the top right of the screen the currently active task "Problem description" is displayed.

You, enacting the process instance as John Doe, can perform this task by using the *Parameters* tab from the previously shown Fig. 3.67. The *Parameters* tab shows all available parameters which can be used to describe the production problem in more detail (Fig. 3.67).

…which is the third tab from the right

www.dbooks.org

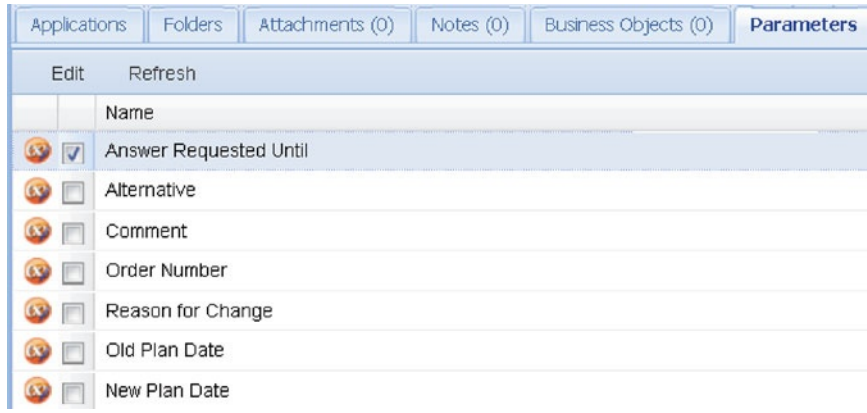


Fig. 3.67 *Parameters* tab – here you can describe your problem in detail

Open the *parameter editing window* by double-clicking one of the parameters shown. Use this window to further describe the problem. An example is shown in Fig. 3.68. In this case the parameters are not arranged in a specific order.

Use business objects to model all thinkable data structures and generate a nicer presentation.

Please note that the parameters are only used as an easy-to-use and easy-to-explain solution. For a more advanced and visually more attractive solution, there is the possibility of using business objects. We do not concern ourselves with business objects because in this book we want to focus on the communication aspects of processes and we do not want to show how to define forms.

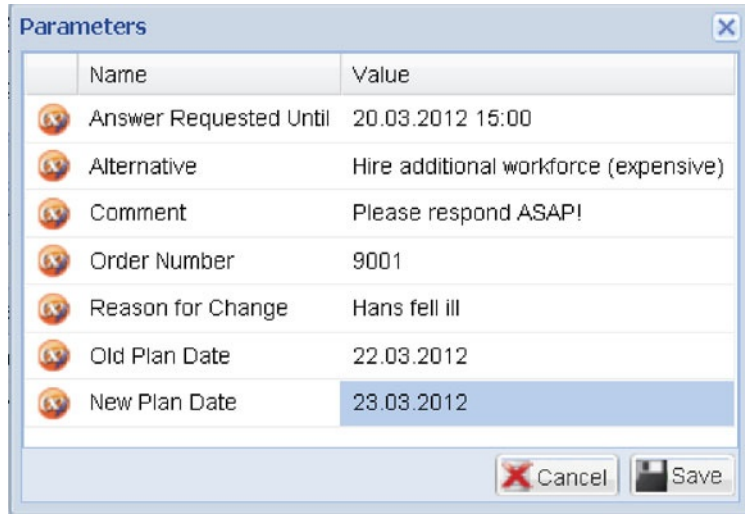


Fig. 3.68 Editing parameters – this is an example of what a problem description could look like

Click the *Save* button to save all entered parameters. Afterwards, check the *Problem description done* check box (see Fig. 3.66) and enter the next stage of the process by pressing the *Next* button at the upper right of the screen.

The next window is similar to the previous one, except that an image indicates that the process is now in a send state of the internal behavior, as highlighted in Fig. 3.69.

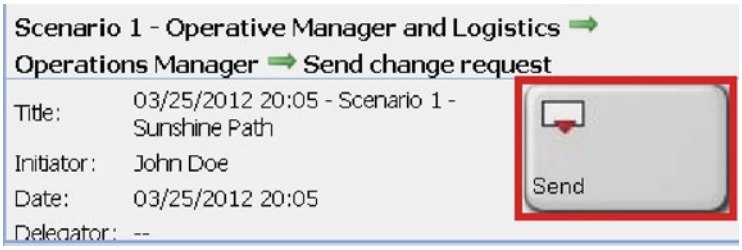


Fig. 3.69 The current state of the process is always indicated by the symbol highlighted with a red square. In this case the process is in a send state

Send the specified parameters as a message by clicking the *To* button (marked in red in Fig. 3.70). Here it is possible to send the message to a single person or to a whole department. In this case, send the message to the whole department, because you do not care who in the logistics department takes care of the problem (Fig. 3.71).



Fig. 3.70 This is what the *To* button looks like



Fig. 3.71 Send your message by selecting the person or department which should receive it

Save the selection and click the *Send* button to send the message. John Doe now waits for an answer to arrive or the timer to run out.

For the next step, logout as John Doe by using the *logout* link at the top right of the screen, and login as one of the members of the logistics department. After you logged in as a logistics user, the currently running process is displayed within the *Active Tasks* area of the browser window.

The login could be for example as Norma Roe from the story part: username "nroe", password "topsecret".

Double-click to open the task. In the opened window select the only available instance by double-clicking it.

A browser window opens and indicates that the internal behavior of the *Logistics* subject is currently in a receiving state.

Click the check box to activate and click the *Receive* button to receive the message from the inbox. In the next window, read the received parameters in the *Parameters* tab as shown in Fig. 3.72.

www.dbooks.org

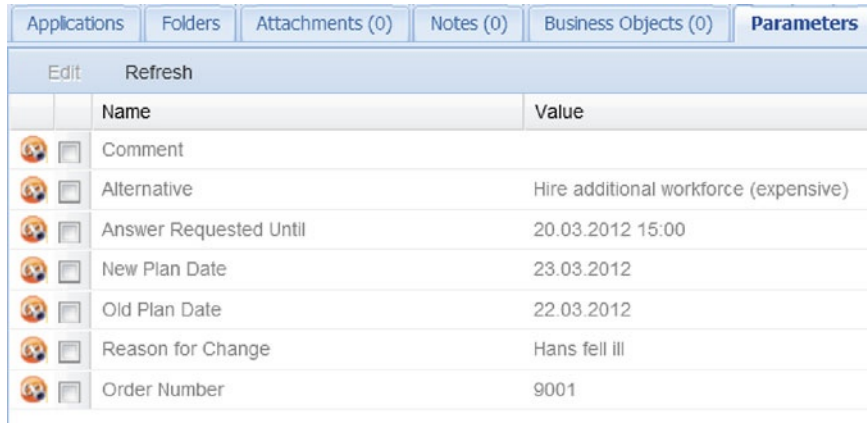


Fig. 3.72 An example of the received parameters (which are read only)

After reading the parameters, you can decide whether they are acceptable or not. In this case we assume that the change request is perfectly acceptable and therefore activate the *Change request acceptable* check box. Then click the *Next* button and the *Approval* message is prepared to be sent. Afterwards, in the *Parameters* tab, you can also edit a *comment* parameter if you like. In this case it is assumed that everything is perfectly fine without the need for any further comments. Therefore, select "John Doe" as the recipient of the message in the *To* form (which opens after you click the *To* button), as shown in Fig. 3.73 and save the selection. Finally, send the message by clicking the *Send* button.



Fig. 3.73 Sending a message back to "John Doe"

Following this action, the process is finished for the *Logistics* subject and there is nothing more to do, which is indicated by the *End state* image in the next window.

Logout of the current user account and login as John Doe once more to receive the approval message.

Open the only active *Task* on the left-hand side with a double-click and afterwards the only *instance* available, also with a double-click.

Receive the message, in this case sent by Norma Roe (Fig. 3.74), by checking the check box and clicking the *Receive* button.

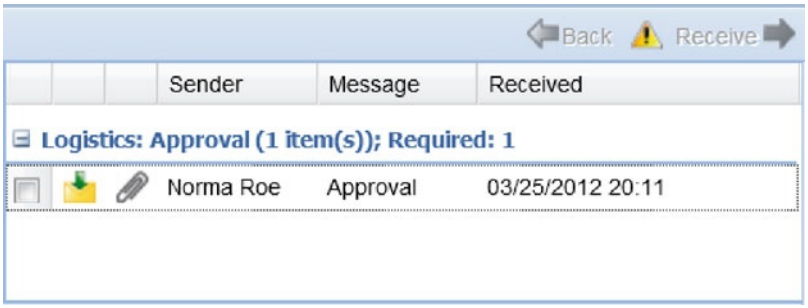


Fig. 3.74 Receive the message by checking the check box and clicking the *Receive* button

After that, the process has ended for John Doe and is terminated. Now that the sunshine path has been validated, all other paths could be validated as well, before using the process in a real production environment.

As already mentioned, the message parameters are neither restricted (e. g., it is possible to enter plain text into a date field) nor arranged in a particular order. To avoid these flaws and to create nice-looking messages, the *Metasonic Suite* offers a feature called business objects. Business objects do not alter the behavior of the process in any way, but affect the visual appearance of the message forms (business objects will not be handled in this book).

3.3 Accomplishments

After you modeled the whole process and enacted it for execution, the process is principally ready to be used in a productive environment.

By handing over the login details to the participating subjects, the process addresses the communication problems outlined at the beginning of chapter 2.

After some time and several process executions, which of the possibilities occurs most often can be evaluated: the "sunshine path" or any other variation? This could lead to a possible process improvement.

The most important accomplishment of the new process should be that employees solve unexpected problems in a structured and well-documented way by using the *Metasonic Suite* and other forms of communication. All processes started in *Metasonic Flow* are logged and can be analyzed for various purposes. For example, one can analyze how often a process is finished with the best possible outcome. Nevertheless, the modeled processes are for demonstration purposes and we hope you can see the power of S-BPM.

3.4 Lessons Learned

After you worked through this chapter, the following concepts of S-BPM and the *Metasonic Suite* as a supporting system should be clear:

- Creating a new Process Group
- Creating a new Process within a Process Group
- Creating Subjects
- Creating Messages, including parameters
- Creating communication flows between subjects

www.dbooks.org

- Modeling the internal behavior of subjects
- The concept of users, groups and roles
- Creating users
- Uploading processes
- Executing uploaded processes using *Metasonic Flow*

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

Transition – Part I

After John Doe played through the whole scenario on his notebook while Norma played the other part on her notebook, he admitted: "I'm baffled … If anyone told me that you could create executable processes this fast I wouldn't have believed it. Seriously – it is barely lunchtime!"

"Yes … it is really amazing that we were able to finish this whole process in such a short time," Norma added. "You know, you are not the first consultants to visit our company. But you are the first ones to provide us with a running solution after only a few hours."

John then asked a question, which he had been itching to ask since he started *Metasonic Flow*: "We have now created a process and executed it. But how does that help our departments? I mean could we take the process we just created and make our departments use it?"

Al answered: "Long story short: yes you could. You would need to install *Metasonic Flow* on one of your servers and probably connect your user database to it. But yes – you could definitely take that process and start using it in practice. With no alterations."

John and Norma looked at each other, impressed. Norma then added: "And if we, for example, found that we had to alter the process? Do we have to start anew?"

This time it was Bob, who answered with a smile: "No, not at all. You just make the alterations in the *Metasonic Suite* and upload the process again in the *Modelmanager* – nothing else."

"We could do this by ourselves? Well, now I'm definitely even more impressed," John told them.

The consultants smiled at each other and one of them said: "And that's not even all we are capable of. We still have a lot of work to do until we are finished here."

John asked: "Still work to do? What is missing? We now have a complete process which we could use in our departments – if we install the program, as you said."

The other consultant smiled: "Well, we still have some things we want to show you to enable you to grasp all the concepts and techniques of subject-oriented business process modeling – but to start with, I suggest we have lunch."

"Now that's a good idea," Norma replied. "But today's coffee is on me. Free coffee for the superheroes of the computer age!"

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_4, © The Author(s) 2013

www.dbooks.org

The Problem – Part II

Observe constantly that all things take place by change, and accustom thyself to consider that the nature of the Universe loves nothing so much as to change the things which are, and to make new things like them. *Marcus Aurelius*

After John, Norma, and the proclaimed "superheroes of the computer age" had lunch, they went back into their room to continue working on processes. Al started by telling them: "Okay, lets get started again. I really want to do one more process today so I can consider it productive."

"Okay," John replied, "but what do you want to do?"

"You know, I have been thinking about that during lunch," Norma started. "We now have our process and everything is fine between my department and John's department. But the whole thing isn't finished even though we have agreed on a solution."

Now John also realized what she was talking about. With sudden realization he continued. "Yes, you're right – if we agree on a solution, we must tell the order processing department."

At this point, Bob interrupted the two. "Excuse me, but could you tell us what the order processing department does, exactly?"

John and Norma looked at each other and Norma started to explain. "Well, the order processing department is responsible for that whole organizational thing. You know, SAP and stuff. They create the schedule which we stick to. Like which order needs to be produced on time so there is no delay."

SAP is an ERP system.

John continued: "Yes, because if there is a delay, things could get very expensive. Like most production companies, we assure our customers that their orders will be produced by a specific date. For delayed orders, we have to pay compensation."

Norma said: "But probably we should get someone from the order processing department to join us. We don't really know how the whole process works from their point of view."

"That's true," John replied. "Let's ask Pete to join us. Peter Smith. It's him I mostly communicate with if there is an issue. He knows what we are talking about."

Peter Smith, Order Management

After that, the team decided to go and get Peter Smith of the order processing department to join them. Not even 15 minutes later, Peter showed up. "Hello guys … you are lucky that for now everything is in order and I can spare the time," he stated.

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_5, © The Author(s) 2013



Fig. 5.1 Peter joins the team

First, he and the consultants were introduced to each other. Then, Norma and John tried to explain the basics of S-BPM to him, just like the consultants did before. In fact, John thought that they were doing a pretty good job in explaining; the consultants only had to interrupt them a few times to correct something.

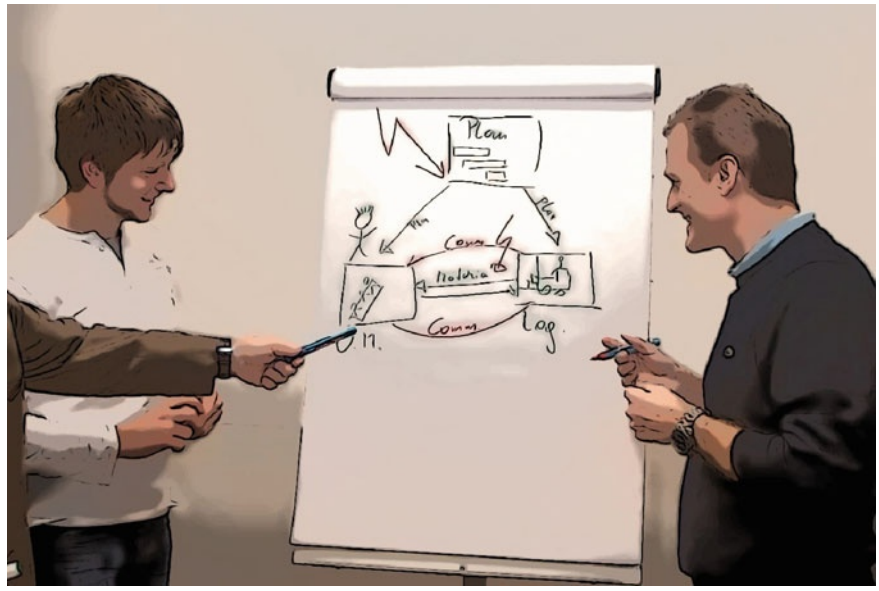


Fig. 5.2 The team is trying to explain the concept in short to Peter

After the explanation, John could see that Peter was not fully convinced. So he told him: "I can see it's hard for you to grasp all this theory. How about we just start?"

"Start with what?" Peter asked.

www.dbooks.org

"We need to know what happens when Norma and I have to switch orders," John replied. "From my point of view, it goes like this," he continued. "At first there is an exceptional process situation, which means that for some reason my department is not able to fulfill the current order. But to minimize the damage, I see that we could switch two orders, which need different materials and manpower. I negotiate with Norma's department that we switch the orders, so her department supplies us with the right raw material. After we agree on a solution, I will need to notify you that we will have to switch orders. Then either everything is fine or we have to work on a different solution. But actually, I have no idea what you are doing at that point."

"Oh, well," Peter answered, "The first thing I do is to check if the switch of orders would be possible. You know, it may be that the penalties for a delayed order would be so unbearable we would have to work out another solution. But if everything is fine, I just accept your solution and send you a notification. If not, we will have to work out a solution – just as we always do. Nevertheless, either way I will have to open my SAP and change the order there as well. You know, it wouldn't be good if the production plan in SAP was different to reality. Afterwards, I would have to re-run the MPS/MRP because some order dates will have changed. And then I will have to decide whether I need to inform the customer or not? This depends on the customer and also on the extent of the delay – if there is any. If I don't have to inform the customer, I'm finished. Otherwise I'm finished after negotiating with the customer."

MPS/MRP – Master Production Schedule/Material Requirements Planning

After listening carefully, Al told them: "Okay. Now that we all know about the process, we should again start to identify the subjects. So what do you think the subjects are here?"

John started: "Well, I guess I am definitely a subject in that process. So is my entire department."

Al took a flip-chart marker and wrote "Operations Manager" on an empty flip chart.

Peter then continued: "Well, John, if you are a subject, then so am I."

With a smirk, Al also wrote "Order Processing" on the flip chart. "Who else would we need?" he asked them.

Norma said "Ain't I a subject too? I mean, at first I communicate with John before he communicates with Peter."

Everybody admitted that Norma had a point in what she was saying. But the consultants suggested the two processes be separated. The second process would only start when the first process was finished, and could possibly start again if something goes wrong.

"So," Bob asked with a cunning smile, "have we now identified all participating subjects in the process?"

"Well, yes, I guess so," John answered, and the others nodded.

Bob then responded with a grin. "But," he said, "where is the customer?"

Never forget the customer. The customer is the most important subject to consider in a process.

John, Norma, and Peter looked at each other, and finally Norma apologized: "You are right, how could we forget the customer? Of course the customer is another subject in the process."

"But Norma," Peter said, "We don't know how a customer will behave! Contrary to our departments which have fixed processes, each customer behaves differently."

At this point, Al interrupted them and told them "Don't worry about the customer's behavior. We will show you a really cool way to model your customer without even knowing their actual behavior."

"That's nice," John responded." So we can get down to modeling now?"

"Not yet," Bob said. "First let me tell you something: the customer is the most important element of all processes. Processes that involve customers, have to fit their needs, not vice versa. Therefore, a process which affects the customer but does not consider it, is a very bad process."

While Al was writing "Customer" on the flip chart, John started thinking and realized how logical that was. Of what use was a process if the behavior of the customer just would not fit in? Also, a process could become pretty pointless if it does not consider a customer's needs.

With a flash of inspiration Norma said: "I know what we forgot. We forgot to define the messages!"

"You are right," John replied. "How could we forget such a crucial thing?"

"Don't worry," Al assured them. "At least we didn't have to tell you. You realized this point by yourselves. So I would say that you are on the right track. So – what messages do we need?"

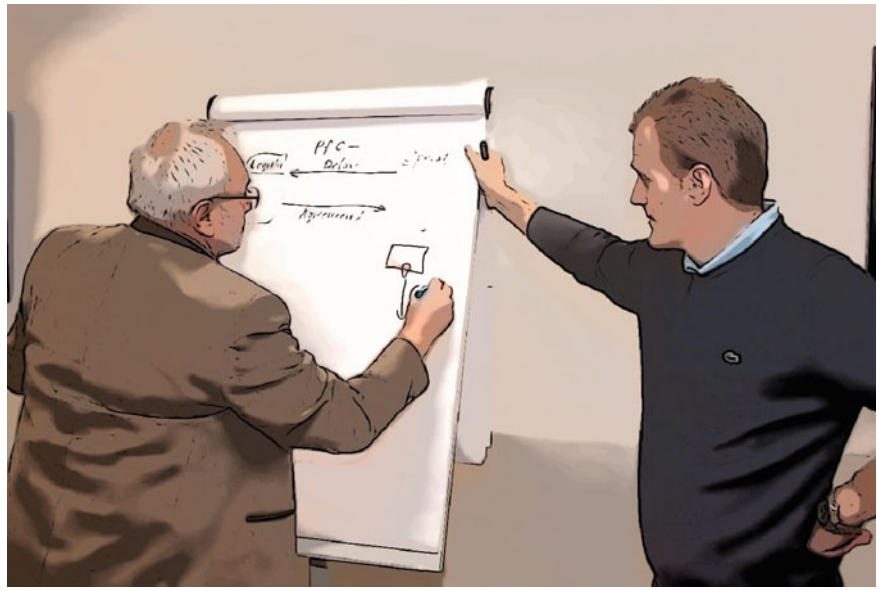


Fig. 5.3 Al and Bob start to depict the structure of the second process

Peter took the chance to answer: "Well, I wouldn't call it message. It is more a kind of notice I get. I would call it a change notice or something like that. The change notice tells me which orders need to be switched and therefore includes the production order number, an explanation of the reason for change, as well as the old and the new starting date and time."

www.dbooks.org



Fig. 5.4 Peter is skeptical about the concept at first and crosses his arms

"You know, Pete, in subject-oriented business process modeling, everything two subjects exchange is a message. Therefore, also the change notice is a message," John answered.

While Al drew the message exchange on the flip chart, Norma asked: "And you either send back an approval if everything is fine, or you talk to each other when something is not right?"

"Yes, in most of the cases it's fine, but if not we need to figure out something else," Peter replied.

Norma went on: "But this is similar to the first process. I think here we have the same types of messages. Either he sends back an approval or something like the 'Agreed upon Change' message we defined earlier."

Al and Bob nodded. "Yes, you are totally right," Al said. "This is probably because a lot of communication problems follow this scheme."

Bob then asked: "And the customer? What do you send the customer?"

Peter then replied: "Well, I guess basically the same information I got. The order number, old date, new date, reason for delay … so I guess we could just reuse the change notice."

"An excellent idea," John said. "Can we get to modeling now? If we have to start everything from scratch, I guess it will take some time."

Al replied with a confident smile, "Just wait and see. Let's get to it!"



Fig. 5.5 The group prepares to start modeling



Fig. 5.6 Al shows the group some features of the software tool

www.dbooks.org



Fig. 5.7 Norma also starts to understand the concept

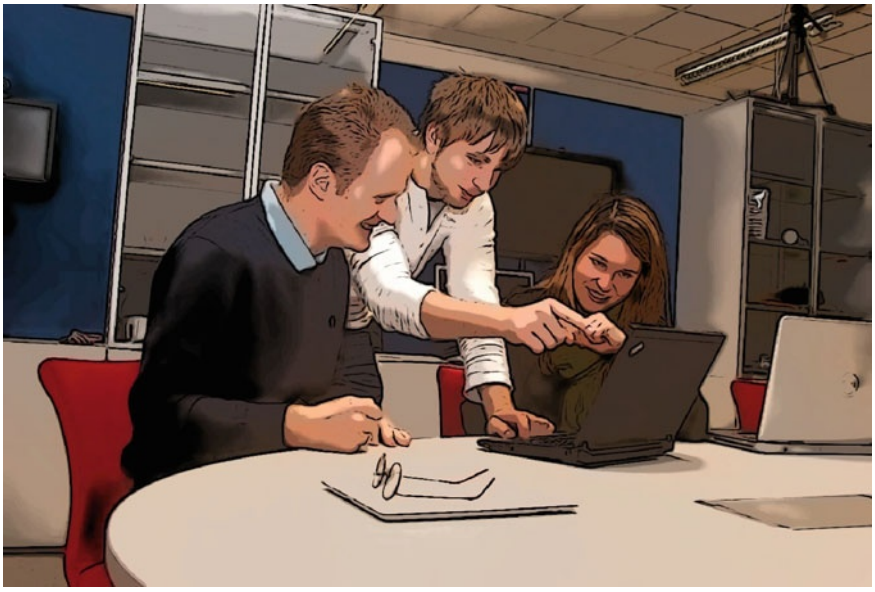


Fig. 5.8 John and Norma ask Bob some questions, while Peter still watches (not in the picture)



Fig. 5.9 Finally, Peter, John, and Norma start modeling on their own

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

www.dbooks.org

The Solution – Part II

I know that many will call this useless work. *Leonardo da Vinci*

6.1 Summary of the Problem

This scenario extends the first scenario to include the order processing department. After logistics and the operations manager agreed on a solution, this agreement has to be communicated to the order processing department which checks the changes against the ERP system. For instance, switching orders could be a problem if it leads to violating a customers due date, as this may result in compensation payments.

Also, if a change in orders occurs, the changes have to be adjusted in the ERP system. This could lead to a situation where the MPS/MRP is no longer valid.

6.2 Solution (Step by Step)

6.2.1 Copying the Process

If you want to practice, you can certainly create a new process from scratch.

One way to model a new process is to duplicate an existing (and – of course – similar) one instead of having to start modeling from scratch. In this case it is recommended that you work with a copy of the previous process because several parts of it can be reused.

To begin work, make sure the *Metasonic Suite* is up and running.

Copy the process created in the previous scenario by selecting it with a right-click from the *Navigator* and selecting *Copy* from the context menu (Fig. 6.1).

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_6, © The Author(s) 2013

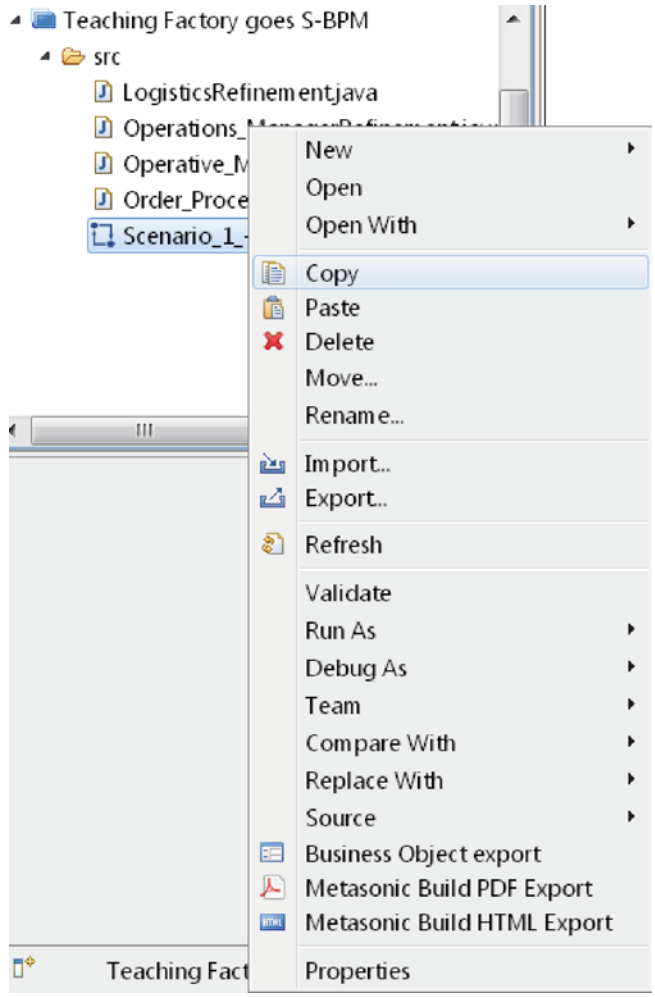


Fig. 6.1 Copying an existing process

It is important that the process is placed in that folder.

The next step is to right-click the folder labeled *src*, which is placed within the *Teaching Factory goes S-BPM* process, and then select *Paste* (Fig. 6.2).

www.dbooks.org

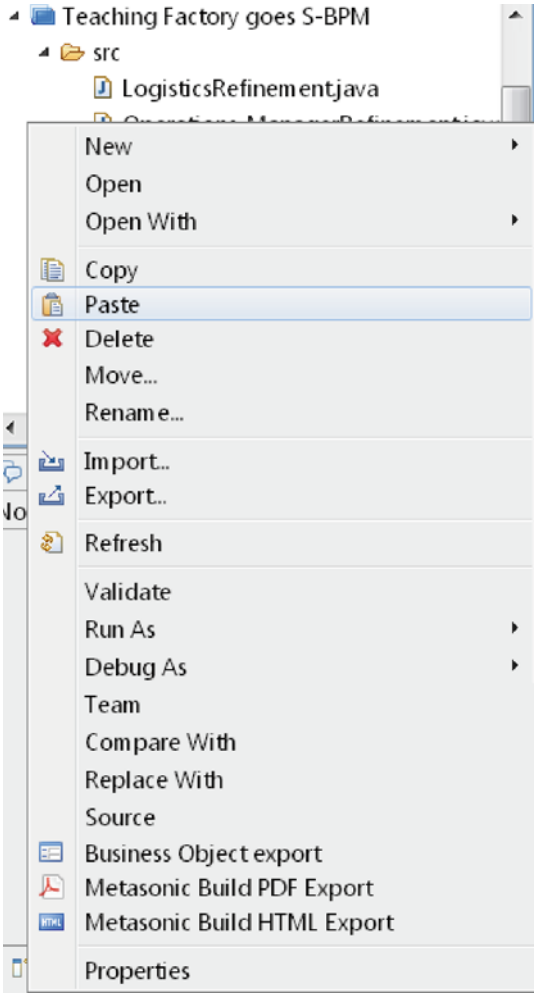


Fig. 6.2 Paste the process from the clipboard

Following this action, a dialogue box named *Name Conflict* appears. If it does not, you did something wrong. Make sure that you pasted the process in the very same folder in which the original process was placed. The dialogue box is crucial, because here you can choose the new name for the process. For our example, the new name will be "Scenario_2_-_Operations_Manager_and_Order_Processing.jpp", as shown in Fig. 6.3. This can lead a dialogue box titled *Renew identifiers* to appear. If it does, choose *Yes* as the answer.

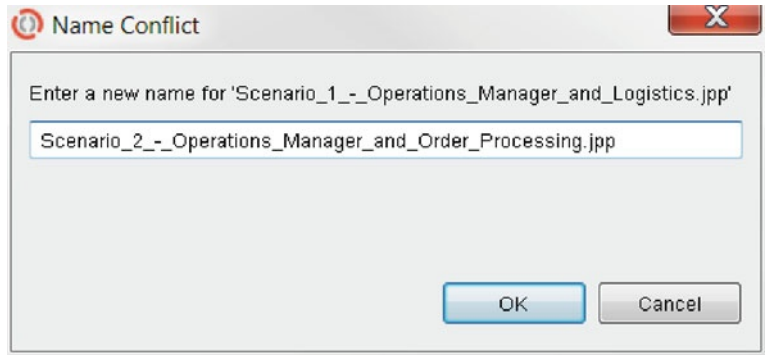


Fig. 6.3 *Name Conflict* dialogue box: choose the new name for your process

Afterwards, you will see the copied process in the *Navigator* section of the page (Fig. 6.4).

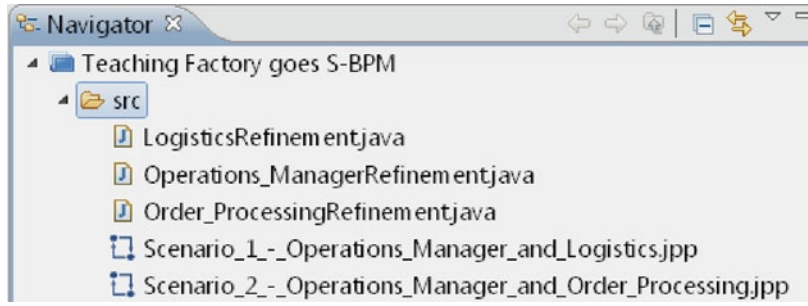


Fig. 6.4 The *Navigator* now shows the copied and renamed process

6.2.2 Altering the Process

Start working with the duplicated process by double-clicking on it.

The process should now look exactly as in scenario 1. Again, if it does not, something went terribly, terribly wrong. Please return to the beginning. If the process looks completely different from scenario 1 do not proceed.

Since this scenario takes place between the operations manager and the order processing department, rename the *Logistics* subject as "Order Processing". To do so, right-click the *Logistics* subject and select *Properties* from the context menu. There, change the text in the text box from "Logistics" to "Order Processing".

Alternatively, if you don't like hitting buttons, you can also right-click the message and select *Delete* from the context menu.

The next step is to delete the *Request for Change* message, passed between the two subjects – the newly exchanged message will be a different one. Delete it by simply selecting the message flow between the two subject boxes and hitting the DEL button on your keyboard.

After these modifications, the process should look similar to the one in Fig. 6.5.

www.dbooks.org

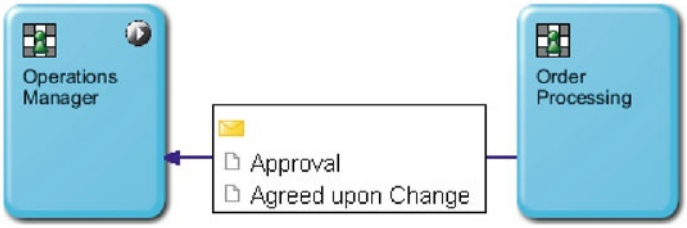


Fig. 6.5 This picture shows some of the communication behavior between two of the subjects

We need another subject in our process: the *Customer* subject. After the operations manager and the order processing departments agree on a change in orders, it may be that the customer needs to be informed because the delivery date would change.

Our customer is not an internal subject, as are all the other subjects, but an external subject. We will use this external subject for modeling purposes only. It has no internal behavior.

This concept is also known as a so-called "blackbox". We do not know how a customer behaves, therefore we just send the message and assume everything is fine afterwards.

Create an external subject by dragging the red *External subject* element from the *Palette* onto the *Canvas* (Fig. 6.6). The procedure is identical to the procedure used in the previous scenario, except that we create an external subject instead of an internal one.

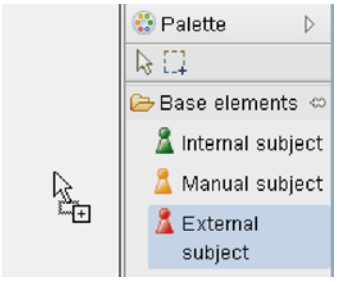


Fig. 6.6 Select *External subject* from the *Palette* and place it somewhere near the other subjects

Name the external subject "Customer" and make sure the *Model* is set to *Instant-Interface*.

The next step is to create the message which is sent from the operations manager to the order processing department and later on from the order processing department to the customer. Name the message *Change Notice* and add the following parameters:

- Production order number
- Explanation of reason for change
- Old starting date
- New starting date

The message is created in nearly the same way as in the previous scenario, see Sect. 3.2.3.2. To bring up the *Available message types* window, click the *Edit message types* button at the top of the window (Fig. 6.7).

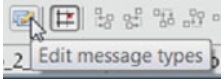


Fig. 6.7 The *Edit message types* button can be found at the top of the window

In this dialog window, create a new message type by clicking the *New…* button.

Name the new message type *Change Notice*. Contrary to all previous messages, activate the "Global Message" checkbox. This is necessary for messages, which are sent to external subjects (as in this case). Global messages are visible throughout all processes within the same process group, so they can be reused and also sent to external subjects outside the current process.

In the parameter tab, click the *Add…* button. Here, add the already existing parameters *Production Order Number*, *Explanation of Reason for Change*, *Old starting date and time*, and *New starting date and time* using the checkboxes and adding them as available parameters using the *ok button*. Finally click the *OK* button one more time and the message will show up in the *Available message types* overview window (which should be open right now). For an illustration, see Fig. 6.8.

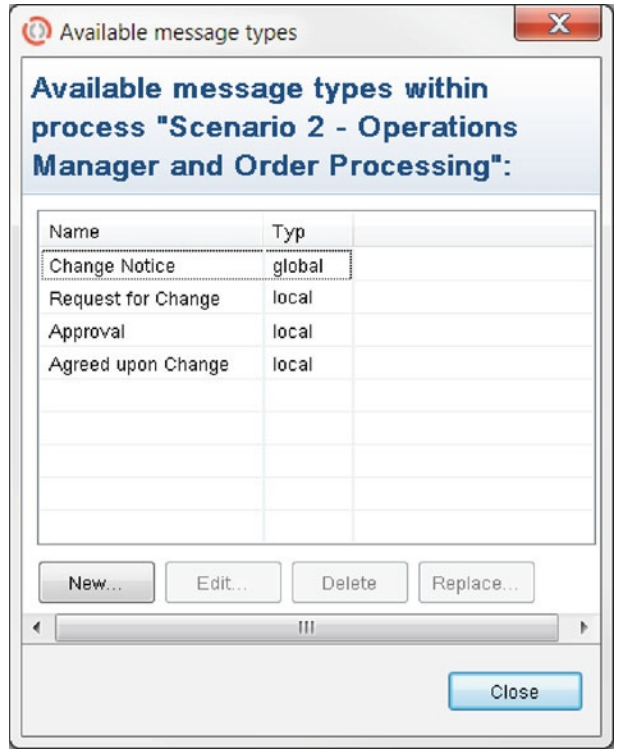


Fig. 6.8 The *Available message types* window shows all messages currently available within the process

After that, connect the *Change Notice* message from the *Operations Manager* to the *Order Processing* subject and from the *Order Processing* to the *Customer* subject.

This is done by first selecting the *Operations Manager*. Then, click the envelope symbol from the pop-up menu and connect it with the *Order Processing* subject. In the following *Create new message* window, select *Change Notice* and confirm your selection by clicking the *OK* button.

After that, click the *Order Processing* subject again, choose the envelope from the pop-up menu one more time and connect it with the *Customer* subject. In the *Create new message* window, again select the *Change Notice* message and confirm your selection. When connecting it with external subjects, the window only shows global

www.dbooks.org

messages to choose from. This means that, in this case, you should only see one message, even though several other nonglobal messages are present in the process.

Afterwards, the canvas should look similar to Fig. 6.9. Messages colored in blue are global messages.

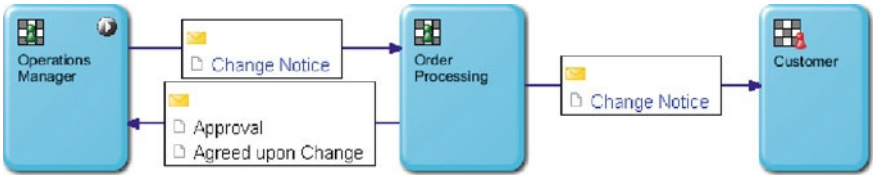


Fig. 6.9 Here you can see the finished subject overview of this process

6.2.3 Altering the Internal Behavior

After having created the basic process model, the internal behavior of the subjects has to be modeled and altered.

The first internal behavior we will modify is that for the *Operations Manager* subject. Some parts of the behavior can be reused, others will have to be deleted or altered.

The first two states and the transition between them have to be renamed. Right-click on each of them and select *Properties* from the context menu. There you can change the text in the textbox.

If you still don't see a textbox, make sure the *General* tab is selected.

- Rename the function state *Problem description* to "Change description"
- Rename the transition *Problem description done* to "Change description done"
- Rename the send state *Send change request* to "Send Change Notice"

You will also need to alter the transition state between *Send Change Notice* and *Waiting for Answer*. Change the *Receiving Subject* to "Order Processing" and the *Messagetype* to "Change Notice (global)". You can change the settings by right-clicking the transition and selecting *Properties* from the context menu (just as before).

By then, the internal behavior of the *Operations Manager* subject should look similar to the one in Fig. 6.10.

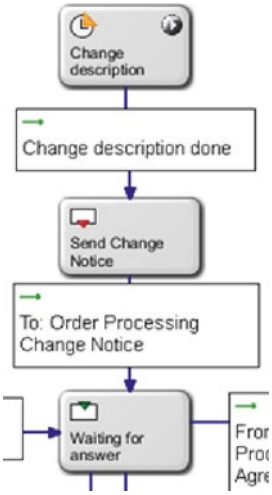


Fig. 6.10 After having altered the internal behavior, it should look like this

Sunshine/Happy path: desired path of a process.

To finish the sunshine path, alter the transition between the *Waiting for answer* state and the *End* state. Change the *Sending subject* to "Order Processing" and the *Messagetype* to "Approval".

The next step is to configure the parameters for the *Change description* state. According to the *Change Notice* message, the parameters should be configured to look as shown in Fig. 6.11. After working through the chapters, you should already know how to do this. If you need help, look it up in the previous scenario (see Sect. 3.2.3).

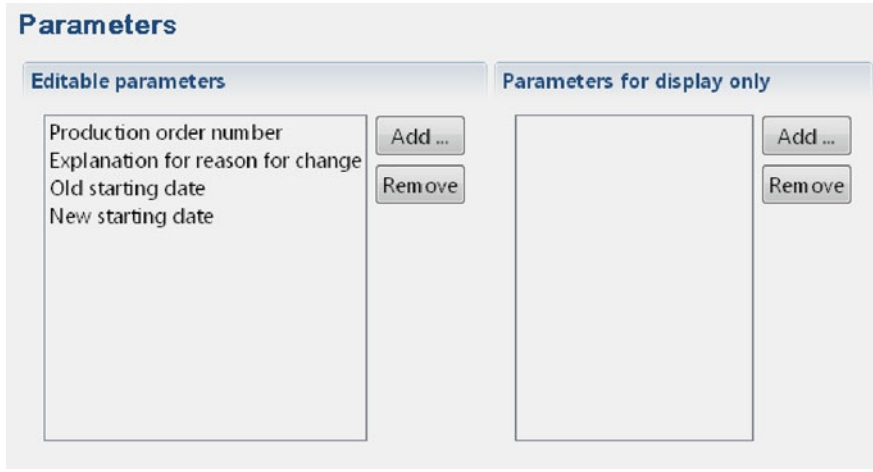


Fig. 6.11 Make sure that you add the three parameters displayed in the figure as editable parameters

If you accidentally deleted something you want back, just hit CTRL + Z.

In this case there are only two possible results: either the order processing department accepts and sends an *Approval* message or something goes wrong and the order processing department and the operations manager agree upon an informal solution. The third path, which can still be seen on the left-hand side from the previous scenario, can be safely deleted.

This is done by selecting all tasks and transitions of that path (for example by holding the CTRL key and clicking them) and deleting them by pressing the DEL button on the keyboard.

The internal behavior of the *Operations Manager* subject should now resemble Fig. 6.12.

www.dbooks.org

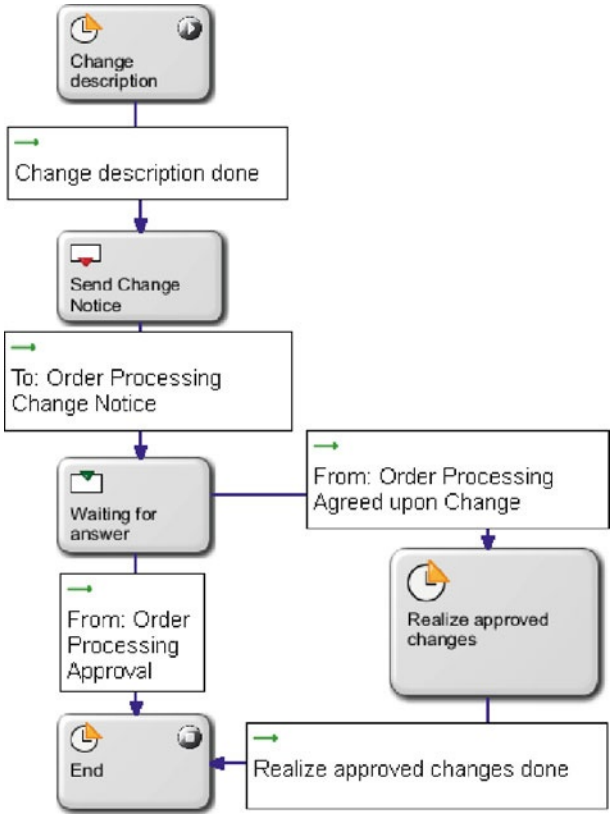


Fig. 6.12 After reworking the internal behavior according to the instructions, it should look similar to this

What is left to do is the possibility of an informal solution, which is the branch from the *Waiting for answer* state to the right. Here, ensure that the transition from the *Waiting for answer* state to the right consists of *Order processing* as the sending subject and *Agreed upon a change* as the message.

In the next step, rename the *Realize approved changes* state. The text is no longer appropriate to describe what happens in this state. Rename the state to "Ensure that all necessary stakeholders are informed". Also, don't forget to rename the transition between the relabeled state and the *End* state, too. After that, the internal behavior of the *Operations Manager* subject is finished and should look similar to Fig. 6.13.

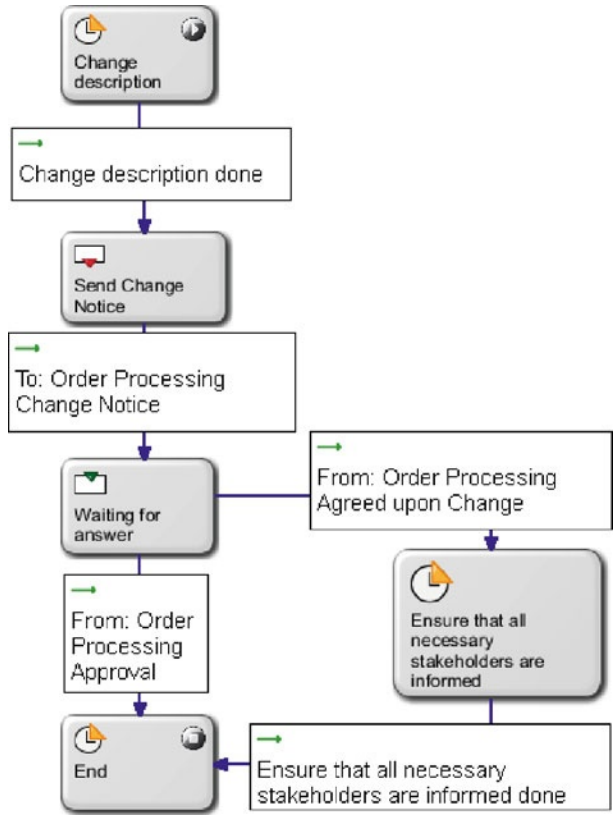


Fig. 6.13 Relabel the function state – it no longer correctly displays the behavior

So what did we do here? If you look at the whole internal behavior, you will see that there are now two possible outcomes. In both possibilities, the operations manager fills out a *Change Notice* message, sends it to the order processing department and waits for an answer. In the sunshine path, the order processing department approves and the process ends.

In the second path, there is something "wrong". The order processing department does not accept the *Change Notice* message for some reason. Then the responsible person from the order processing department uses various means of communication to resolve the issue with the operations manager (phone, personal meeting etc.). After they agree on something which is fine for both of them, the operations manager makes sure that all necessary stakeholders within the company are involved. This could for example mean, that he or she would have to inform the logistics department again, because another change has to be made. In that case, the process would start again with the process we modeled before. After the operations manager had made sure that all stakeholders from within the company are informed, the process ends. The operations manager does not need to inform any stakeholders outside of the company (for example the customer) because this is not typically done by someone in this role.

Now navigate back to the process overview, where the different subjects are displayed. The next step is to alter the behavior of the *Order Processing* subject.

Use the already known approach of a double-click to open the internal behavior of the subject.

www.dbooks.org

You will need to alter several states, by using the same approach described when altering the *Operations Manager* subject.

Hint: right-click the state or transition, select *Properties* and, if necessary, click the *General* tab.

Change the first three states and their transitions in the following way:

- The state *Receive change request* is changed to *"Receive Change Notice"* and is no longer an end state. Remove the check in the *Is end state* check box below the *Name* text box.
- The following transition *Sending Subject* is changed to *Operations Manager* and the *Messagetype* is now *Change Notice (global)*.
- The function state *Change request acceptable?* is changed to *"Change Notice acceptable?"* and is no longer an end state.
- The transition *Change request acceptable* is changed to *"Change Notice acceptable"*.
- The state *Agree on change request* is changed to *"Agree on Change Notice"* and is no longer an end state.

Now remove the *Is end state* from all remaining states except the last one (the one named *End*).

After that, the internal behavior of the *Order Processing* subject should look similar to Fig. 6.14.

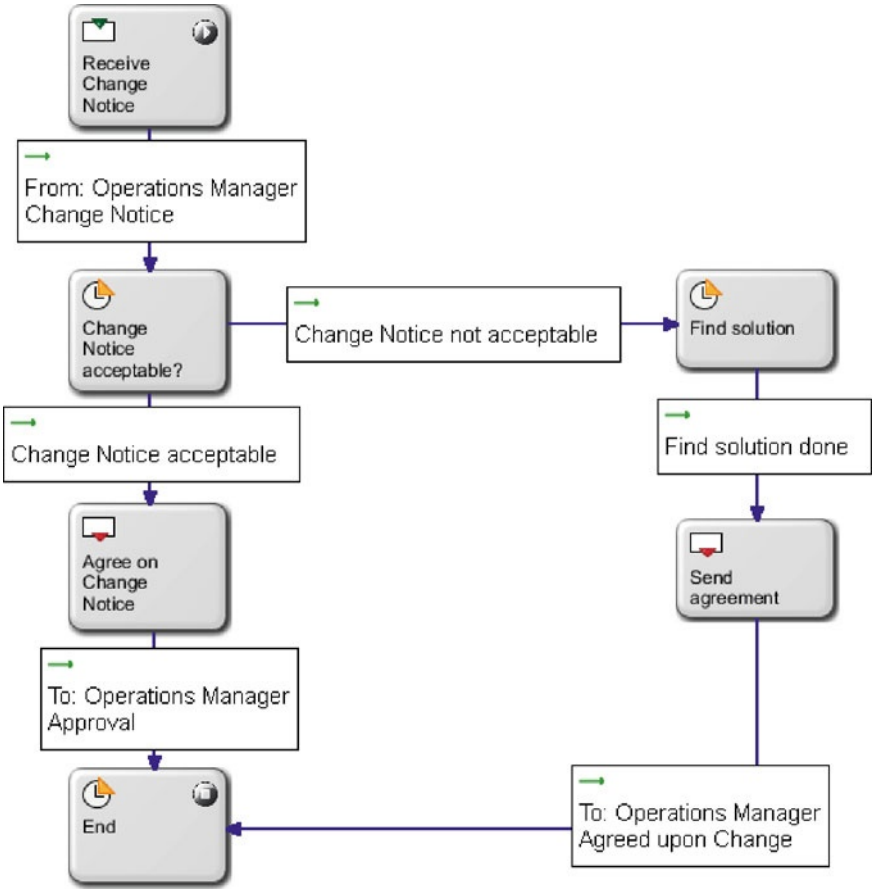


Fig. 6.14 The altered internal behavior of the *Order Processing* subject should look similar to this

In the next step, adjust the parameters of the *Change notice acceptable?* state to fit the *Change Notice* message. Make sure that the *Parameters* tab of the *Change Notice acceptable* properties looks like that shown in Fig. 6.15.



Fig. 6.15 Don't forget to add the four parameters as *Parameters for display only* – or you're gonna have a bad time when using *Metasonic Flow* later on

Then change the parameters of the *Agree to Change Notice* state so that they look like Fig. 6.16.

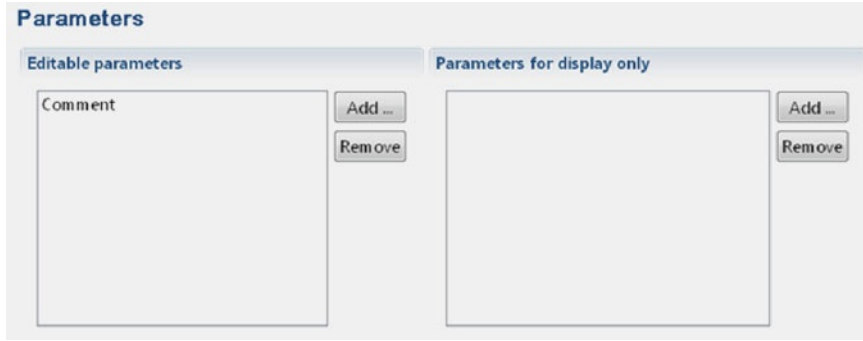


Fig. 6.16 In this case, *Comment* is the sole editable parameter

To finish the path from the *Change request acceptable?* state to the *Find solution* state, rename the transition from *Change request not acceptable* to "Change Notice not acceptable". No further changes are needed in this path.

Now, back to the sunshine path. Here you have to add a few states before the *End* state. After having agreed on a change request, in our scenario the *Order Processing* subject has to adjust the order in the SAP system, do an MPS/MRP re-run with the changed values and probably also inform the customer, to which the order belongs, that something has changed.

This means that several new states have to be created.

First, create a new function state called "Open SAP and change order" and link it to another newly created function state called "Re-run MRP". Label the transition "Opened SAP and changed order". Then make sure that the transition after the *Send agreement* state is connected to the *Open SAP and change order* state rather than the *End* state (Fig. 6.17).

www.dbooks.org

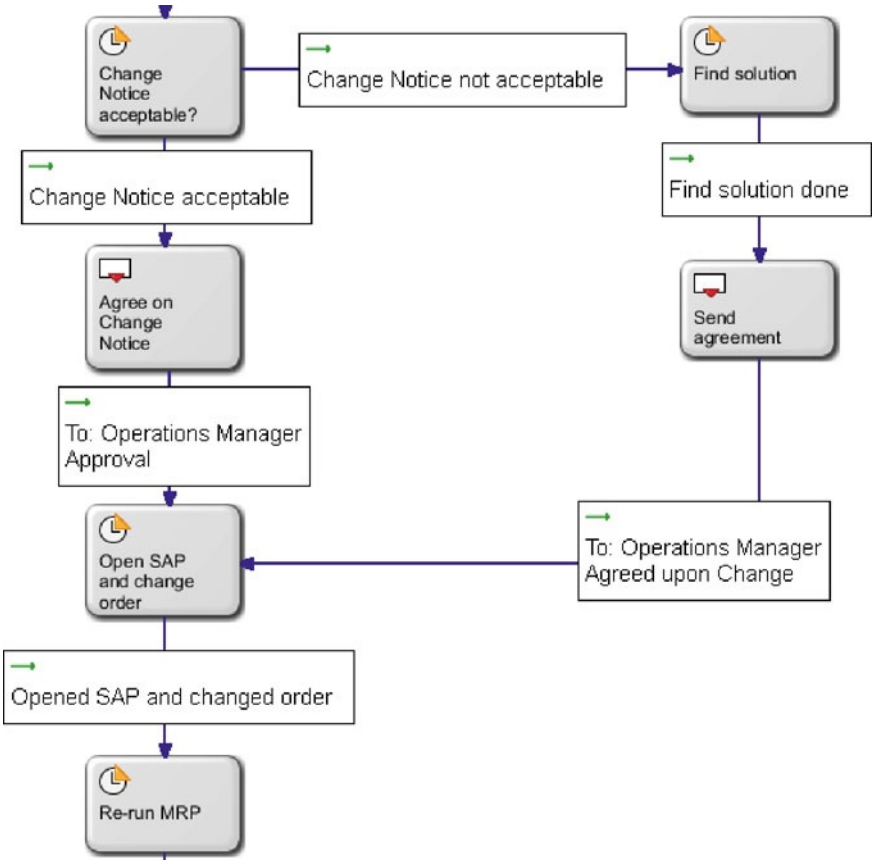


Fig. 6.17 Connect the *Send agreement* function state to the *Open SAP and change order* state

Then there is another fork: the *Order Processing* subject decides whether to inform the customer or not. It may be the case that, even though the schedules are switched, there is no delay in the customer's orders. In that case the customer does not have to be informed. However, it may also be the case that a minor change to the production schedule causes the customer's order to be delayed. In this case the order processing department informs the customer who then negotiates appropriate compensation with another department.

As a first step, create a function state labeled "Inform customer?". In the first step, as usual, model the function state: connect the *Inform customer?* state to the *End* state with the transition "Do not inform customer". This is the desired outcome – orders are switched, nothing changes, nobody has to be informed.

Then click the *Inform customer?* state and create another send state. The new send state as well as the transition are called "Inform customer".

The *Inform customer* state also needs the necessary parameters, which are shown in Fig. 6.18.

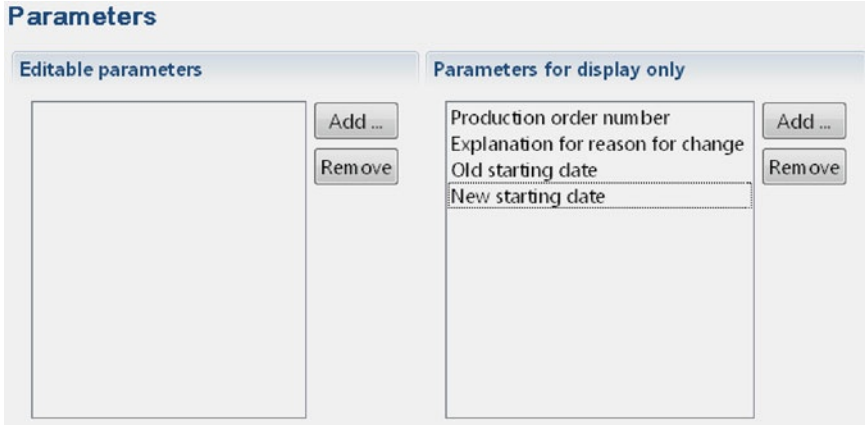


Fig. 6.18 The parameters of the *Inform customer* state should include four parameters for display only

To finish the internal behavior of the *Order Processing* subject, click the *Inform customer* state and connect it to the *End* state. This should make the *Inform customer?* decision path look similar to the one in Fig. 6.19. In the following transition, select *Customer* as the *Receiving Subject* and *Change Notice (global)* as the *Messagetype*.

The *(global)* indicates that the *Change Notice* message is also visible to other processes.

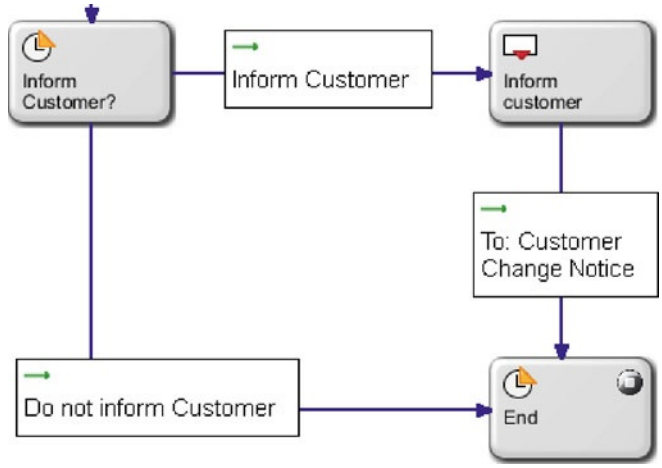


Fig. 6.19 This is a good example of a decision fork in S-BPM: the subject actively decides either to inform the customer or not to inform the customer

Afterwards, the complete internal behavior of the *Order Processing* subject should look similar to Fig. 6.20.

www.dbooks.org

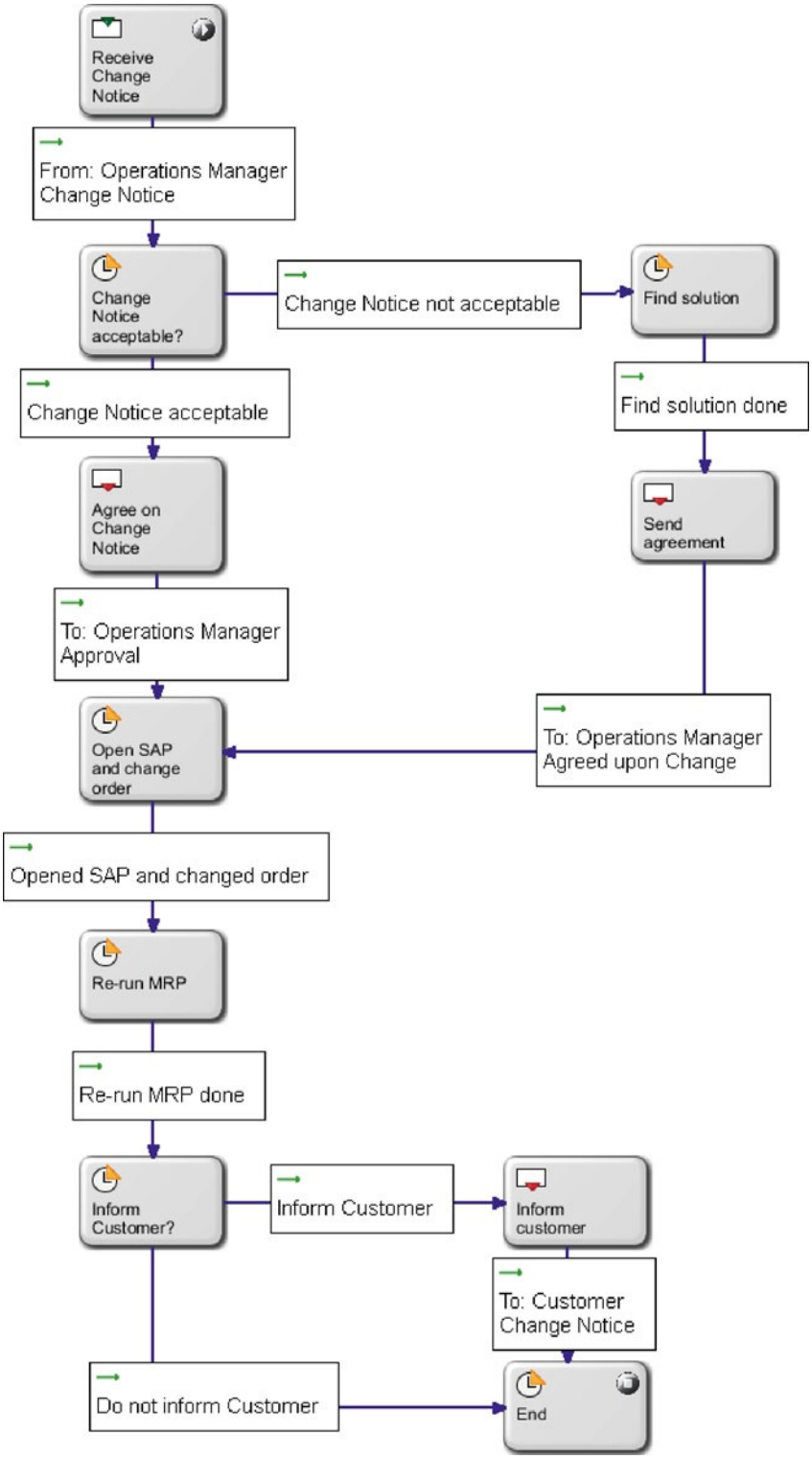


Fig. 6.20 Here you can see the complete internal behavior of the *Order Processing* subject. Yours should look similar. If not, make sure it does

6.2.4 Executing the Process

Before we can execute the process, the two roles that are still missing in the *Usermanager* (namely "Order Processing" and "Customer") have to be created. Even though the *Customer* is an external subject and can not login, this subject still needs to have a role assigned for *Metasonic Flow* to function properly.

First, start the validation environment by choosing *File/Start Validation Environment* from the menu, exactly like you did in the previous scenario. After that, click the *Usermanager* from the familiar *Metasonic Suite – Choose an application* window.

Then create a new group called "Order Processing" in the *Group administration* tab (Fig. 6.21). If you don't remember how you did it in the previous scenario, just look it up. The steps are analogous.



Fig. 6.21 Create a new group in the *Usermanager* for the *Order Processing* subject to function properly

Create a role with the same name associated to the group in the *Role administration* tab (Fig. 6.22). Here, again, the steps are analogous to the previous scenario.

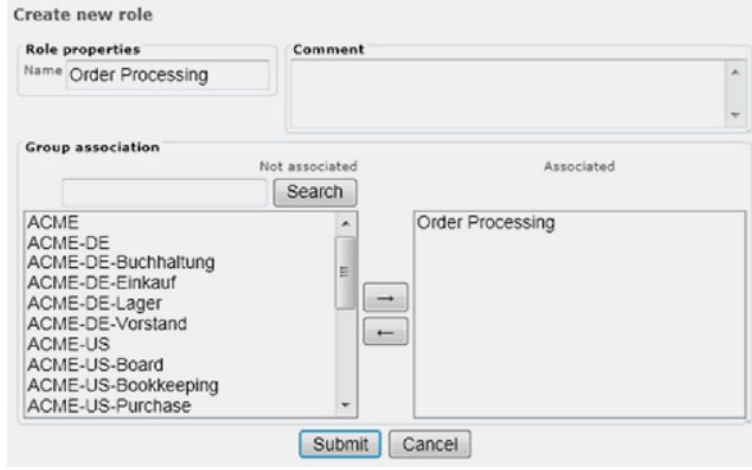


Fig. 6.22 Create the corresponding role to link the subject with the group

Create a new user in the *User administration* tab. Name him "Peter Smith" with a login name of "psmith" and a password of "topsecret". The attributes can be seen in Fig. 6.23.

www.dbooks.org

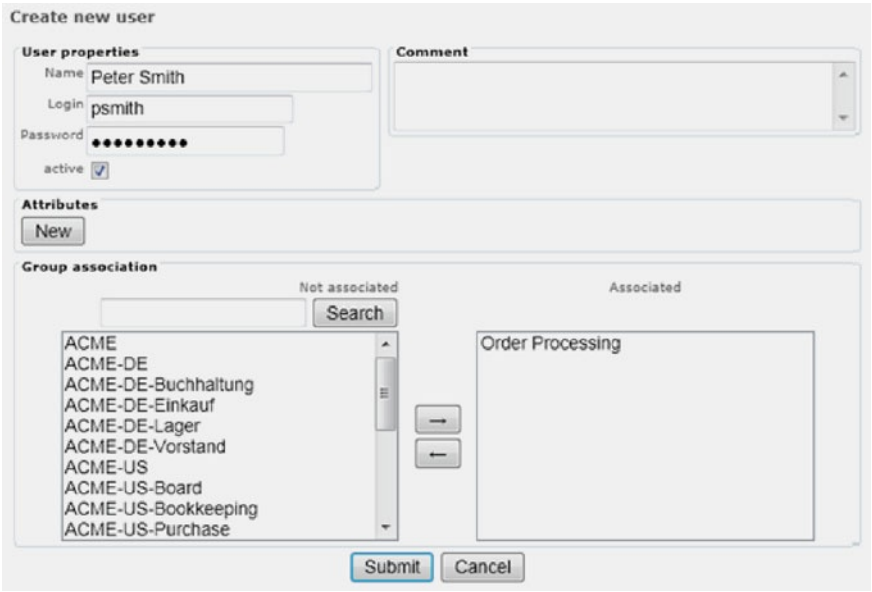


Fig. 6.23 For the process to function in *Metasonic Flow*, you need a user with valid credentials

After that, the next step is to create the role *Customer* for the external *Customer* subject to function properly in *Metasonic Flow* (Fig. 6.24). The creation of a group or user is not necessary, the role suffices.

Even though the customer is an external subject without internal behavior, a role must be created for the process to work properly.

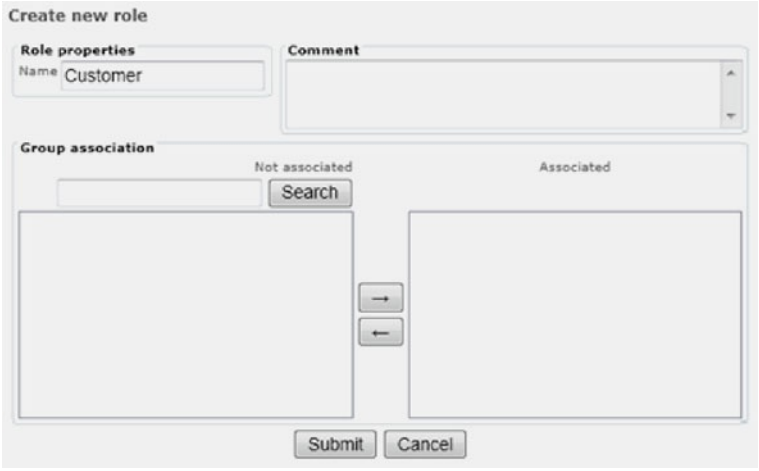


Fig. 6.24 Don't forget to create a role for our *Customer* subject. Even though the subject is empty and has no internal behavior, a role is required. Without that role, *Flow* will refuse to execute the process

With all necessary roles, users, and groups created, the roles then have to be synced with the process.

Do so by right-clicking the process group *Teaching Factory goes S-BPM* in the *Navigator* of the *Metasonic Suite* and choose *Metasonic Group – process group specific settings/Roles* from the menu. This brings up the already familiar *Roles for process group* window.

The *Synchronize* button downloads the userdata present in the *Validation Environment* database directly to the process.

Here click the *Synchronize* button, exactly like you did in the first scenario. In the following *Roles match* dialogue, again select "Usermanager" and click the *OK* button.

The two new roles should now be visible in the list as shown in Fig. 6.25.

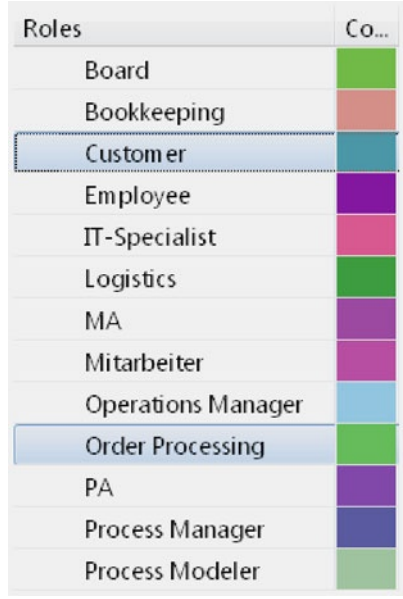


Fig. 6.25 Make sure you can find your two newly created roles in the list

Either look up the detailed steps in the previous scenario or right click each subject in the subject overview, select *Properties* from the context menu and choose the correct roles.

Save the synchronized roles either by selecting *File/Save* from the menu or with the shortcut CTRL + S. You can then close the *Roles for process group* window.

Afterwards, assign the two roles to the subjects *Order Processing* and *Customer*, respectively.

Finally, the process is ready to be uploaded and executed.

Do so by starting the *Modelmanager* from the overview window of the *Metasonic Suite* validation environment.

In the *Modelmanager* click the *Teaching Factory goes S-BPM* folder created in the previous scenario. Within this folder create a new folder called "Scenario 2" (Fig. 6.26).

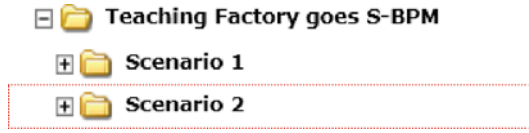


Fig. 6.26 The *Modelmanager* automatically places the new folder for Scenario 2 below the one created in the previous chapter

If you need more detailed steps, please go back – you can find them in the previous scenario.

Afterwards, the program should direct you automatically into the folder you just created. If not, click on *Scenario 2*. In the *Scenario 2* window, upload the file "Scenario_2_-_Operations_Manager_and_Order_Processing.jpp" from the *src* folder. The *src* folder can be found within the project folder (Fig. 6.27).

www.dbooks.org

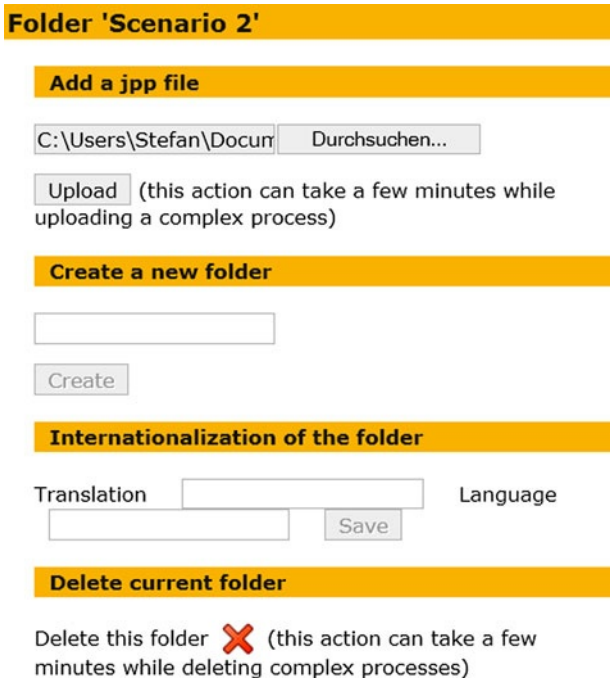


Fig. 6.27 Upload the .jpp from your hard drive. Typically, the .jpp is placed within the *src* folder of the project

After the upload is finished, activate the check box next to *Metasonic Flow* and the process is ready for execution (Fig. 6.28).

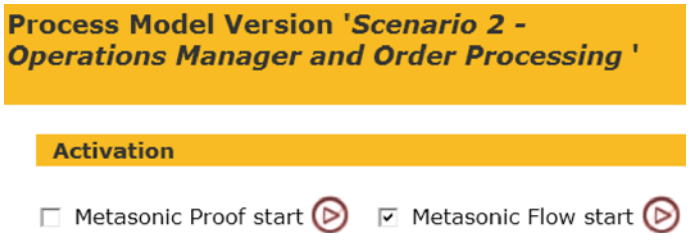


Fig. 6.28 Execute! If you can see an error message, please check if you correctly created the role for the *Customer* subject and linked them

Start the execution by clicking the *Play* button next to the text *Metasonic Flow start*.

The next step is to simulate the sunshine path of this process.

Login as an operations manager, for example our well-known John Doe, username jdoe. Then select *Task/New Task* from the menu. Here, the only process displayed within the *Scenario 2* folder is selected with a click. Adjust the title to "Scenario 2 – Sunshine Path" and start the process with the *Start* button (Fig. 6.29).

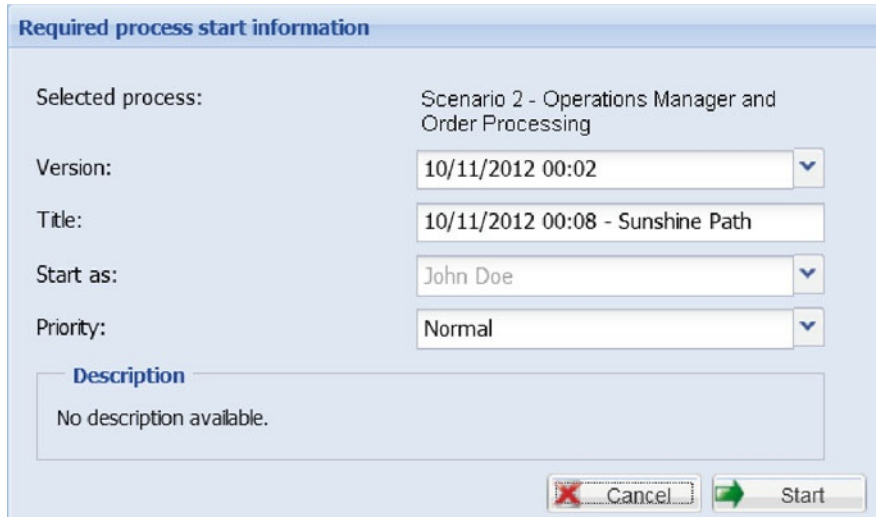


Fig. 6.29 Consider giving your process a meaningful name

After having started the process, you will find yourself in the *Start state*. Define the necessary parameters for the *Change Notice* in the *Parameters* tab. Figure 6.30 shows an example set of the parameters.

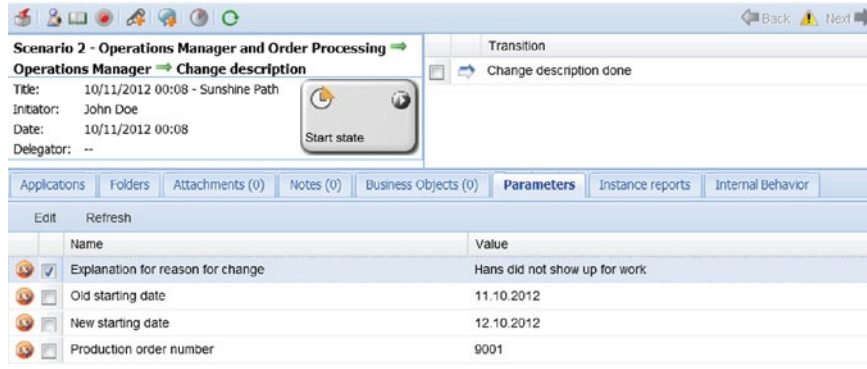


Fig. 6.30 Meaningful parameters can help people understand the process. Of course you could also fill in the parameters with rubbish or numbers

After the parameters are assigned, activate the *Change description done* check box and click the *Next* button.

Here, choose the recipient (the order processing department) after clicking the *To* button. Save the selection by clicking the *Save* button, and send the message with the *Send* button.

After that, it is necessary to login as the previously created user Peter Smith. So John Doe is logged out by clicking *logout*, and Peter Smith is logged in with the previously created credentials.

After logging in, select the only available process from the *Active Tasks*. Receive the inbound message by checking the checkbox and clicking the *Receive* button. You can read the parameters in the *Parameters* tab, exactly like in the previous process.

www.dbooks.org

In the sunshine path it is assumed that the *Change Notice* message is acceptable, therefore check the *Change Notice acceptable* check box. In the following browser window, you may optionally edit the *Comment* parameter in the *Parameters* tab. Select *John Doe* as the recipient by clicking the *To* button and send the message back to him. The next step is to change the order dates in SAP.

Here it is assumed that the user, in this case Peter Smith, opens SAP, navigates to the desired order and changes the order attributes according to the *Change Notice*.

Upon finishing, tick the check box and let the process proceed to the next step by clicking the *Next* button. After having changed an order, the overall MPS/MRP schedule typically does not comply anymore. Therefore, it would be necessary for Peter to re-run the MPS/MRP in SAP. So here it is assumed that Peter navigates to the MPS/MRP dialog in SAP and starts a re-run. When this is finished, tick the checkbox and proceed to the next step.

Here it is necessary to determine whether the customer needs to be informed about the changes. Then it is assumed that Peter evaluates the new results of the MRP run and then decides whether the customer must be notified or not.

Again it is assumed that switching the two orders produces no delay and the customer also does not have to be informed. Therefore, activate the *Do not inform customer* check box and the process terminates for the *Order Processing* subject (Fig. 6.31).

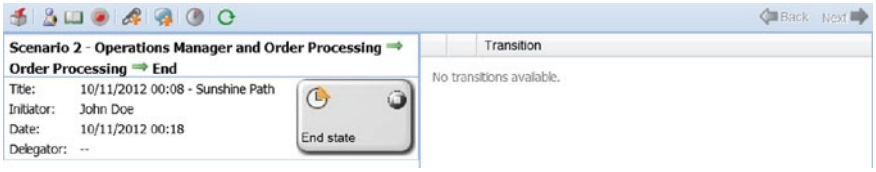


Fig. 6.31 If you made it this far, we have good news for you

Logout as Peter Smith and login as John Doe again to accept the *Approval* message.

Choose the sole scenario available from the *Active Tasks*, receive the message by ticking the check box and choosing *Receive*.

You can read the optional comment *Parameters* tab, but nevertheless the process is now finished for all participating users.

6.3 Accomplishments

After having completed this scenario, it can also be used in combination with scenario 1 to deal with emerging collaboration and communication situations. The scenario can also be adjusted to fit individual business needs, including the users of the reader's company so that scenarios 1 and 2 could be used in practice.

The only question mark here is the communication with the external customer – this could be done either by creating a customer subject with defined internal behavior (if you know exactly how the customer reacts) or by other means of communication (email, letter, phone call …).

6.4 Lessons Learned

After having worked through this chapter, the following additional concepts of the *Metasonic Suite* should be clear:

- External Subjects as "black boxes"
- Duplicating and altering an existing process
- Global Messages

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

www.dbooks.org

Transition – Part II

"This is amazing!" Peter said, after he completed the process on his notebook. "Seriously, this is exactly what we need!"

"It's good to see that you are also convinced by the S-BPM concept," John answered.

Peter replied, "I do admit that I didn't fully grasp the concept of it. But the results speak for themselves!"

Although they tried to explain the concept to him again, the problem was he had missed the introduction the consultants gave at the very beginning as well as scenario 1. "Don't worry, we will give you all the details about S-BPM later," John told him.

"Well, Bob, it's nice to see that they already have such a good understanding of S-BPM," Al said.

"That's true," Bob replied, "But you know, there is always room for improvement."

"Improvement?" Norma asked. "What improvements?"

"Before we speak about improvements, I would like to talk a bit about your expectations," Al stated. "You brought us here because of a problem that arose when something occurred that was not covered by any of your processes. Together we have now modeled two processes which can help you in this matter. Does this solve your problem? Did we meet your expectations?"

John was the first to answer. "For myself I can say that you exceeded all of my expectations. Seriously, we are here, modeling on the first day, and already have two working solutions!"

Norma also took the chance to answer: "I am also impressed. I came here with no expectations, therefore it wasn't hard for you to fulfill them – but I really came to like the S-BPM concept."

Peter also answered: "As I already said, I didn't fully understand how it works. But I like the outcome."

"I also like what we were able to accomplish in such a short period of time," John added.

"Yes, we took a bit longer than expected – and we have reached the end of the working day," Bob said. "I would suggest that we leave it for today, and tomorrow we can meet again for to make some improvements."

"Improvements?" John asked. "But we created two perfectly working processes! What do you want to improve?"

"Oh, just for a start … As you already said: you have TWO processes. How about we make it one?"

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_7, © The Author(s) 2013

The Problem – Part III

It's not that I'm so smart, it's just that I stay with problems longer. *Albert Einstein*

The next morning, the small party assembled again in the seminar room. Even Peter was there again. John said, "Nice to see you again, Pete!" "How come you are still with us?" Peter answered: "Well, after witnessing that S-BPM stuff yesterday, I really want to see the outcome of this project."

Then Al took the floor: "So, now that we are all together again, I think we can start with our second day. I must say that I really like what we have accomplished so far. But there is still room for improvement."

"Yes," Bob continued. "As I already told you yesterday … Now we have two processes. But isn't that a little inconvenient?"

Process 1: Operations Manager and Logistics.
Process 2: Operations Manager and Order Processing.

"Inconvenient?" John asked. "Why should that be inconvenient? We have two perfectly fine working processes."

"Let's play this through, John. Just assume that now there is a problem in the factory and you have to use the processes. What would you do?" Bob replied.

"Okay. So, let's assume I already know which orders to switch, right? So then I take my notebook, log into *Metasonic Flow* and start a new task. There I notify Norma about the problem. After the process is finished and we agreed on something, I notify Pete," John said.

"But isn't that quite inconvenient?" Al asked. "You have to start two different processes because of one task you want to accomplish. Therefore, you always need twice the number of processes than there are problems."

"I see your point," John replied. "You want to tell me that even though we now have a working solution, it is not yet a beautiful one."

Al smiled. "Sort of" he said. "Let's merge the two existing processes into one."

"And this works? Just like that?" Norma raised her voice. "Of course it does," Bob answered. "This is S-BPM, only limited by a few rules and your imagination. But before we get into modeling, there is one thing which is not clear to me."

"And what is that?" John asked. "Well," Bob replied, "since the very beginning you stated that you always notify the logistics department first. Why is that?"

John took the floor: "Well … I think I am just used to doing it that way. This is how we always solved those kinds of problem. I would tell the logistics department, and then the order processing department. This is how it always was."

"But, there is no real reason to do that?" Bob asked.

"No, not as far as I know." John replied.

"So that means, it could be that other colleagues from your department do it differently? It could be that they notify the order processing department first?" Bob said.

John thought for a second. "Well, now that I think about it … this could actually be the case. But hey, as far as I understood it, it doesn't matter. Instead of starting process 1 first, they just start process 2 – so everything is the other way round."

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_8, © The Author(s) 2013

www.dbooks.org

"This is how I see it." Bob paused for a moment. "It doesn't matter which department is notified first. Either way, the process must be designed so that, if necessary, both parties are notified."

"If necessary?" Norma asked. "What do you mean by that?"

"Well," Bob answered, "even though you may deny it, there could be a case where your departments, operations and logistics, decide to just swap orders without causing any big trouble. Because if you notify the order processing department, the MPS/MRP has to be re-run, the order has to be changed in SAP and stuff like that. Probably just because of a tiny switch in orders. Therefore, one can envisage a case where both of you agree on a change, but don't tell the other department because it just does not matter for the overall output."

"We have seen that in other companies, too," Al added with a nod.

"So, basically what you are saying is, that there must also be the possibility not to notify the other department?" Norma asked.

"Yes, this pretty much sums it up," Bob concluded.

"Okay, now let me sum this up," John said. "Let's try to look at the big picture of the whole process we are now trying to build. So at first there is a problem. Then I log into *Metasonic Flow* and there is only one new process to start, which covers all subjects. Right?"

"Go on," Bob encouraged him.

"So before I can do anything, the process asks me whom to notify first, because it doesn't matter who I choose at that point." Everybody nodded in approval. "So I choose for example to notify the logistics department first. Here everything is like in process 1: first I fill out a request for change, send it and finally, one way or another, the process ends. But before the internal behavior reaches the end state, it has to ask me whether I want to notify the other party."

"… or whether you want to end the process," Norma added.

"Yes, thank you, Norma. Almost forgot about that. But then – what? If I choose to notify the other party, the process must begin anew, but different …" John struggled for words. "Seriously, how exactly do you do that in S-BPM?"

"Don't worry," Al smiled, "we will show you. That's what we're here for."

"But it really sounds like there is still a lot to do. We want to accomplish everything we did yesterday, only in one process … doesn't sound easy," John responded. "But let me guess," he added "There is an easy way to do this?"

"Well, nothing is ever really easy, John." Bob said. "But this time you are right: there is an easy way to do this. Which we will show you now."

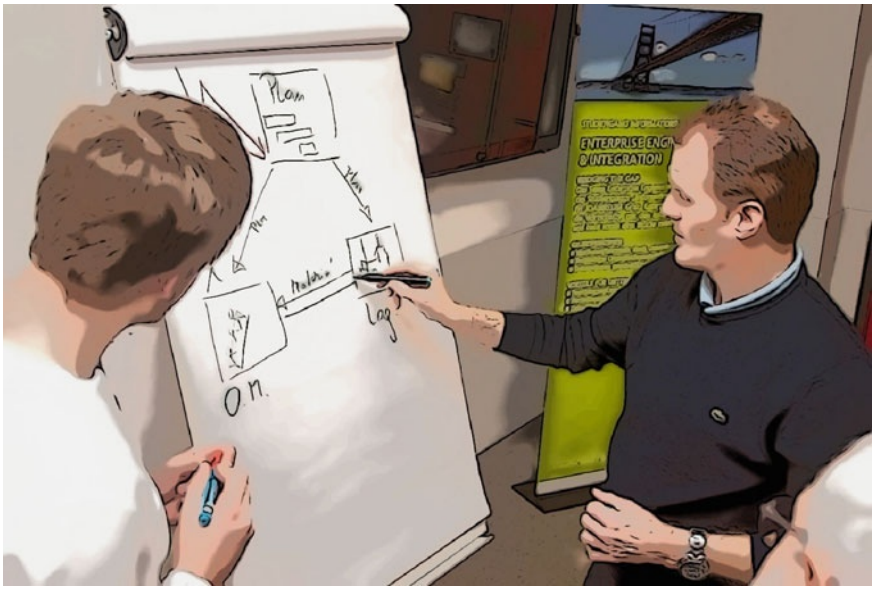


Fig. 8.1 John finally starts drawing himself, supporting Bob



Fig. 8.2 AI explains other features to the team

www.dbooks.org



Fig. 8.3 Peter, John, and Norma model the third process on their own

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

The Solution – Part III

Genius is one percent inspiration, ninety-nine percent perspiration. *Thomas Edison*

9.1 Summary of the Problem

This scenario amalgamates the various elements of the previous scenarios to create one integrated process. In this process, all necessary stakeholders are involved and able to communicate with each other. There is not much new internal behavior in there – the challenge is to merge two different scenarios into one. This will involve some copying and pasting from the previous models, including a remodeling of the operations manager, who now has to perform both internal behaviors.

Another challenge is that it should not matter whether the operations manager informs the logistics department or the order processing department first. The process should work either way and the decision whom to inform first should be up to the operations manager.

The result of this chapter could be integrated into a company's infrastructure and used in a production environment.

9.2 Solution (Step by Step)

9.2.1 Copying the Process

Just like in the last chapter, we start by copying a process. We want to merge the functionality from scenario 1 and scenario 2. Therefore, the easiest way is to copy one of them. We will copy the process from scenario 2 as it provides more functionality, and thus reduces modeling effort.

At first make sure that the *Metasonic Suite* is up and running.

Then copy and paste scenario 2. Right-click the file *Scenario_2_-_Operations_Manager_and_Order_Processing.jpp* on the left-hand side of the *Navigator* and select *Copy* from the context menu.

Then right-click onto the folder labeled *src* and click *Paste*. Just like in the previous scenario, a dialogue box with the title *Name Conflict* appears. Here, choose the new name for the process. The new name is "Scenario_3_-_Operations_Manager_and_Logistics_and_Order_Processing.jpp", which perfectly reflects the content of the new scenario.

If a dialogue labeled *Renew identifiers* appears, agree by clicking the *Yes* button.

Your navigator window should now look similar to Fig. 9.1.

Using the shortcuts CTRL + C and CTRL + V is fine too.

Of course you can name your scenario differently if you like. The name should somehow describe what the scenario is about.

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_9, © The Author(s) 2013

www.dbooks.org

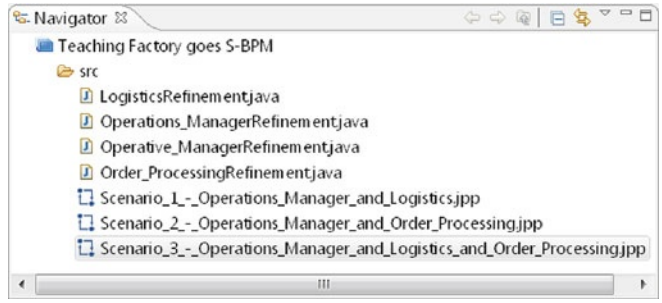


Fig. 9.1 This is what the *Navigator* part of the screen should look like after successfully duplicating scenario 2

After opening the newly created scenario, it should look exactly like scenario 2. If it does not, you did something wrong. In this case, please go back to the beginning. If you followed this tutorial exactly, your scenario 3 should look like Fig. 9.2.

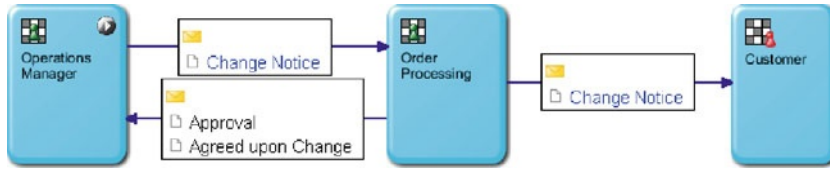


Fig. 9.2 Your scenario 3 should now look exactly like this

9.2.2 Altering the Process

The result of this chapter should be one integrated process which includes all subjects. Therefore, the *Logistics* subject is also needed here. Simply open the file *Scenario_1_-_Operations_Manager_and_Logistics.jpp* from the *Navigator* with a double-click. Here you should see one subject called *Logistics*. Right-click it and select *Copy* from the context menu.

Now go back to scenario 3, either by navigating through the open windows or by double-clicking it in the navigator.

Right-click on a free space and select *Paste* from the context menu. Now move the mouse pointer to where you want to place the copied subject. Place it below the *Operations Manager* subject by left-clicking there. Afterwards, your screen should look similar to Fig. 9.3.

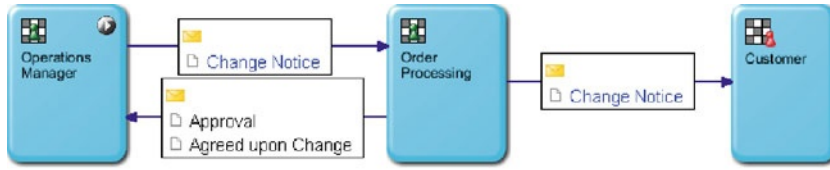




Fig. 9.3 The *Logistics* subject was copied from scenario 1

Of course you will notice that the subject is not connected to any other subject. Change this by creating the same messages as in scenario 1: the *Logistics* subject sends a *Request for Change* message to the *Operations Manager* subject and receives either an *Approval* message or an *Agreed upon Change* message.

Click the *Logistics* subject and then click the *Envelope* symbol from the context menu. Following this, click the *Operations Manager* subject.

Now the *Create new message* window should pop up. If it does not, you probably did not hit *Operations Manager*. In this window, select the *Approval* and the *Agreed upon Change* message types (Fig. 9.4) and click *OK*.

Reminder: you can select multiple messages by holding the CTRL key while clicking.

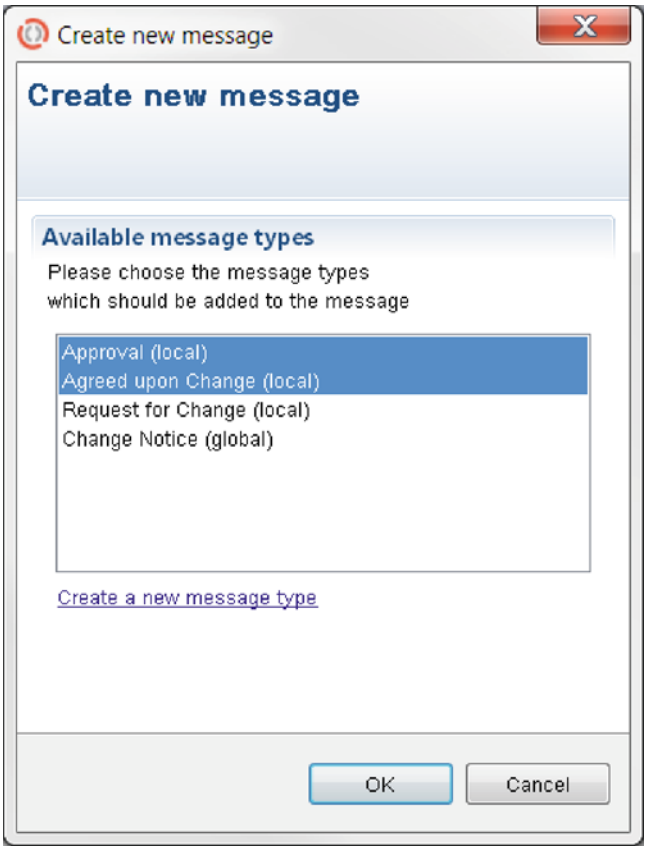


Fig. 9.4 Make sure that both the *Agreed upon Change* and the *Approval* message types are selected

You also need to define the message flow from the *Operations Manager* to the *Logistics* subject. To do so, click the *Operations Manager*, then the *Envelope* from the menu and connect it with the *Logistics* subject.

In the *Create new message* window, select the *Request for Change* message type (Fig. 9.5) and click the *OK* button.

www.dbooks.org

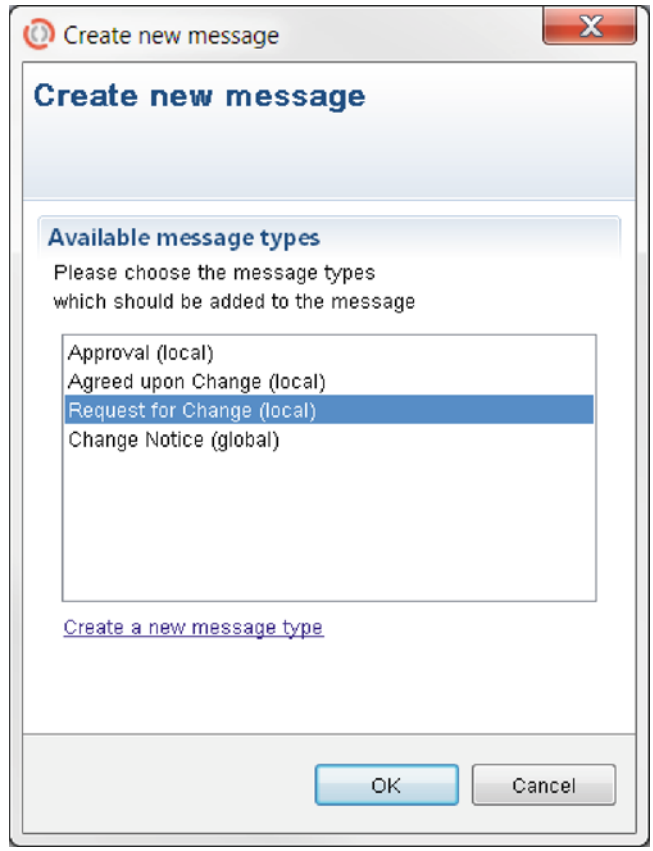


Fig. 9.5 The *Request for Change* message type needs to be selected

Your process should now look similar to Fig. 9.6.

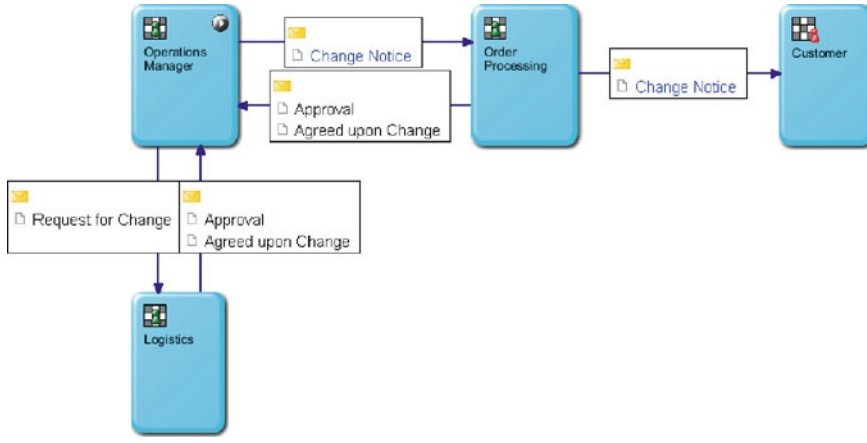


Fig. 9.6 This is what the process should eventually look like

9.2.3 Altering the Internal Behavior

Now that the top-level view of the process is finished, the internal behavior needs to be altered.

The central point of the process is the *Operations Manager* subject, which initiates the communication. At the moment, the only communication currently addressed in the internal behavior of that subject is the one with the *Order Processing* subject. So the next step is to merge the functionality of the *Operations Manager* from scenario 1 with the functionality of the *Operations Manager* from scenario 2. There are only minor changes to the internal behavior.

To begin, open the internal behavior of the *Operations Manager* subject (from scenario 3) by double-clicking it.

At the moment, everything should look exactly like it did in the previous scenario.

Copy the contents of the internal behavior of the *Operations Manager* subject from scenario 1. Open scenario 1 from the *Navigator* and double-click the *Operations Manager* subject. There, copy the whole internal behavior, for example with the shortcuts CTRL + A and CTRL + C. Afterwards, go back to the internal behavior of the *Operations Manager* from scenario 3, right-click somewhere and select *Paste* from the menu. Then you are asked where to place the copied internal behavior. The best option is to place it to the left of the currently present behavior, which results in the screen looking similar to Fig. 9.7. You will probably notice that the transitions of the receive and end states do not display any value. If this is the case then don't worry, this can happen when copying over parts of an internal behavior.

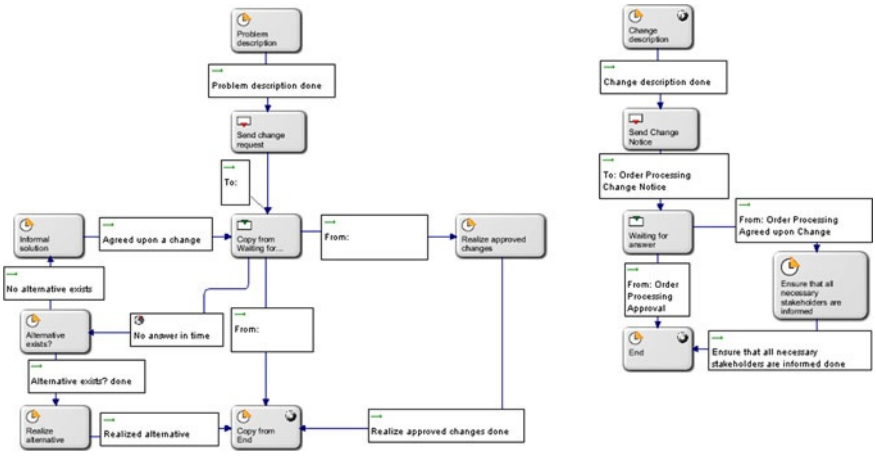


Fig. 9.7 The two internal behaviors are placed next to each other

You will also notice that one of the end states was automatically renamed to *Copy of End*, because of the fact that a state name must be unique.

Now solve the problem of which way to go in the internal behavior. As stated, it does not matter whether the *Logistics* or the *Order Processing* subject is notified first. Therefore, we need to define a state right at the beginning of the internal behavior where the *Operations Manager* can decide whom to notify.

To do so, create a new function state labeled "Who to notify?" and place it between the two internal behaviors of the *Operations Manager* subject. For the *Operations Man-*

www.dbooks.org

ager to choose whom to notify first, there need to be two transitions out of that function state: one to the *Change description* state, where the *Order Processing* subject is notified first, and one to the *Problem description* state, where the *Logistics* subject is notified first. Click the *Who to notify?* function state and choose the arrow from the context menu. First, connect the arrow to the *Change description* state and label the transition "Notify Order Processing". Repeat the step but connect the arrow to the *Problem description* state and label the transition "Notify Logistics". When you are done, the internal behavior should look similar to Fig. 9.8.

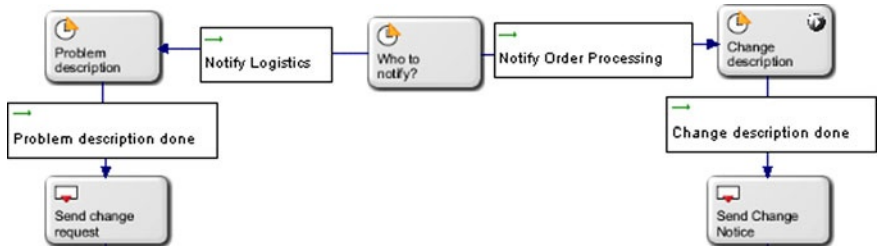


Fig. 9.8 The *Operations Manager* can now decide whom to notify first

Another way is to simply right-click the state, choose *Properties* from the context menu and click the *Set state as start state* link there.

As you can see, *Change description* is now wrongly selected as the start state of the subject. One way to change this is by right-clicking on an empty space in the internal behavior of the *Operations Manager* and then selecting *Properties* from the context menu. In the *Properties* window, click the *Configure Subject* link (Fig. 9.9).

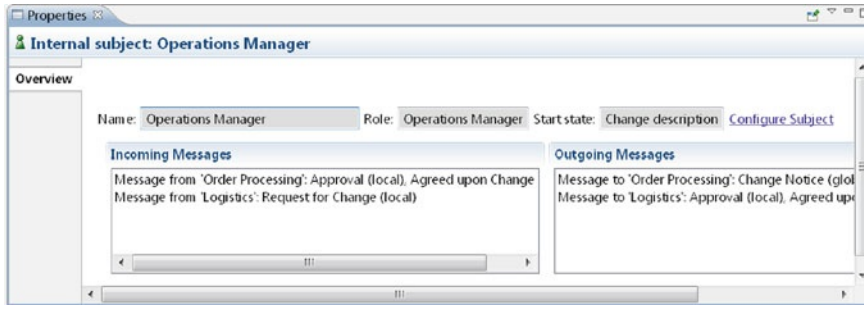


Fig. 9.9 Use the *Configure subject* link in the *Properties* window to change the start state

After clicking the link, you should end up in the *Properties* window of the *Operations Manager* subject. Here, select *Who to notify?* as the *Start state* from the drop-down menu as shown in Fig. 9.10. Afterwards, save your selection (e. g., with the shortcut CTRL + S).

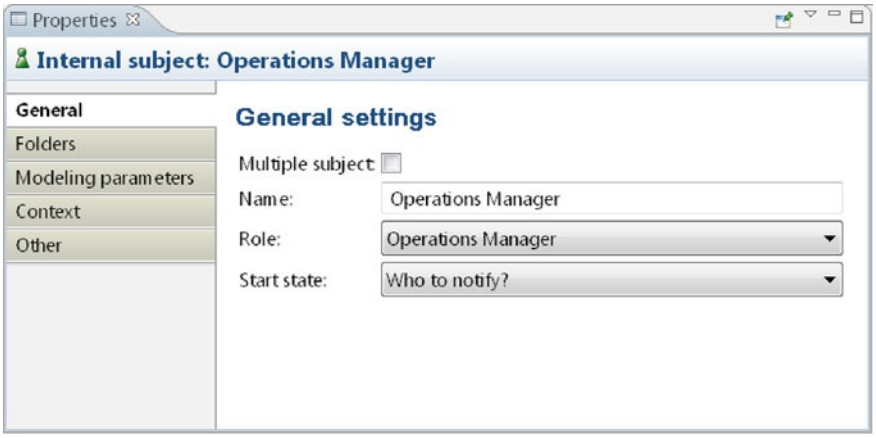


Fig. 9.10 Choose the correct start state from the drop-down menu

After setting the correct start state, return to the internal behavior of the *Operations Manager* by double-clicking the subject on your screen. As you can see, the *Who to notify?* state is now correctly marked as the start state (Fig. 9.11).

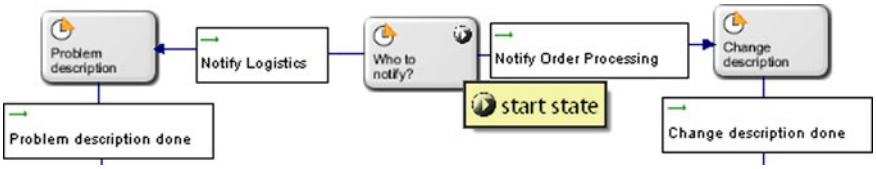


Fig. 9.11 *Who to notify?* is now the start state

Another problem we now have are the two end states. They will be eliminated when reworking the left part of the internal behavior, the one where the *Logistics* subject is notified first.

Relabel the receive state present there – when copying over the internal behavior, it was automatically relabeled to *Copy from waiting for answer*. Rename it to *Waiting for answer from Logistics*.

The next step is to "repair" the two receive and send transitions, which are possibly empty. If they are not empty, please check whether they contain the right values. To edit them, right-click on them and choose *Properties* from the context menu. In the following window, set the corresponding settings. Afterwards, the internal behavior should look like the one shown in Fig. 9.12.

www.dbooks.org

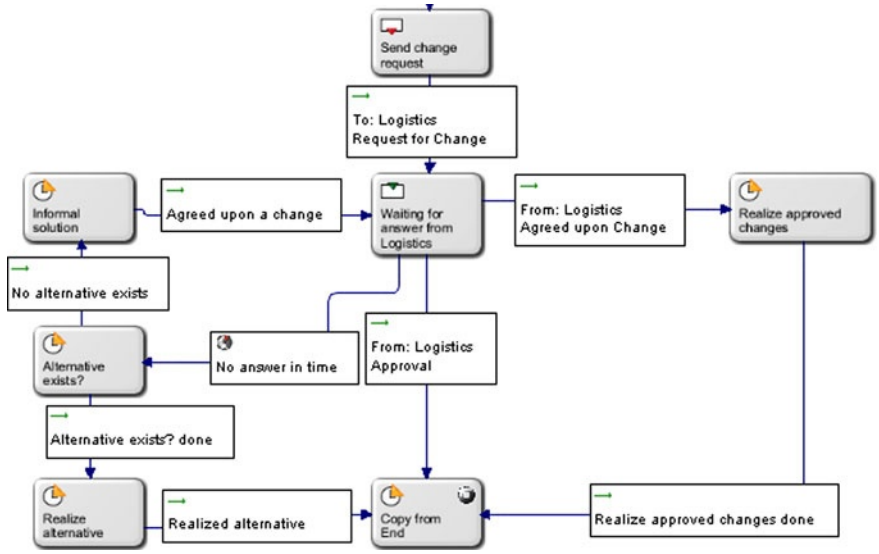


Fig. 9.12 After relabeling the receive state and correcting the transitions, the whole internal behavior looks better

The next step is to solve a problem: if the *Operations Manager* decides to notify the *Logistics* subject first, it should be possible to notify the *Order Processing* subject also, without the need to start a new process. The same action should be possible vice versa. Therefore, before reaching the *End* state, the *Operations Manager* needs to be asked whether to notify the other party as well to end the process.

To make this happen, right-click on the *Copy from End* state, select *properties* from the context menu and rename it to "Notify other party?". In addition, uncheck the *Is end state* checkbox. If this state is reached, the *Operations Manager* should have two possibilities: either notify the other party or end the process. Therefore, connect the *Notify other party?* state with the *End* state and label the transition "End process". Also connect the *Notify other party?* state to the *Who to notify?* state and label the transition "Notify other party". Afterwards, the left part, which involves the *Logistics* subject, should be finished. We recommend rearranging the states for a better overview.

An example can be seen in Fig. 9.13. The behavior involving the *Order Processing* subject is not shown here. So what you basically did now, was the insertion of another state right before the end state, which makes the *Operations Manager* able to go back to the beginning of the process. Of course the situation may arise that an inexperienced *Operations Manager* chooses to inform the *Logistics* subject twice. This will not work without starting a new process, if the *Logistics* subject had already been informed. Therefore, the user here also needs to fully understand the process, so the actions make sense.

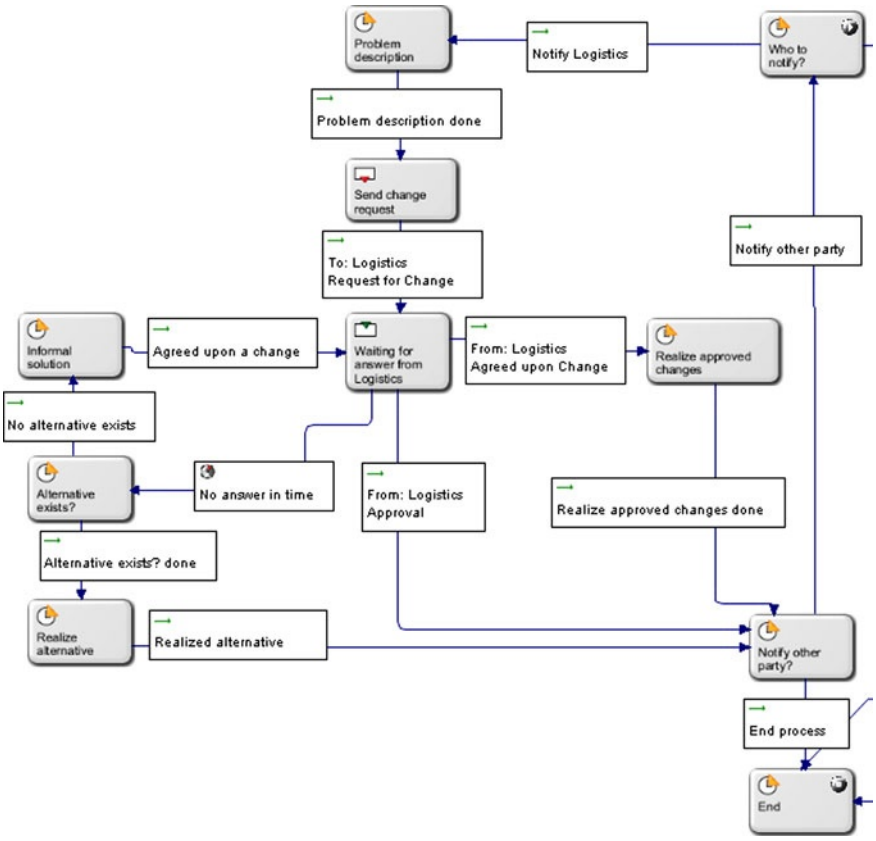


Fig. 9.13 Rearranging the states can make an internal behavior appear more clearly

The next step is to rework the remaining parts of the internal behavior, which involves the *Order Processing* subject. There is a state called *Ensure that all necessary stakeholders are informed*. This state is not needed anymore, because in this case the other party can be directly informed within the same internal behavior.

We first modify the process letting the arrows no longer point directly to the end state. So make sure that the *Waiting for answer* state is connected to the *Notify other party?* state rather than the *End* state. Simply click on the transition and move the endpoint from the *End* to the *Notify other party?* state.

Afterwards, change the endpoint of the transition between *Waiting for answer* and *Ensure that all necessary stakeholders are informed* also to the *Notify other party?* state. After that action, you can safely delete the *Ensure that all necessary stakeholders are informed* state, which will also delete its transition to the *End*. So, what now happens after the *Operations Manager* subject sends a *Change Notice* message is that both possible answers lead to the same state. But depending on the answer, the subject must decide whether to inform the other party or not. If the *Logistics* subject was notified first, then it is not necessary to notify the other party. Also, if both parties agree upon a change, the *Operations Manager* also must decide whether to inform the other party or not. It's possible that all three sides were in a telephone conference where they discussed their problem – in that case, the other party is already informed, even if it was not notified earlier.

www.dbooks.org

Figure 9.14 shows an overview of the now-finished internal behavior of the *Operations Manager* subject.

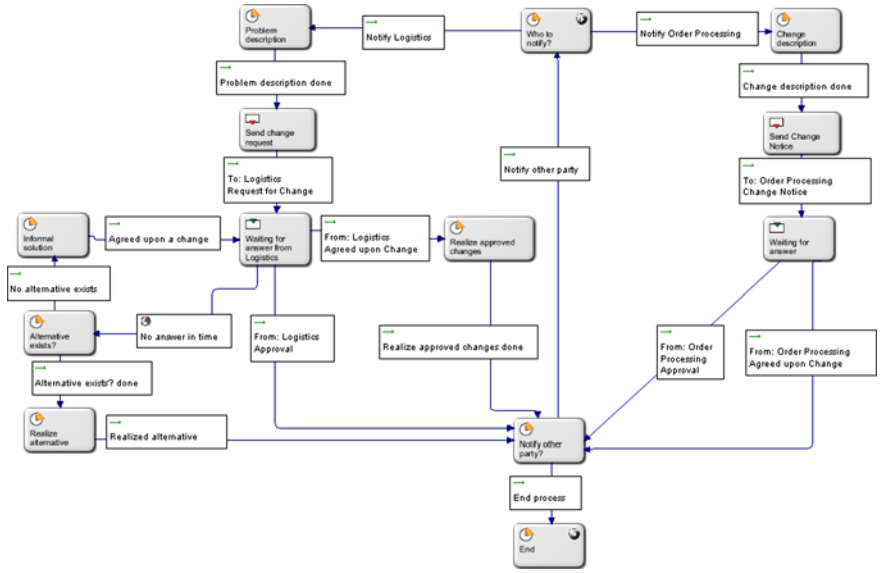


Fig. 9.14 The finished internal behavior of the *Operations Manager* subject

When copying the internal behavior from the other processes, the parameters should have remained the same; but just in case the parameters of the different states need to be examined.

The next step is to make sure that all important states have the correct editable and/or readable parameters so the process can be played through in *Metasonic Flow*.

- The *Problem description* function state should have the editable parameters *Answer requested until*, *Alternative*, *Explanation for reason for change*, *Production order number*, *Old starting date*, and *New starting date* (as shown in Fig. 9.15).

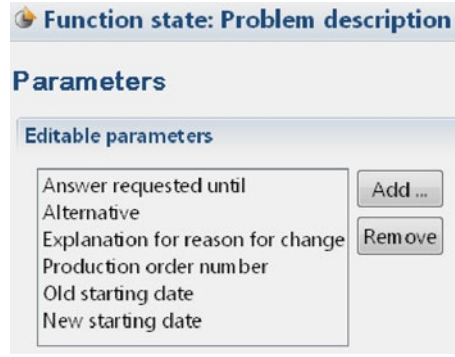


Fig. 9.15 The *Problem description* function state needs to have six editable parameters

- The next important function state, *Realize approved changes*, should have the parameters *New starting date*, *Comment*, and *Production order number* for display only (Fig. 9.16).



Fig. 9.16 The *Realize approved changes* function state needs to have three parameters for display only

- The *Change description* function state is similar to the *Problem description* function state but has fewer editable parameters: *Explanation for reason for change*, *Production order number*, *Old starting date*, and *New starting date*. See Fig. 9.17 for a screenshot.

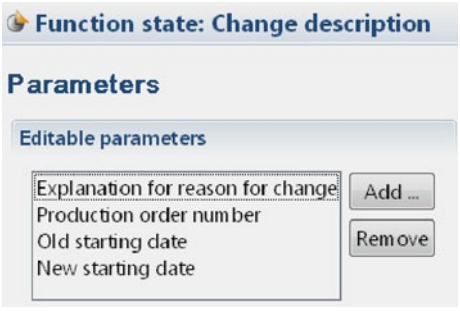


Fig. 9.17 The *Change description* function state needs to have four editable parameters

- The function state *Notify other party?* is important when it comes down to parameters. Three receive states with two different message types redirect here. Therefore, it needs the readable parameters from both message types, which are *Comment*, *Production order number*, and *New starting date* (Fig. 9.18). If you add all necessary readable and editable parameters, all three messages can successfully be read by the *Operations Manager* when in this state.

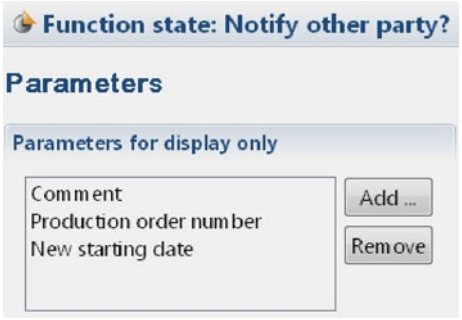


Fig. 9.18 The *Notify other party?* function state needs to have three parameters for display only

The last thing to do is to check the internal behavior of the other subjects. Sometimes copying a subject can cause problems.

www.dbooks.org

First, open the *Logistics* subject and make sure that the internal behavior looks similar to Fig. 9.19. Especially, pay attention to the transitions following send and receive states, as they are the most likely to be messed up in the copying process. Also check the parameters.

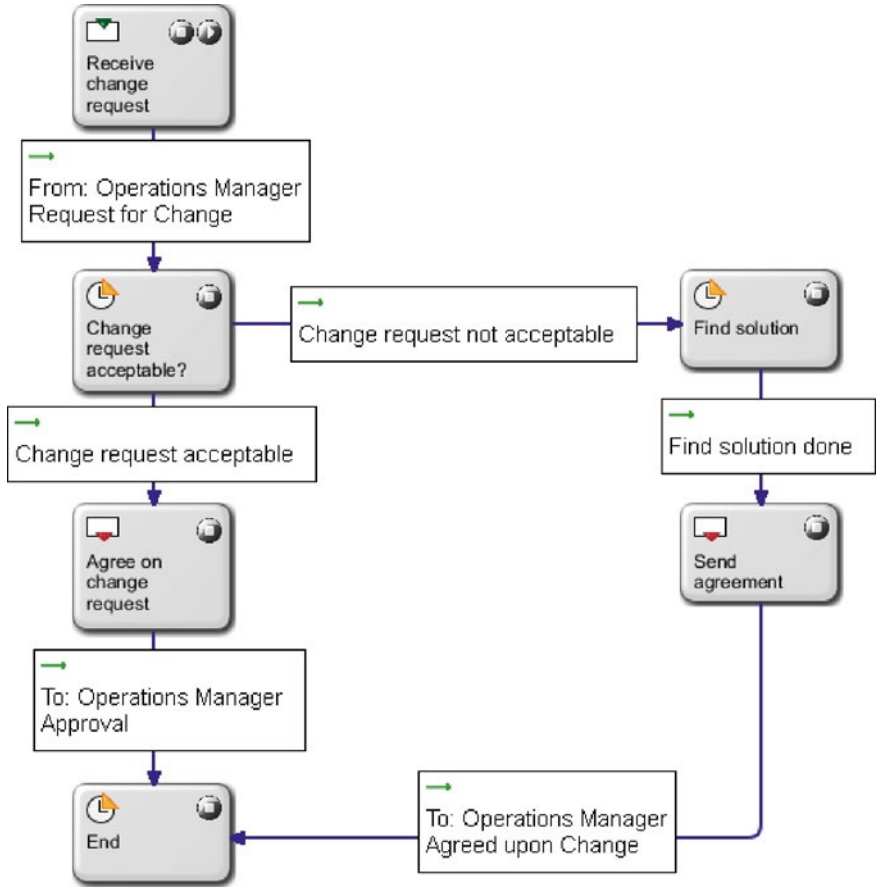


Fig. 9.19 This is what the internal behavior of the *Logistics* subject should look like

Afterwards, open the *Order Processing* subject and make sure that it resembles Fig. 9.20.

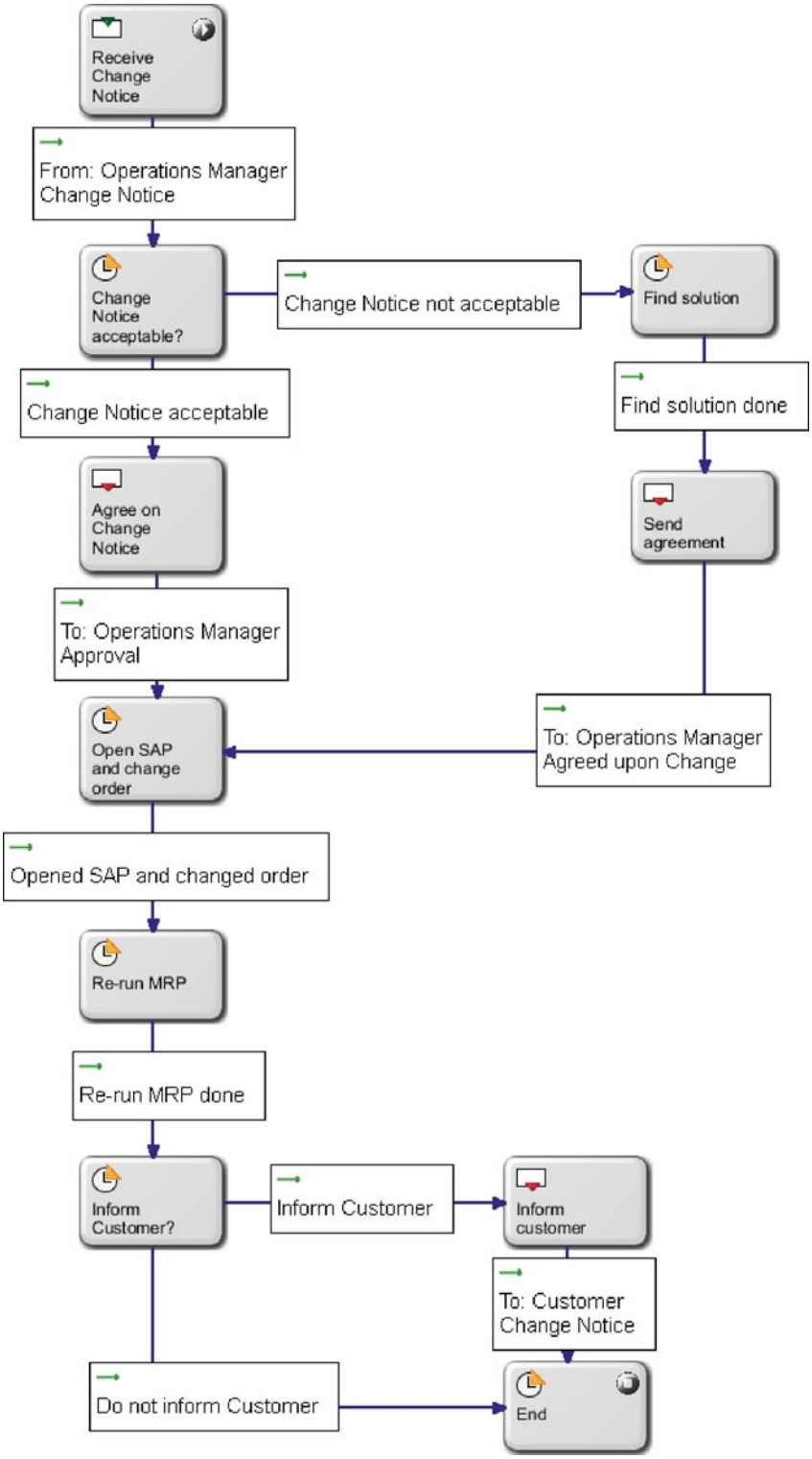


Fig. 9.20 Pay attention to the transitions after send or receive states and make sure that your *Order Processing* subject looks similar

www.dbooks.org

9.2.4 Customizing the Process

You should be familiar with the S-BPM concept by now if you have reached this point of the book. Allow yourself some time to reconsider the whole process. Of course you don't need to stick entirely to the description we provided – feel free to make some alterations to the process if you think that you can handle it. The way the processes are modeled in this book is not the only right way because there isn't exactly a right way. S-BPM is a very dynamic concept and therefore very adaptive. Nothing is written in stone here. Just for example, you could add the possibility for the *Operations Manager* to read the filled out parameters again before sending them (by adding readable parameters to the send state) or even modify them. We only provided an example for demonstration purposes.

You could also add extra states, if you like. For example, you could implement a *Read message* state after a receive state, instead of directly proceeding with the next task. You could also split some function states if you think it better fits your process. Conversely, you also could merge several states if it works better for you. You don't like the messages that are passed around? Alter them, or even redesign them completely. The possibilities for customizing S-BPM processes are only limited by your imagination. Feel free to develop your own process modeling style and stick to it[1]. Don't be afraid to choose a different approach – always model your processes in a way that supports your needs.

9.2.5 Executing the Process

Just like in the previous chapters, this process will also be executed. Again, only the happy path is played through. The first step is to start the validation environment by clicking *File/Start Validation Environment*. If you don't start it, the next step will fail.

You can not continue here without having created all users.

If you worked through the previous scenarios, it will not surprise you that this time you will not have to create any users, roles, or groups. They are already present because we just merged two scenarios. The only thing left to do is to ensure that the subjects are linked to their corresponding roles.

To avoid any problems, the roles need to be synced with the scenario again. Therefore, right-click the *Teaching Factory goes S-BPM* process group in the *Navigator* and select *Metasonic Build – process group specific settings/Roles* from the context menu (Fig. 9.21).

[1] Basic discussions about modeling can be found in the complementary book A. Fleischmann et al., Subject-Oriented Business Process Management, Springer, 2012

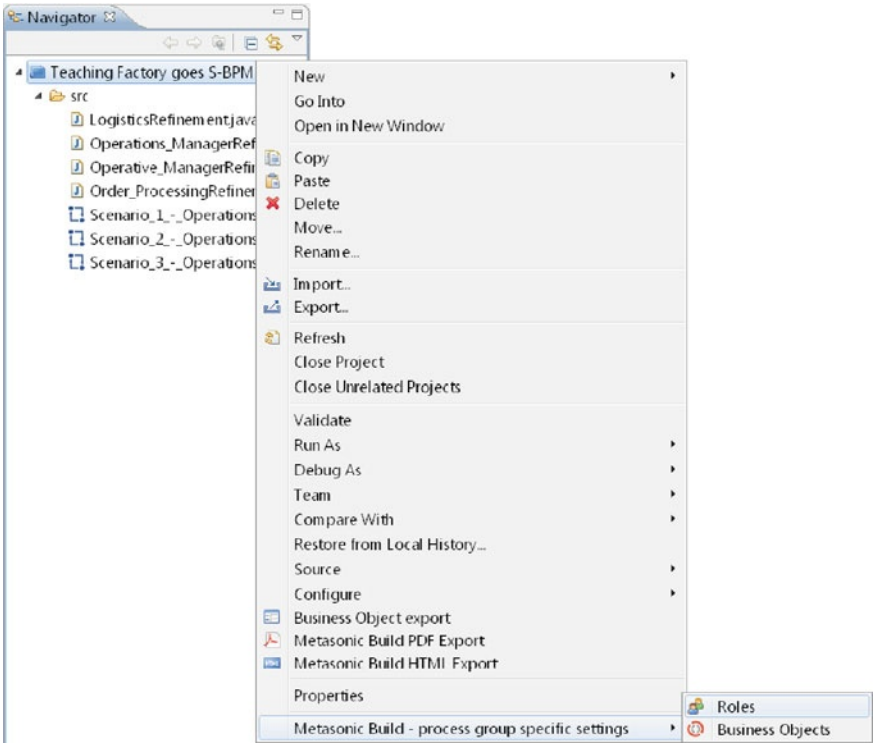


Fig. 9.21 The role-specific process settings can be found in the context menu

One more time the roles need to be synchronized with the *Usermanager*. Therefore, click the *Synchronize* button and confirm the *Roles match* dialogue by clicking the *OK* button. Then save, for example using the shortcut CTRL + S.

After synchronizing of the roles, return to the process overview of this scenario via the *Navigator*.

Make sure that each subject has the correct role assigned. First, right-click on the *Operations Manager* subject and select *Properties* from the context menu. Make sure that the *Role* is *Operations Manager* and the *Start state* is *Who to notify?* (Fig. 9.22).



Fig. 9.22 The properties of the *Operations Manager* subject should look like this

Repeat this step for the *Order Processing* subject, which should have *Order Processing* as the *Role* and *Receive Change Notice* as the *Start state* (Fig. 9.23).

www.dbooks.org

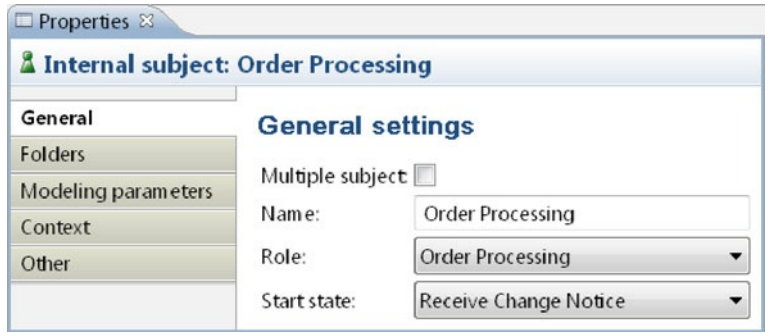


Fig. 9.23 Make sure that the *Role* and *Start state* of the *Order Processing* subject look similar to this

Finally, validate the *Logistics* subject in the same way and confirm that the *Role* is set to *Logistics* and the *Start state* is *Receive change request* (Fig. 9.24).

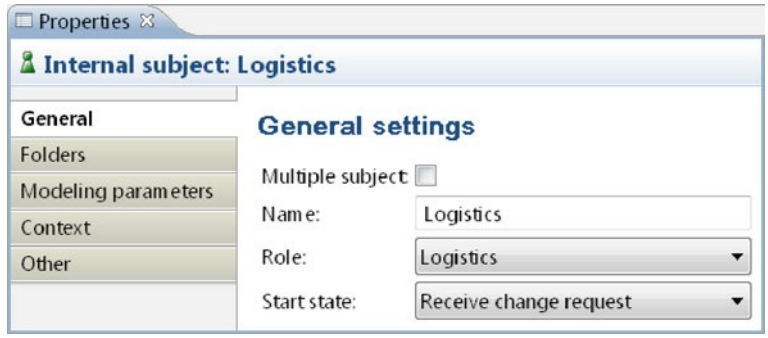


Fig. 9.24 Also the *Logistics* subject should be checked for these settings

Now upload the process in the *Modelmanager*. Open the *Metasonic Suite* window (the validation environment should already be running) and click *Modelmanager*.

Again, create a new folder by clicking the *Teaching Factory goes S-BPM* item. There, type "Scenario 3" in the text box labeled *Create a folder* and click the *Create* button.

After the folder is created, click it. Search for the .jpp file, which should be named *Scenario_3_-_Operations_Manager_and_Logistics_and_Order_Processing.jpp*, and upload it by clicking the *Upload* button. Following this, the newly uploaded process should be selected. If not, click it from the menu on the left.

The only thing left to do before execution is to activate the check box next to *Metasonic Flow start*. If your screen looks similar to Fig. 9.25, start the process by clicking the *play* button to the right of *Metasonic Flow start*.

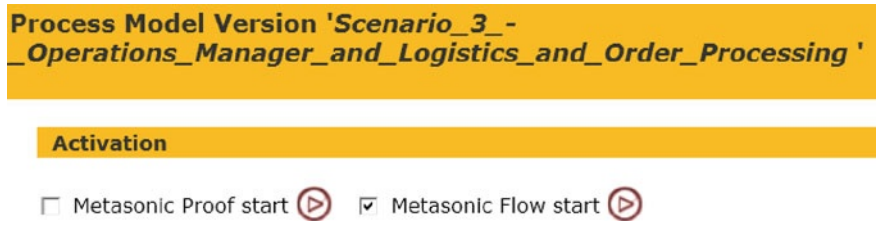


Fig. 9.25 Activate the checkbox next to *Metasonic Flow start*

As in the processes before, the goal is now to play the process through, considering the sunshine path where everything goes fine.

After the *Login* window is open, login with John Doe (username "jdoe", password "topsecret"). He will be the *Operations Manager* who starts the process. To actually do this, click *Task/New Task* at the top left of the screen. Afterwards, select the process within the *Scenario 3* folder as the process you want to start. You could give the process instance a meaningful title, like "Scenario 3 – Sunshine Path". To start the process, click the *Start* button.

Following this, the process is started, and, you should be prompted with the decision whether to notify the order processing or the logistics department (Fig. 9.26). It does not matter which one you notify first – nevertheless, click *Notify Logistics* and then the *Next* button.

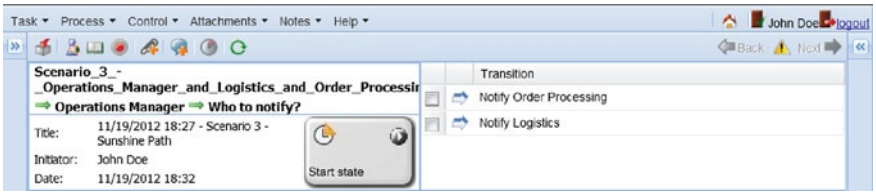


Fig. 9.26 Choose the decision *Notify Logistics*

The next process step should already be familiar from scenario 1. Here you are prompted to describe your problem. Do so by clicking the *Parameters* tab, selecting a parameter and clicking the *Edit* button. Figure 9.27 shows an example of meaningful parameters. Afterwards, click *Problem description done* and then the *Next* button to continue.



Fig. 9.27 This is an example of how useful parameters may look

In the next step, click the *To* button to choose the recipient of the message. Click the checkbox next to Norma Roe to send the message exclusively to her. Confirm by clicking the *Save* button and proceed to the next step by clicking the *Send* button.

Since John Doe is now in a receive state, log out by clicking the *logout* link on the top right of the screen. Log in again as Norma Roe (username "nroe", password "topsecret"). Double-click the sole available task in the *Active Tasks* part of the screen to open it and then double-click it again to participate in the process.

The first step for Norma is to accept the message sent to her by John. Click the inbound message to activate the checkbox and receive the message by clicking the *Receive* button (Fig. 9.28). Afterwards you could examine the received message in the *Parameters* tab. Assume that everything is fine and click *Change request acceptable* (Fig. 9.29). Proceed to the next step by clicking the *Next* button. Here you will send the message

www.dbooks.org

back to John. To do so, click the *To* button and select the single available recipient – John Doe. Click the *Save* button followed by the *Send* button to send the message. As you can see, the process is now finished for Norma. Logout as Norma and login as John again. Participate in the process by first double-clicking the only process available in the *Active Tasks* and then double clicking it again.

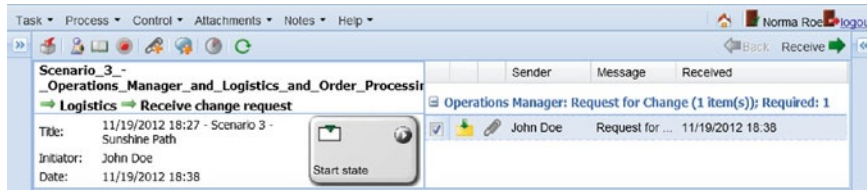


Fig. 9.28 Messages need to be actively received

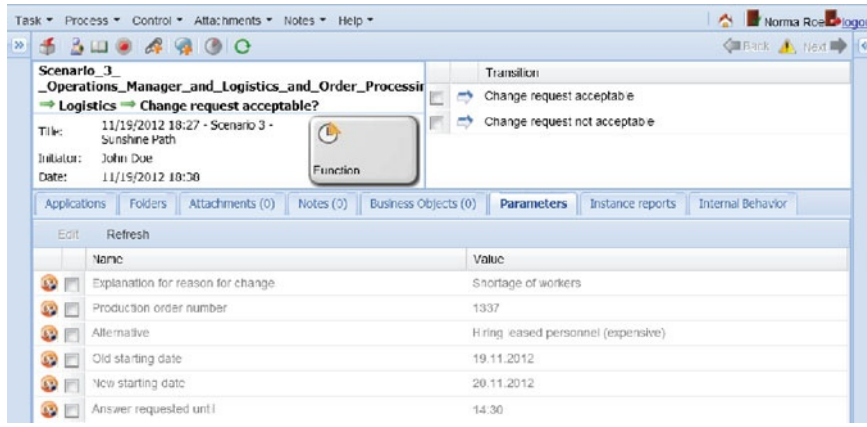


Fig. 9.29 Examine the parameters and accept the change request

Click the message from Norma to activate the checkbox and receive the message by clicking the *Receive* button. Now John is in a state where he can choose whether to end the process or to notify the other party. Note that in the *Parameters* tab the parameters can be examined. The comment, if not specified, will show as an empty parameter (Fig. 9.30).



Fig. 9.30 In this case, the *Comment* parameter is empty. Decide to notify the other party

Decide to notify the other party by clicking *Notify other party* and then the *Next* button. Now John is back in the very first state where he can decide whom to notify. Please note that even though it is theoretically possible to notify the logistics department again, the process would be stuck (i. e. a deadlock situation), because the *Logistics*

subject is already in an end state. Click *Notify Order Processing* and then the *Next* button. In the *Parameters* tab of the following step you can edit the parameters. But because you already filled them out when notifying the logistics department, this should not be necessary (Fig. 9.31). Click *Change description done* and then the *Next* button. After clicking the *To* button in the following window, select Peter Smith as the recipient and save your selection by clicking the *Save* button. Send the message by clicking the *Send* button which causes John to be in a receive state again.

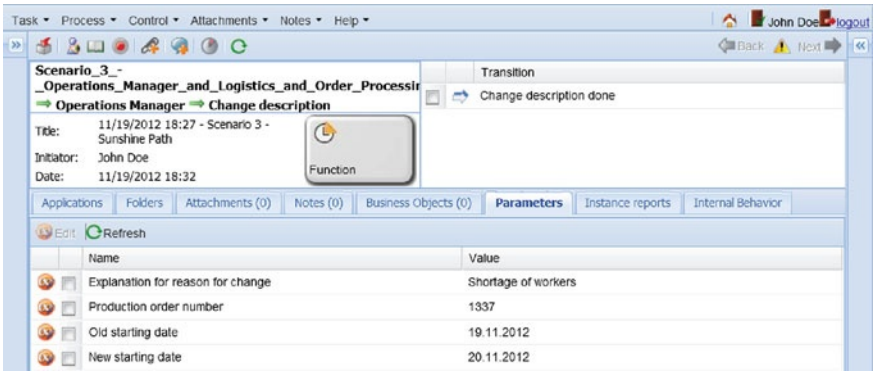


Fig. 9.31 If you already filled out a parameter, the data is still there if you send another message with the same parameters

Now logout as John and login as Peter Smith (username "psmith", password "topsecret").

Participate in the process as already described (two double-clicks). Receive the message by ticking the checkbox and clicking the *Receive* button. After receiving the message, the parameters can be examined in the *Parameters* tab (Fig. 9.32). If you want, you can specify a *Comment* parameter. Click the *Change Notice acceptable* and then the *Next* button. Send the message back to John Doe by clicking the *To* button, then *John Doe*, followed by *Save* and *Send*.

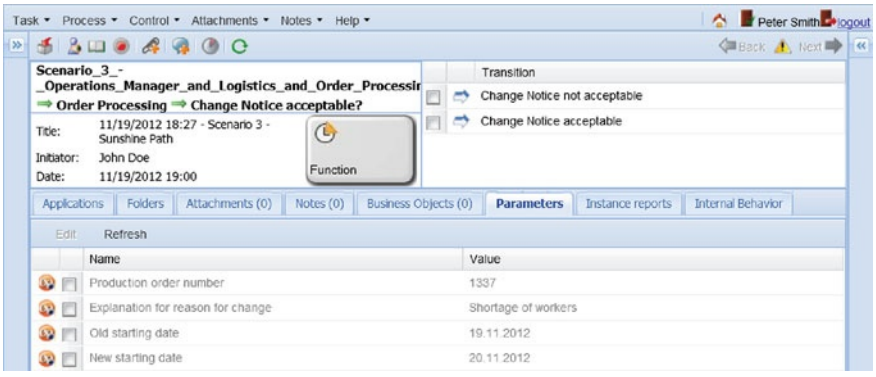


Fig. 9.32 The parameters received from the *Operations Manager* are read only

In the next step, assume that SAP is open and the order was changed. Therefore, click the transition and then the *Next* button. Afterwards, assume that a MPS/MRP re-run was done which changed nothing important concerning delivery dates. Click *Re-run MRP done* and then the *Next* button. It is thus not necessary to inform the customer because it is assumed that nothing vital has changed. Therefore, click *Do not inform customer* and then the *Next* button.

www.dbooks.org

The process now ended for Peter Smith. One last time, logout as Peter Smith and login as John Doe again and participate in the process. Receive the message, which should include an empty *Comment* parameter if none was specified (Fig. 9.33). Now click *End process*, followed by the *Next* button, which finally ends the process for John. Now the process has been played through for the sunshine path, the most comfortable path. You also could have clicked *Notify other party* instead of *End process* – but this would have caused problems, because the other two subjects have already finished their internal behavior. Therefore, John Doe would be stuck endlessly in his part of the process, because nobody would receive his messages.



Fig. 9.33 Finally, end the process by choosing the right transition

9.3 Accomplishments

After having completed this scenario, you will have as a final outcome one integrated scenario designed as a framework for structured communication, which includes the *Operations Manager*, the *Logistics*, the *Order Processing*, and the *Customer* subjects; this business process synchronizes the work of all participating parties, including the customers. We could further enhance the process to include suppliers and other internal subjects; the discussed situation is a typical end-to-end process connecting customers downstream with suppliers (value system).

Just like the other scenarios, this scenario could be further adjusted to fit individual needs. For example, more subjects could be added or the process could be totally redesigned. Even though scenarios 1 and 2 can also be used in practice, this one is different because it combines both scenarios into one process. In general, it can be considered as fit for use in a real-world environment.

9.4 Lessons Learned

After having worked through this chapter, it should be clear how to merge different processes into one, and how to overcome different problems (like missing transitions). Another lesson learned is recursiveness (a process can go back to the very beginning without restarting it) and the concept of instances: if a subject is in an end state, nothing will happen if that subject receives an already received message again because the instance is already terminated.

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

www.dbooks.org

Transition – Part III

After they finished modeling, John said: "Well, I figured that after the first two processes I couldn't be more impressed, but I guess I was wrong."

"Yes," Norma continued, "This process is amazing! I know I probably already said that about the previous processes. Maybe I am easy to impress. But hey, I could absolutely imagine using this one in production!"

Even Peter, who was kind of skeptical in the beginning, was impressed: "Well, I wouldn't have thought of something this productive as an outcome. I was merely thinking of a prototype or something, not a fully working process."

The consultants smiled. "We hear this a lot from our customers," Al said. "This is the problem with modern process management. A lot more time is needed for most methods until they can finally be executed or used in a productive environment. This is why we are presenting S-BPM and not BPMN or something else."

"I can confirm that," John answered. "We have been on the BPMN train before. You know, BPMN 2.0. Design your processes with BPMN 2.0, they said. Execute them right away, they said. But the only things we got out of it were overpaid consultants, a few BPMN processes, and nothing even close to being executable. Since that experience I always suspect the worst. This is why you have succeeded in surprising me."

"But I will talk to our IT department right away," John continued. "You really have convinced me so far. I want this process deployed on the departmental PCs. I mean … all the stress and money we can save with that third process!"

"And not only money," Norma added excitedly. "Just think of how much time we will save. You know how I just hate arguing with you. We may never argue again!"

"Thank you," John said to the consultants. "This was kind of an eye opener. I think that we should also start an S-BPM modeling team." Norma responded, "I'm in!" "But before we get to that, let us have lunch."

"Ah, yes," John said with surprise. "It's noon already. I didn't even realize how quickly time was passing …"

"Wait a minute," Al interrupted them. "Peter, you still look skeptical. What's your opinion on this?"

"You know, as I said … I am still impressed with the progress we have made. I mean, we now have a fully working process."

"But?" Bob responded.

"Well …", Peter responded, "For the other two departments it is probably an improvement. But for my department it is just an additional system to use. I mean, John tells me to change an order. Then I have to maintain everything in the *Metasonic Suite* – I have to keep track of the parameters there, while also having to work within the SAP system. That's two different systems where I have to make changes. I don't like this, because there is a high potential for errors. I have valid reasons to believe that it is really easy to screw a process up just by mistyping something … Also, I don't like the idea of opening two applications at the same time and doing everything twice."

"Now you come to mention it – I think you have a point there," Norma added. "We all have to use an ERP system and this which may lead to errors."

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_10, © The Author(s) 2013

"Well, Pete," Bob began to reply. "First, thank you for your input. And second: you are totally right. To be honest, I was kind of waiting for this issue to come up."

"Just to clarify this," Al continued, "What you're saying is the possibility to integrate the *Metasonic Suite* with other IT systems like SAP is lacking?"

"Yes," Peter answered. "I just don't like the idea of implementing an additional system, which generates more workload instead of relieving me of some."

Bob smiled. "I don't mean to surprise you, but this is also something we hear from our customers a lot," he said. "What would your dream solution be?"

"Well …", Peter answered. "My dream solution would be that the *Metasonic Suite* knows which order to change. I mean, John would provide me with all the necessary details, like which order to change. So a solution would be that the *Metasonic Suite* automatically changes the order if I decide it should do so; and it should also re-run the MPS/MRP. I don't want to open SAP at all in this case – I already have *Metasonic Flow* open."

After Peter finished speaking, the consultants looked at each other, smiling.

"Seriously, guys," John said. "Don't tell me you also have a solution for this one. It's just unrealistic for you to have a solution for almost everything!"

"In fact, we do," Bob replied.

"But we will tell you about our solution after lunch," Al concluded. "I'm starving."

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

www.dbooks.org

The Problem, the Solution and the End – Final Part

Veni, vidi, vici. *Gaius Julius Caesar*

After lunch, the team was waiting for Al and Bob to show up. They were late. When they finally showed up, John said jokingly: "Seriously, guys. What took you so long? Really, it isn't suitable for computer wizards to be late."

Bob smiled and answered: "To quote an actual wizard: 'A wizard is never late. He arrives precisely when he needs to.'"

Al explained: "When we talked about your further expectations before having lunch, we decided to prepare something we want to show you."

"Now that we're all fed I think that we can get to the final part of our visit," Al continued.

"Peter said that what is lacking is integration of the *Metasonic Suite* into the existing IT environment, like for example SAP. He dislikes having to maintain two systems at one time – and he is totally right."

"But of course the *Metasonic Suite* can also do that," Bob continued. "The *Metasonic Suite* has a special feature called refinements. Refinements are added to function states and have a certain behavior. But before I continue – do any of you know what a web service is?"

John, Norma, and Peter looked at each other. Finally, John asked: "Well, I heard it's some IT thing you can connect to. Over the web. Which then does something." He paused. "Okay, that's not really helping."

Bob smiled and started to explain: "You can think of a web service as a kind of interface. Only that this kind of interface is designed to be used by other programs. These programs can connect to the interface and perform certain actions. Does this make any sense to you?"

After everyone nodded, Bob went on with the explanation. "Nowadays, a lot of programs have this kind of interface. The reason is interoperability. This way, the program can also be used from the outside to perform certain actions."

"Could you give me an example of such an action? This might clarify this issue…" John asked.

"Of course. An example: you have a process in the *Metasonic Suite* and also an ERP system. In this process, a customer sends an order to an employee. The message sent includes several parameters, like customer number or requested delivery date. After the employee reviews the order and accepts it in *Metasonic Flow*, there is an automated connection to your ERP system, where the system is told to create a new order with the parameters provided."

"Wait a minute." Peter said. "Isn't that exactly what I said before lunch?"

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_11, © The Author(s) 2013

"Yes it is," Bob answered. Smiling he said "That's why we are telling you about this." "Most big companies offer web services for their applications nowadays. For example Facebook, Amazon, and Microsoft, just to name a few."

Al continued his explanation. "As Bob already told you, we call this feature 'refinements' – because it refines – or in other words, clarifies what exactly to do in a function state; 'exactly' means, to define it as software code."

"Sounds reasonable," John said. "But what was that Bob was talking about concerning certain behavior of states? So I can tell each state to call such a web service?"

"Not only web services," Bob explained. "Refinements aren't limited to web services. A refinement can do everything possible by program code. It's not necessary that it even calls a web service. It could also do something like sending an email with your mail program, bring up a website, or format your hard drive."

"Format my hard drive? Seriously?" Norma asked. "Well, it's certainly possible for a refinement to do that, even though it's not recommended." Bob responded with a grin.

"Okay, let me get this straight," John tried to sum it up. "You are basically saying that I have the possibility to tell each state, be it function, receive or send, to do some magic. That magic is only limited by what the programming language can do. Did I understand that correctly?"

Bob and Al nodded. "Yes, that's correct." Al said.

"Where's the downside of that?" Peter asked.

"Well, for one you are limited to the restrictions of a programming language," Bob told him. "But probably the major downside is that you need a programmer to develop each refinement individually. And depending on the requirements for the refinement, this can take a lot of time. Depending on the programmer, that time could be costly. But these are the only downsides I can think of."

"Why don't we just show you what it looks like?" Al said.

"Good idea," Bob responded. "We tried to prepare a scenario similar to what Peter said before lunch," he continued. "That's why we were late."

"It's not really perfect, it's just a proof of concept, so you can see what's possible," Al told them.

"We will show you an example of a refinement that connects to the web service of an ERP system and creates an order using the parameters submitted," Bob continued. "We don't have a SAP instance here, but I happen to have a Microsoft Dynamics NAV server installed on my laptop because I am currently writing a book about it."

"What exactly is Microsoft Dynamics NAV?" Norma asked.

"I have heard about that," John answered. "It's Microsoft's approach to an ERP system. Their answer to SAP. Right?"

Bob nodded. "Right. Dynamics NAV principally can do everything SAP can do, it just looks different. The refinement approach also works fine for SAP if there are web services. We will show it to you with Dynamics NAV because it's installed on my machine."

"Please understand that, contrary to the previous workshops, this time we won't show you how to do this by yourself. You are not programmers and it would also take a while to explain how refinements work. So we will just show you what the result looks like."

"I'm fine with that," John answered. "I probably wouldn't understand that programming stuff anyway."

www.dbooks.org



Fig. 11.1 One last time the team gathers around the laptop so Bob and Al can explain how refinements work

After Al opened his laptop, the team gathered around him and Bob.

"Please note that what we will show you is just example data. If you install Dynamics NAV on your laptop, the database is filled with sample data for demonstration purposes only," Bob said.

First, Al opened the *Microsoft Dynamics NAV* desktop client and navigated to the *Sales Orders* section and then to the *Sales Orders – Open* section (Fig. 11.2).

"Here you can see all open orders present in the Microsoft Dynamics NAV database. We customized the view so you can see the order number, customer number, customer name, requested delivery date, and posting date," he continued.

"As you can see, for today there is no order present as yet," Bob told them.

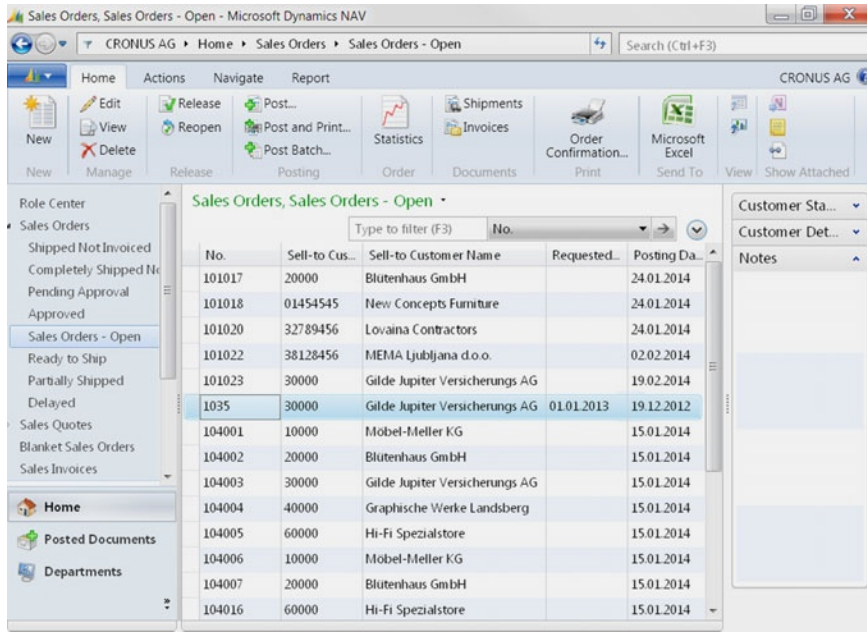


Fig. 11.2 This figure shows the *Sales Orders – Open* view of the *Microsoft Dynamics NAV 2013* client

After he showed them the order page of *Microsoft Dynamics NAV*, he opened the *Metasonic Suite*. "Now we will show you the process we prepared," he said (Fig. 11.3).

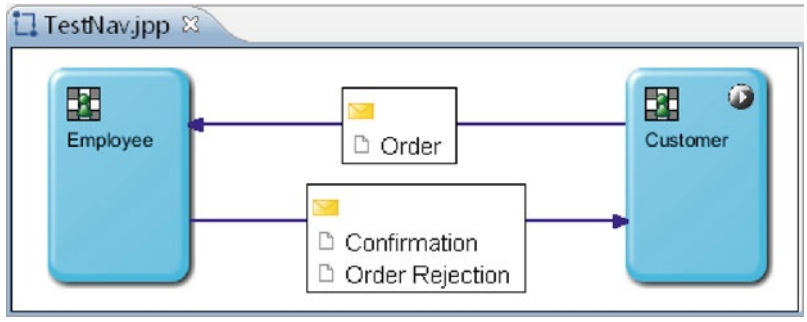


Fig. 11.3 The process consists of two subjects, *Employee* and *Customer*

Bob continued: "You know, time was short so we just refactored a process we prepared previously for another customer, to show you the essence of refinements. Therefore the process is just called *TestNav*, as you can see on the screen."

Al continued to explain. "As you can see, there are two subjects: the *Customer* and the *Employee*. There are three messages: *Order*, *Confirmation*, and *Order Rejection*. The process goes like this: the customer fills out an order and sends it to the employee. The employee checks the order and either accepts or rejects it. If he accepts the order, he

www.dbooks.org

sends back a confirmation and the order is automatically created in *Microsoft Dynamics NAV*. If there is something wrong with the order, the employee rejects the order."

"Let's have a look at the internal behavior of the *Customer*," Bob said.

Al opened the internal behavior of the *Customer* subject and started to explain it (Fig. 11.4). "You see, the behavior of the customer is pretty straightforward. First the order is filled out. Afterwards, the order is sent to the employee. Then the customer waits for an answer. No matter what answer the customer receives, the process ends."

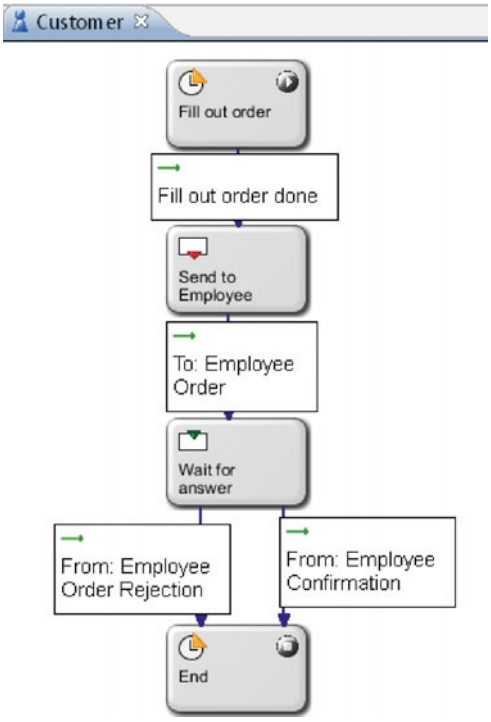


Fig. 11.4 The internal behavior of the *Customer* subject consists of very few states

"As far as I can see, no magic happens here?" John asked. "No, there are no refinements involved," Bob responded.

Al then opened the internal behavior of the *Employee* (Fig. 11.5). He explained: "As you can see here, the behavior is pretty simple too. First, the employee waits for an order. Then the order is examined. If everything is correct, the order is accepted and automatically created in the ERP system. After that, the employee sends a confirmation to the customer and the process ends. If there is something wrong, the employee sends a letter of rejection to the customer and then the process ends too."

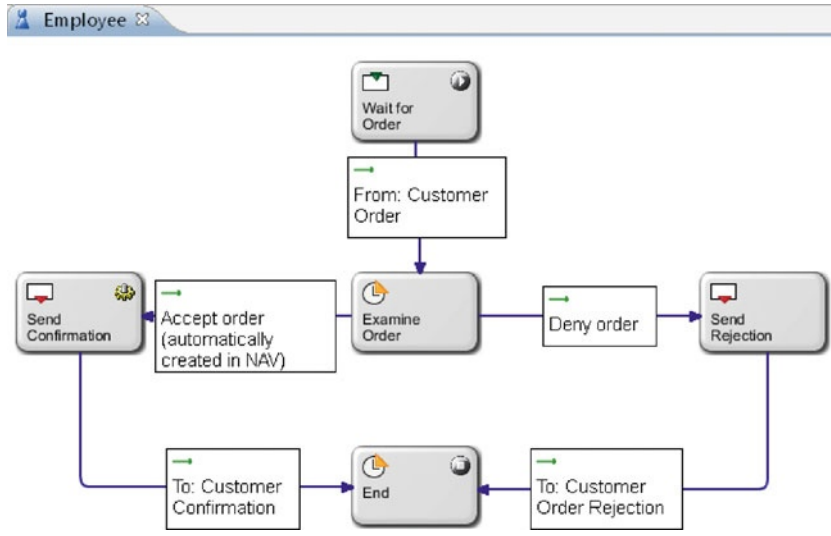


Fig. 11.5 The internal behavior of the *Employee* subject contains the *Send Confirmation* state, which includes a refinement

"So this is where the magic happens?" John asked. "In the *Send Confirmation* state?" "Yes," Bob confirmed. "The small gearwheel within the state indicates that there is a refinement behind it."

"You know, I'm just curious …" Peter said. "I probably won't understand anything, but can you show me that refinement? Just so I have a picture in my mind when I think of it."

Al smiled. "Of course we can do that."

He then right-clicked the *Send Confirmation* state, navigated to the *Refinement* tab and clicked the *Configure…* link. After that, the program code of the refinement was displayed on the screen (Fig. 11.6).

EmployeeRefinement.java

```
/**
 * This function provides the data to send in the named state.
 * transition1356109412034 State: Send Confirmation Confirmation
 * TransitionType SEND Customer
 *
 * @return a SendStateResponse containing the id of the transition to follow
 *         or null to stay in this state e.g.
 *         "return new SendStateResponse("transitionXXXXXXXX");"
 */
@RefinementGenerator(id = "state1356109378628")
public SendStateResponse sendStateSend_Confirmation(
        SendStateRequest sendStateRequest) {
    try {
        SalesOrderService salesOrderService = new SalesOrderService(
                salesServiceURL);
        SalesOrderPort salesPort = salesOrderService.getSalesOrderPort();
        SalesOrder s = new SalesOrder();

        s.setSellToCustomerNo(paramCustomerId.getValue());

        s.setRequestedDeliveryDate(XMLGregorianCalendarImpl
                .parse(paramRequestedDeliveryDate.getValue()));
        Holder<SalesOrder> sv = new Holder<SalesOrder>(s);

        salesPort.create(sv);

    } catch (Exception e) {
        System.out.println(e);
    }

    return null;
}
```

Fig. 11.6 This is what a refinement written in Java looks like. Please note that this is not the full code used in the example

www.dbooks.org

"Okay. I have no idea what that is," John said. He went on to ask, "So the refinement is just a magical set of numbers and letters?" "If you put it that way – yes," Al responded with a smile.

"Does that help you in understanding?" Bob asked Peter. "Well, now I know that a refinement is something beyond my understanding," Peter answered with a smile. "But I have a picture in my mind now."

"Okay, that's fine." Al said. "Now we will show you what the process looks like in *Metasonic Flow*. We prepared two users who we had already created and linked to the subjects. The user linked to the *Customer* subject is called *cus1* and the one linked to the employee is called *emp1*."

After he finished talking, Al started the validation environment and then opened *Metasonic Flow*. "Of course we already uploaded the process and the refinement."

When the *Metasonic Flow* browser window was open, he logged in as *cus1*. "The customer starts the process. So I now log in as our *cus1* user and start a new task of the *TestNav* process."

After doing so, the browser window changed to the first function state. There, Al clicked the *Parameters* tab and filled out the two parameters. He then continued talking: "I will set the *CustomerId* to '30000', which represents one of the German example companies present in Dynamics NAV, and the *Requested Delivery Date* to '2013-01-01'."

After that, the parameters were displayed on the screen and Bob said: "We could of course specify many more parameters. But we just want to show you an example."

Al nodded and continued. "Now I activate the *Fill out order done* checkbox and proceed to the next state." (see Fig. 11.7) There he chose *emp1* as the receiving subject because it was a send state, and reviewed the parameters one more time. "Here I send the message to the only available subject, *emp1*," he told them.

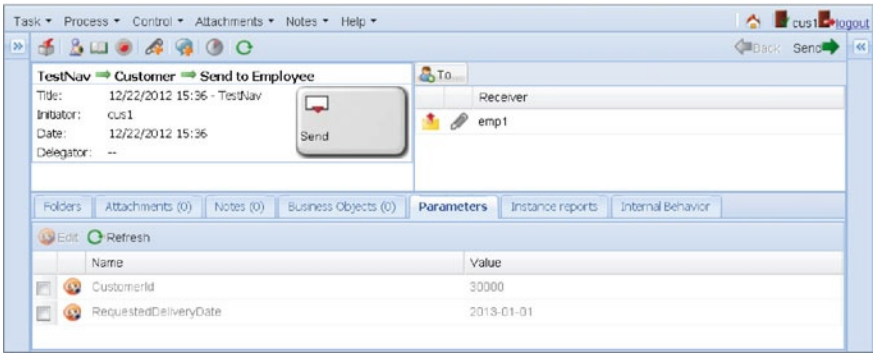


Fig. 11.7 The customer fills out the parameters and sends the order to the employee

"Now I will have to log out as the current user and log in again as *emp1* because the *cus1* subject is in a receive state," he continued. After he logged in as *emp1*, he opened the sole available active task.

"Here I receive the message sent by *cus1*," Al said and proceeded to the next step (Fig. 11.8). "Here I review the parameters *cus1* sent to me and decide whether I want to accept the order or not," he said, bringing up the *Parameters* tab. He went on to say, "I decide they are good to go, activate the *Accept order* checkbox and then – boom – the magic happens!"

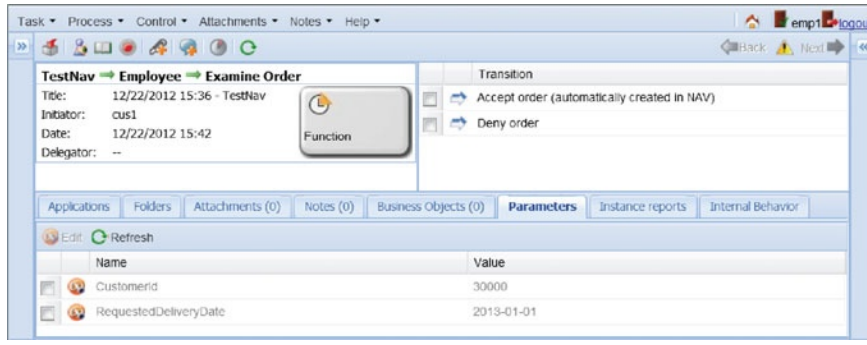


Fig. 11.8 The employee can review the received parameters and decide whether to accept the order or not

After he clicked the *Next* button, the loading took a while, but finally the next page showed up. The process was then in the *Send Confirmation* state (Fig. 11.9).

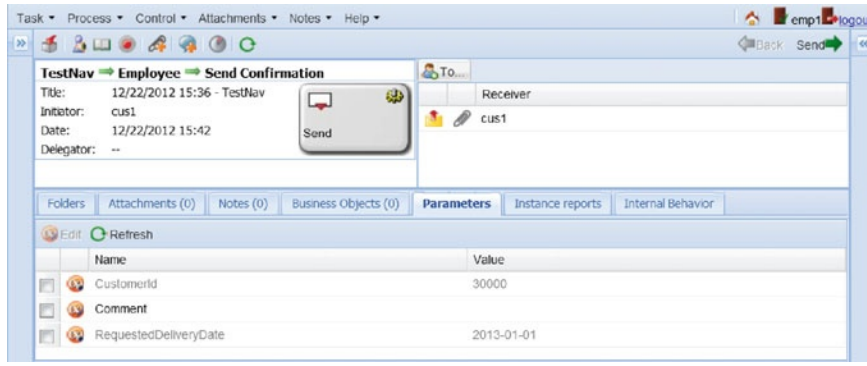


Fig. 11.9 The employee can now send a confirmation to the customer. The gearwheel indicates that this state contained a refinement which was executed

"Wait," John said. "That was it?" "Yes, John." Bob responded.

Al smiled and brought up the *Microsoft Dynamics NAV* window again. "Do you remember the parameters I entered?" Al asked. He refreshed the page and there it was – a new order for customer number 30000 with the requested delivery date of the first of January 2013 and the posting date of that day (Fig. 11.10).

www.dbooks.org

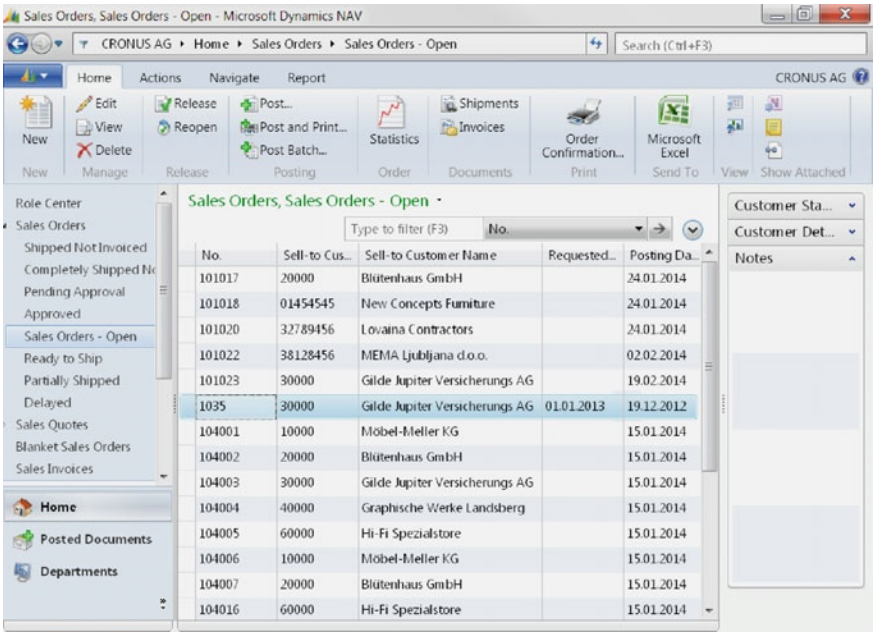


Fig. 11.10 The *Sales Orders – Open* page of the *Microsoft Dynamics NAV* client now displays the order created by the refinement

"I'm impressed," Peter said. "I could imagine working with that! I wouldn't even have to open my ERP client. Everything happened automatically in the background. I like that!"

"Glad you like it," Al said. "But to finish the process, our *emp1* has to send a confirmation back to the customer. In the *Parameters* tab, he has the option to specify a comment. I will just send the confirmation back without any additional details."

After that, he chose *cus1* as the recipient of the message and sent the message back. "The process is now finished for *emp1*. But we still have to accept that message for *cus1*," he said. So he logged out as *emp1* and logged in as *cus1* again. There he opened the active task and received the message. "As you can see, the process is now finished for the customer," he said (Fig. 11.11).

"And that's basically it," Bob concluded.

Norma, John, and Peter looked at each other, nodding in appreciation. John spoke for them, "I think it's fair to say that we really liked that demonstration."

"Yes, it's amazing what is possible with S-BPM," Norma added.

Also Pete liked it: "Yes, I think I can say that I have no more concerns. It's nice to know what's possible and what's not. Now that we know, I think we can build on that."

"I think so too," John said. "I will try to get an S-BPM task force together so we can lift our company to the next level. That shouldn't be much of a problem – the 'just think about how much money we could save' argument has always worked so far."

"For Al and me it has been very interesting to work with you," Bob said. "We always learn something new from each of our customers. And we really like your open-minded approach."

"But now our time is up," Al said. "As Bob said, it was fun working with you. But now another challenge awaits, and the 'superheroes of the computer age' have to move on now," he smiled. "I like that title," Bob said. "I think I will put it on my business card."

"We also want to thank you from our side," John told them. "We have learned much in such a short amount of time. Now we will try modeling S-BPM ourselves."

"Well, if you need help you know how to contact us," Bob said.

After that, the consultants said their goodbyes to John, Norma, and Peter and headed home. "Well then, my fellow superhero," Bob said to Al. "Let's look for another company that needs the help of a pair of superheroes or computer wizards in solving their communication and business process problems."

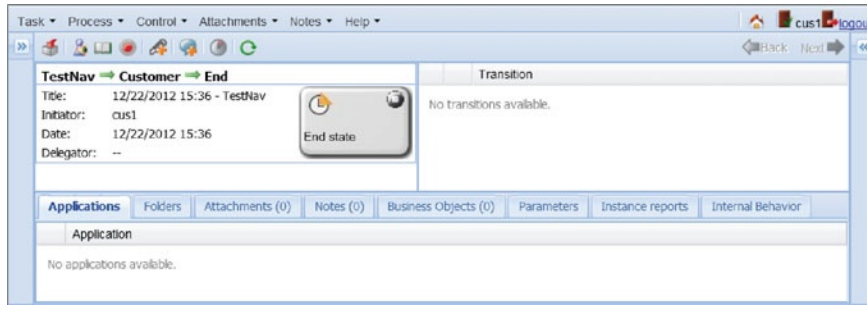


Fig. 11.11 After the internal behavior ends for the customer, the process is finished

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

www.dbooks.org

Troubleshooting

General note: If you are working with a computer program, always keep Murphy's Law in mind: anything that can go wrong will go wrong. It is a well-known fact that computer programs have the bad habit of being able to crash. You never know when it will happen, so the best approach is to be always ready for it. Save your work after each step. Make backup copies frequently. There are only a few things in this world that are more frightening than the rage of somebody whose computer has crashed after working with a computer program for hours without saving his/her work.

? I received a message in *Metasonic Flow* but can't see the parameters!

✓ Check the internal behavior of the affected subject in the *Metasonic Suite*. Make sure that it has the right readable parameters set.

? I can't upload my process in the *Modelmanager*! It says there is an error!

✓ Check your process in the *Metasonic Suite*. Make sure that every subject has a start state and is assigned a role (this also applies to external subjects). Make sure that only one subject begins with a function state – all other subjects have to start with receive states.

? I can't save my process anymore – I get a nullpointer exception!

✓ I'm afraid you will have to create a new process and model everything again. Life can be hard sometimes, deal with it.

? I edited a parameter. Now the parameter exists twice. If I add the edited parameter, it is displayed twice.

✓ You will have to delete all parameters and messages and create them again. Shit happens.

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Noncommercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9_12, © The Author(s) 2013

The Institute of Innovative Process Management

The Institute of Innovative Process Management (I2PM) was established in 2010 as a nonprofit organization. It brings together scientists and practitioners in the area of Business Process Management and related topics. The objective is the transfer of know-how between theory and practice. Deliverables are both concepts, prototypes, solutions etc. developed in science to be applied in practice and demands, problems, and experience formulated by practitioners as triggers and inputs for research.

In order to reach our goals

- we manage and foster scientific work,
- we organize events to share knowledge and experience,
- we publish findings and results,
- we support research and development projects,
- we cooperate with other academic organizations, associations, institutions, and enterprizes.

The most visible activities so far are the international S-BPM ONE conference series and the Open S-BPM initiative. Open S-BPM was initiated early in 2012 to stimulate research on the S-BPM approach and to further spread its paradigm. Since then numerous institutions have been contributing concepts and solutions to a set of methods and tools which can be combined in order to introduce agile BPM within organizations and across organizations.

Werner Schmidt
Institute of Innovative Process Management (I2PM)
Ingolstadt, January 2013

A. Fleischmann et al., *S-BPM Illustrated*, DOI 10.1007/978-3-642-36904-9, © The Editor(s) (if applicable) and the Author(s) 2013. The book is published with open access at SpringerLink.com 2013

www.dbooks.org

S-BPM ONE Conference Series

Established in 2009 the S-BPM ONE conference series provides an exciting opportunity for researchers as well as business and education practitioners to contribute to knowledge advancement in and across the continuously growing S-BPM community.

The conferences serve as a forum to discuss S-BPM's potential to foster and leverage business innovation, operational excellence, and intra- and interorganizational collaboration by integrating advanced information technology with organizational and managerial methods. However, topics are not limited to S-BPM's straightforward approach towards the analysis, modeling, implementation, execution, and management of interaction patterns with an explicit stakeholder focus.

Participants are also invited to bring into discussion further, widely undefined themes pertaining to the engineering and management of systems and organizations, particularly with respect to the areas of interaction culture, process-aware information systems, strategic alignment, and governance structures. Event mottoes like "Learning by Doing – Doing by Learning," "Enabling Transition," or "Running Processes – Opening up for new approaches to practical and successful business process management" underline the conference philosophy.

Attracting more than 100 researchers and practitioners from more than eight countries in a vibrant, still balanced way, the conference plays an important role for developing (S-)BPM theory and practice. The contributions are published as proceedings either in Springer's prestigious Lecture Notes in Business Information Processing (LNBIP) or Communications in Computer and Information Science (CCIS) series.

Readers of S-BPM illustrated are explicitly invited to submit papers and participate in the conference. For more details on the S-BPM ONE conference series see www.s-bpm-one.org.

Werner Schmidt
Institute of Innovative Process Management (I2PM)
Ingolstadt, January 2013