Product based workflow design with case handling systems

Product Based Workflow Design with Case Handling Systems

Irene Vanderfeesten, Hajo A. Reijers, Wil M.P. van der Aalst

Technische Universiteit Eindhoven,
Department of Technology Management,
PO Box 513, 5600 MB Eindhoven, The Netherlands
{i.t.p.vanderfeesten, h.a.reijers, w.m.p.v.d.aalst}@tm.tue.nl

Abstract. *Case handling systems* offer a solution to the lack of flexibility and adaptability in workflow management systems. Because they are data driven they potentially provide good support for *Product Based Workflow Design* (PBWD). In this paper we investigate to which degree current workflow technology, in particular two case handling systems (FLOWer and Activity Manager), are able to support PBWD. This is done by elaborating the design process of a case from industry in both systems. From this evaluation we concluded that current workflow technology is not yet completely ready for supporting PBWD. Therefore, we recognize that better tool support is needed to make PBWD more suitable for practical use.

1 Introduction

In the past decades, process-orientation has gained a strong foothold in various fields, notably in the business management and information systems disciplines. This is illustrated by the emergence of process-oriented transformation approaches, like Business Process Redesign (BPR) [11,16], on the one hand and process-aware information systems, like workflow technology [3], on the other. With this rise, the historic focus on the *data* that is being processed within businesses settings - and by information systems in particular - has blurred. It should be remembered that during the 70s and 80s the majority of information systems development projects would start with a thorough data analysis, leading to conceptual data models, while nowadays similar projects typically start with mapping the business to be supported in the form of process models.

Recently, nothing short of a *data revival* has set in in the Business Process Management (BPM) community, bringing back attention for data aspects. This phenomenon can be distinguished in at least two places. Firstly, various problematic issues with workflow and BPM systems are being countered with the introduction of systems that put much more emphasis on the data that is being handled (e.g. case handling systems [2,7]), in this way moving away from a purely control-flow centric perspective. Secondly, innovative BPR approaches are emerging that, rather counter-intuitively, take business data processing requirements as starting point for generating a new business process design (e.g.[26,30]).

In this paper, we will investigate to what extent synchronous movements towards a higher data awareness in the fields of (i) workflow management and (ii) business process design can mutually reenforce each other. In the recent past, we have worked on the development and application of the method of Product-Based Workflow Design (PBWD). This method takes a static description of an (information) product as a starting point to derive an improved process design. The idea to focus on the product instead of on an existing process when redesigning a process was introduced by Van der Aalst [1] and is based on a similar approach in manufacturing processes. Since its conception, this method has been worked out in some detail [25,26,27] and has been successfully applied in industry in over a dozen of occasions. At the same time, the manual application of PBWD in practice proves to be a time-consuming and error-prone affair. It is likely that the absence of automated tools to support the application of PBWD hinders the wider adoption of the method, despite its successes in bringing back cycle time and service times of actual business processes with 30% or more [25]. On the road to the development of PBWD support tools it

seems wise to consider some of the existing tools that could already deliver (partial) support for the application of PBWD. A notable candidate for such support would be current case handling technology. After all, just like traditional workflow management systems, case handling systems operate on the basis of a pre-defined process model. In contrast to workflow technology, however, case handling systems implement various data management features [7].

The objectives of the paper can now be formulated as follows: (i) to determine whether the concepts of PBWD can be translated to the concepts in current case handling systems, (ii) to establish to what degree build-time features of case handling systems support the design of workflow models based on PBWD, and (iii) to find out how current case handling tools could be enhanced to support PBWD more thoroughly. Fulfilling these objectives could also be useful to determine the desirable features of a specifically tailored support tool for PBWD, i.e. without using current case handling systems.

The structure of this paper is as follows. In the next two sections, we will shortly review case handling systems and the PBWD method respectively, forming the fundaments of this paper. In Section 4, we will present our assessment of two existing case handling technologies, i.e. Pallas Athena's FLOWer and BPI's Activity Manager. To conclude the paper, we present the major implications from our assessment and directions for further research.

2 Case handling systems

Traditional workflow and business process management (BPM) systems are characterized by well-known limitations in terms of flexibility and adaptability [5]. These limitations can be associated with the dominant paradigm for process modelling found in these systems, which is almost exclusively activity-centric [12]. The lack of flexibility and adaptability leads to many problems and inhibits a broader use of workflow technology. In recent years many authors have discussed the problem [5,9,10,13,17,19,20] and different solution strategies have been proposed. Basically, there are three ways to provide more flexibility: (i) dynamic change [13,24,28], (ii) worklets [8,29,32], and (iii) case handling [2,7].

The basic idea of *dynamic change* is to allow changes at run-time, i.e., while work is being performed processes may be adapted [5,13,24,28]. Clearly, dynamic change mechanisms can be used to support flexibility and adaptability. A dynamic change may refer to a single case (i.e., process instance) or multiple cases (e.g., all running instances of a process). Both changes at the instance level and the type level may introduce inconsistencies, e.g., data may be missing or activities are unintentionally skipped or executed multiple times. A well-known problem is the "dynamic change bug" which occurs when the ordering of activities changes or the process is made more sequential [13]. These issues have been addressed by systems such as ADEPT [24,28]. Using such a system the consistency of a process can be safeguarded. However, an additional complication is that the people changing the processes should be able to modify process models and truly understand the effects of a change on the *whole* process. In real-life applications few people are qualified to make such changes.

Worklets [8] allow for flexibility and adaptability by the late binding of process fragments, i.e., in a process some activities are not bound to a concrete application or subprocess and only when they need to be executed a concrete application or subprocess is selected. YAWL [4] is an example of a system that supports this idea. In YAWL activities may be handled by a worklet handler, this handler uses an extensible set of ripple-down rules to select the right worklet (i.e., a concrete application or subprocess). Similar ideas have been proposed by other authors (e.g., [32]) and even implemented in commercial systems (cf. the Staffware extension that allows for process fragments [29]). Although the worklets mechanism is easier to be used by end-users than most dynamic change mechanisms, the scope is limited and only particular forms of flexibility and adaptability can be supported.

Case handling is another paradigm for supporting flexible and knowledge intensive business processes. The concept of *case handling* offers a solution to the lack of flexibility in traditional workflow systems [7]. Case handling is supporting knowledge intensive business processes and

focuses on what *can* be done instead of on what *should* be done. To support this, a case handling system is much more data driven than a workflow system. The central concept for case handling is the *case* and not the routing of work or the activities. The case is the product that is manufactured in the process based on the data that is processed. The core features of case handling are [2,7]:

- avoid context tunneling by providing all information available (i.e., present the case as a whole rather than showing just bits and pieces),
- decide which activities are enabled on the basis of the information available rather than the activities already executed,
- separate work distribution from authorization and allow for additional types of roles, not just the execute role,
- allow workers to view and add/modify data before or after the corresponding activities have been executed (e.g., information can be registered the moment it becomes available).

These core features of case handling are supported by systems such as FLOWer [22]. Other systems such as BPi's Activity Manager [15] only support some of these features. Unlike dynamic change and worklets, case handling provides implicit flexibility, i.e., there is no need to change a process model or to select a particular worklet. Moreover, as the list of core features suggests, case handling takes a broader perspective also incorporating aspects as work distribution and information collection.

3 Product Based Workflow Design

PBWD [1,25,26,27] is a revolutionary approach to workflow process design, i.e. a clean-sheet of paper is taken to design the complete process from scratch. Rather than the activities and the workflow process itself, it takes the processing of data and the workflow end product as the central concepts. This approach has several advantages that are described in [25,31]. The most important advantage is that PBWD is *objective*. In the first place because the product specification is taken as the basis for a workflow design, each recognized information element and each production rule can be justified and verified with this specification. As a consequence there are no unnecessary tasks in the resulting workflow. Secondly, the ordering of (tasks with) production rules themselves is completely driven by the performance targets of the design effort.
The workflow product is represented by a *Product Data Model* (PDM), i.e. a network structure of the construction of the product. The approach of PBWD is very similar to the way in which manufacturing processes are structured. This will be explained in more detail in the remainder of this section.
Section 3.1 shortly describes the similar concepts in manufacturing, while Section 3.2 subsequently elaborates on the important concepts of PBWD. Finally, Section 3.3 introduces an industry case as an example of PBWD, that will be used throughout the assessment of the two concrete systems in Section 4.

3.1 Bill-Of-Material (BOM)

In manufacturing, often a static representation of the product is used to define the assembly lines. Figure 1.1 shows such a representation for the assembly of a car. A car is made of 4 wheels, a chassis, and an engine. The structure of the assembly line can be derived from the picture as follows: first, the four wheels and the chassis are put together, resulting in a subassembly product. Next, the final assembly takes place by putting the subassembly product and the engine together. The result then is the car. The representation of the product and its parts is referred to as the Bill-Of-Material (BOM) [21] and is also used in information systems, e.g. MRP- and ERP-systems for production planning and control.

Manufacturing and service-oriented processes have a lot in common [23], e.g. managing the process focuses on the routing of work and the allocation of work to resources. Because of these similarities it was considered worthwhile to explore the applicability of some concepts from the field

(1.1) (1.2)

Fig. 1. (1.1) The Bill-Of-Material (BOM) of a car, (1.2) The product data model of suitability to become a helicopter pilot. The meaning of the elements is as follows: (a) decision for suitability to become a helicopter pilot, (b) psychological fitness, (c) physical fitness, (d) latest result of suitability test in the previous two years, (e) quality of reflexes, (f) quality of eye-sight

of manufacturing to administrative and information intensive processes (referred to as workflow processes). The PBWD method derives a process model on the basis of the structure of the product in an administrative business process. This product structure is called a PDM and is explained in the next section.

3.2 Product data model (PDM)

The product of a workflow process can be an insurance claim, a mortgage request, a social benefits grant, etc. Similar to the BOM, a PDM of this product can be made. However, the building blocks are not the physical parts that have to be assembled, but the data elements (e.g. name, birth date, amount of salary, type of insurance and register of holidays) that have to be processed to achieve new data.

Figure 1.2 contains a small and simple example, comparable to the simple BOM of the car in Figure 1.1. It describes the decision whether the applicant is allowed for a training to become a helicopter pilot (see also [25]). Persons that want to become a helicopter pilot should meet some requirements: they should be healthy, their eye-sight should be excellent, they should pass a psychological assessment, and they should not have been rejected in the previous two years. The figure shows that the final decision whether a person can become a helicopter pilot (data element a) is dependent either on the data elements (b) and (c), or on (f), or on (d). In reality, these different combinations reflect the different conditions under which certain operations can be executed. In case there is a result of a recent suitability test (d), this information directly determines the outcome (a). Also, in case the value for the quality of eye-sight of the applicant is bad (f) this directly leads to a rejection (a). In the other cases, the results of both a psychological (b) and a physical test (c) are needed. One level lower, the physical test (c) consists of the results for the quality of reflexes (e) and for the quality of eye-sight (f).

The *data elements* of the PDM are depicted as circles. The *operations* on these data elements are represented by arcs. The arcs are 'knotted' together when the data elements are all needed to execute the particular operation. Compare for instance the arcs from (b) and (c) leading to (a) on the one hand, to the arc from (d) leading to (a) on the other in Figure 1.2. In the latter case only one data element is needed to determine the outcome of (a), while in the case of (b) and (c) both elements are needed to produce (a).

The helicopter pilot example, that we discussed here, is very small. Typically, in industry the PDMs are much larger; possibly containing hundreds of data elements. *Based on such a PDM, a*

workflow process model can be obtained by grouping data elements and operations into activities (see also [25,27]).

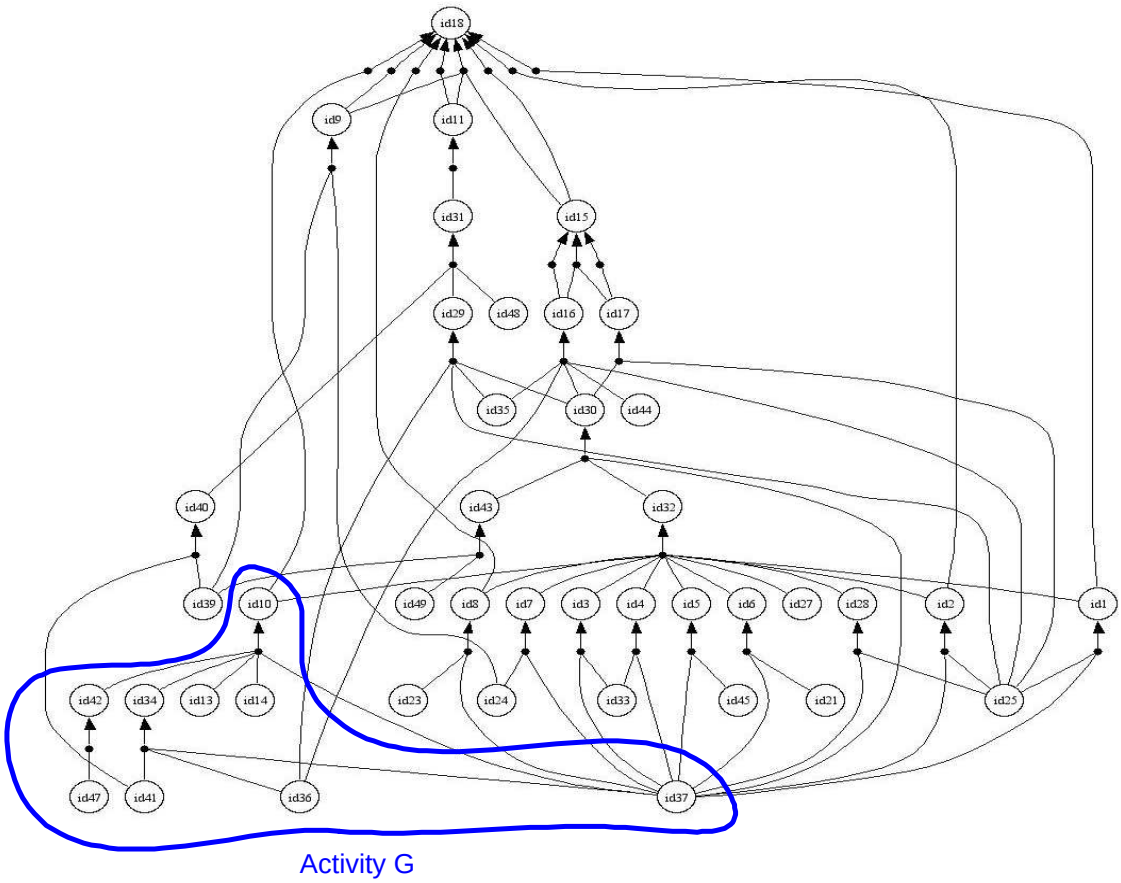

Fig. 2. The PDM for the GAK case

3.3 The GAK case

In this section we introduce a case from industry as a motivating example. This example is used *in the assessment of the two contemporary case handling systems*, as described in Section 4. The subject of the case study is the GAK agency (currently known as UWV) which is responsible for awarding unemployment benefits in the Netherlands. The process in question deals with the decision that the GAK has to make on whether or not to grant such benefits once a request has been received. The typical factors that should be taken into account are the reason for the applicant to have become unemployed, the length of the period that the previous job was held, and the coverage regulations.

The PDM for the GAK case is shown in Figure 2. A detailed description of the case and of the data elements can be found in [25]. The next section describes how we have assessed the process of design in two contemporary case handling systems based on the GAK PDM. The used design is elaborated on in [25]. The process model is manually derived from the PDM, taking into account several process requirements and restrictions. For example, the processing order of the various operations was determined such that the expected number of additional work at any point in the process was minimized for the average case. The result of this manual design process is also shown

in Figure 3. In this figure the contents of each activity (i.e. data elements and their operations) are also indicated. For example, the three operations on the data elements $i10$, $i13$, $i14$, $i34$, $i36$, $i37$, $i41$, $i42$, and $i47$ (Figure 2) are grouped into activity G (Figure 3).

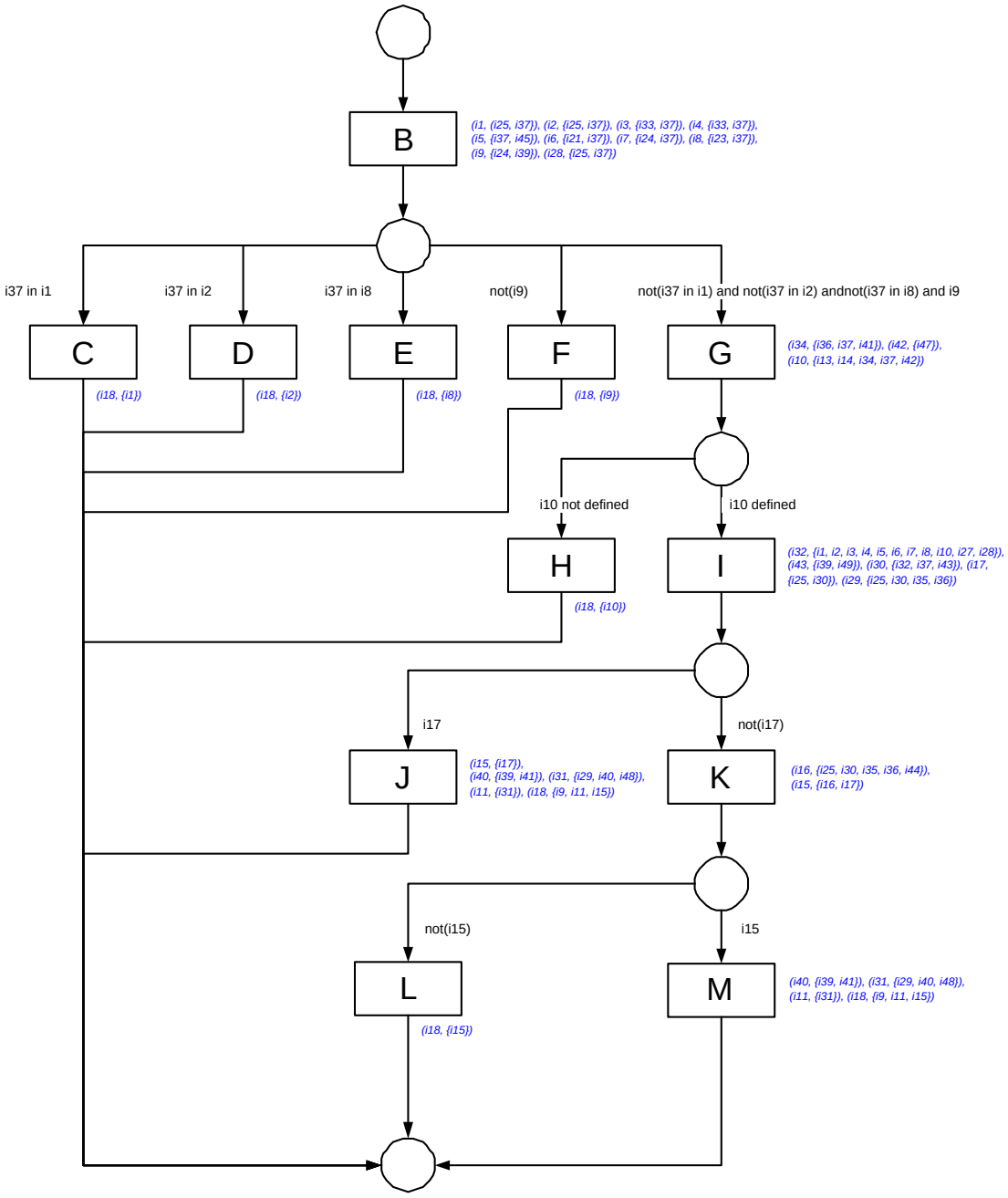

Fig. 3. The process model for the GAK case

4 Assessment

As was explained in the introduction, workflow management systems focus on the control-flow perspective, while case handling systems are more data-driven. Because of their focus on data, case handling systems may provide support for PBWD. In order to investigate their possibilities and potential support for PBWD, we have selected two case handling systems: FLOWer, by Pallas Athena [22], and BPi's Activity Manager [15]. Both systems use their own terminology, which is sometimes confusing. However, in this paper we stick to a more standardized workflow terminology for reasons of understandability (cf. Table 1).

Table 1. Different terminology: Standard workflow terminology, FLOWer, Activity Manager

Workflow	FLOW*er*	Activity Manager
Process	Case type	Procedure
Activity	Action	Step
Operation	-	Activity
Data element	Data object	Constant / Case variable

The next two sections elaborate on the way in which PBWD can be used to design a process model in FLOWer and Activity Manager. The focus in both assessments is on the process of designing and defining the process model based on the PDM [1]. In general, the following steps should be taken and supported by the system to get from a PDM to a process model:

1. The PDM must be translated to the specific system. This means that either the data elements or the operations (or both) must be mapped on concepts in the system and must be specified.
2. The activities must be defined as groups of data elements and/or operations. There must be an easy way to transfer an operation or data element from one activity to another, as a way of exploring various designs. Also, the order of activities must be established, because precedence relationships should be respected.
3. The process model must be finalized with for instance information on resources, etc.

Sections 4.1 and 4.2 describe to what extent this strategy could be followed when designing a process model in FLOWer and Activity Manager, starting from the PDM of the GAK case.

4.1 FLOWer

FLOWer is a case handling system developed by Pallas Athena [22]. It consists of a number of components, of which FLOWer Studio is the graphical design environment. FLOWer Studio is used during build-time to define case definitions, consisting of activities, precedences, data objects, roles and forms. In the assessment of FLOWer we kept the grouping of operations and their data elements independent of the definition of activities as long as possible. This gave us the flexibility of easily transferring data elements and operations from one group to the other, as described in step 2 of the general strategy. The procedure we followed to get to a process model is described below and illustrated by a series of screenshots from the FLOWer system. Note that we chose to include a considerable amount of system screenshots to give the reader sufficient insight in the systems and the steps we conducted.

1. All data elements are put in the FLOWer model. In Figure 4 and 5 the list of data elements is shown on the left hand side of the main window.

[1] Note that the process of designing and defining a process model based on a PDM is different from the common way in practice to design a process model. Instead of using a subjective workshop setting (i.e. interviews, observations, etc.) to discover the process model, a more objective approach is used starting from the product structure.

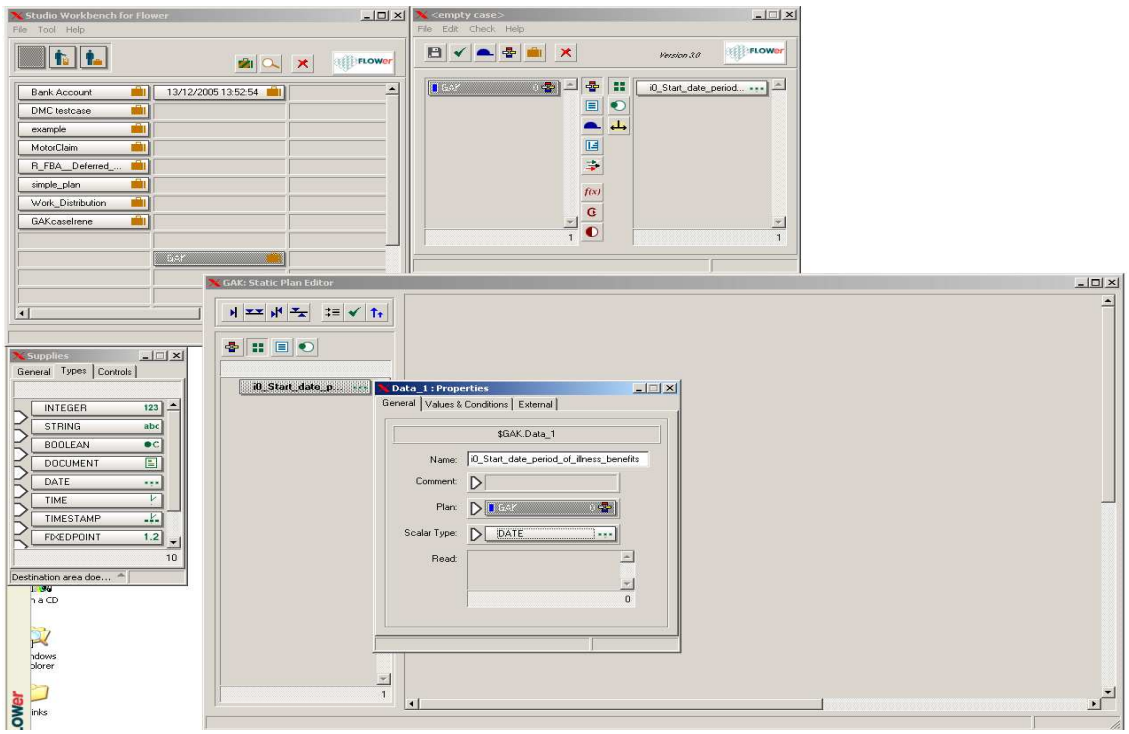

Fig. 4. After starting a new case type definition the first step we conducted is the definition of a data element in FLOWer Studio.

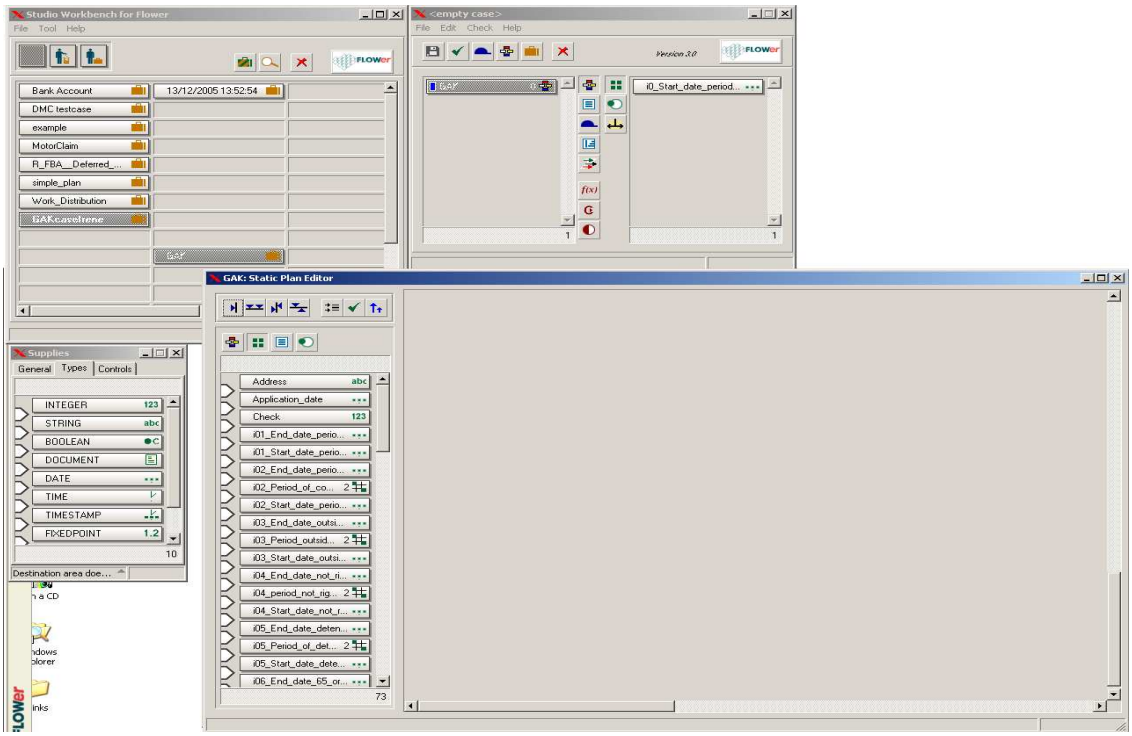

Fig. 5. All data elements are defined. An overview of the data elements is given on the left-hand side of the Static Plan Editor.

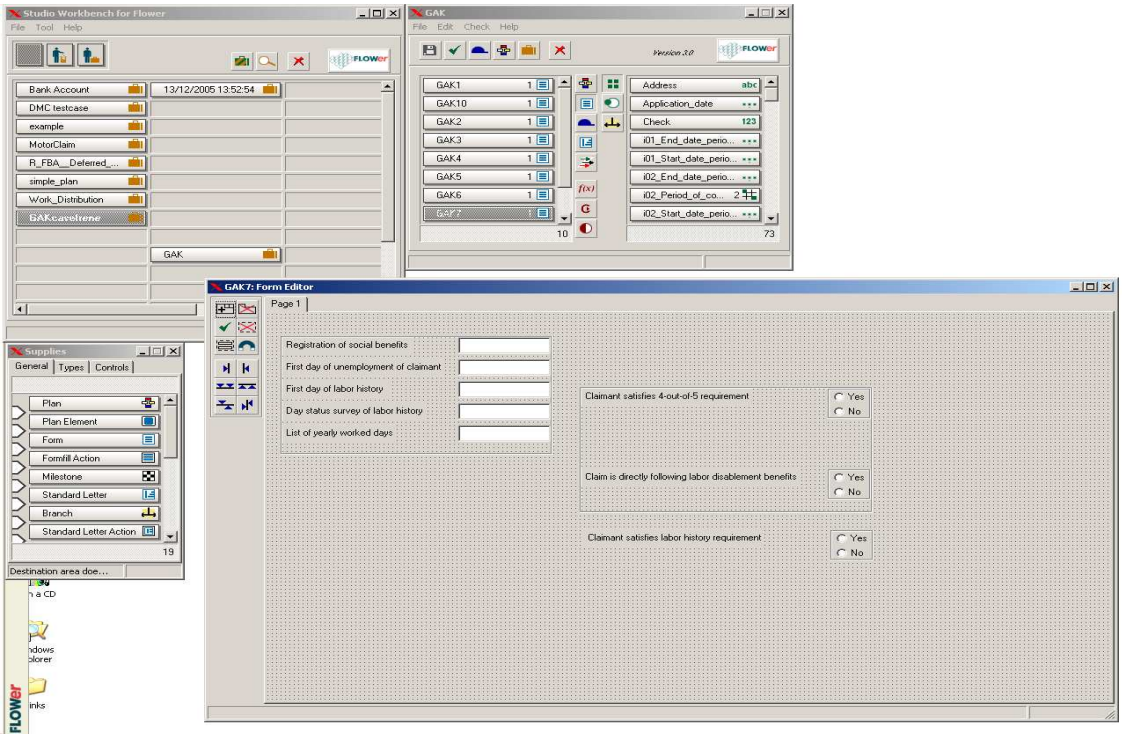

Fig. 6. Next, several forms are defined that contain groups of data elements. This picture shows a list of the forms on the left-hand side of the upper right window, and a detailed view on form GAK7. This form corresponds with a group of data elements: $i25$, $i30$, $i35$, $i36$, $i44$, $i16$, $i17$, and $i15$

.

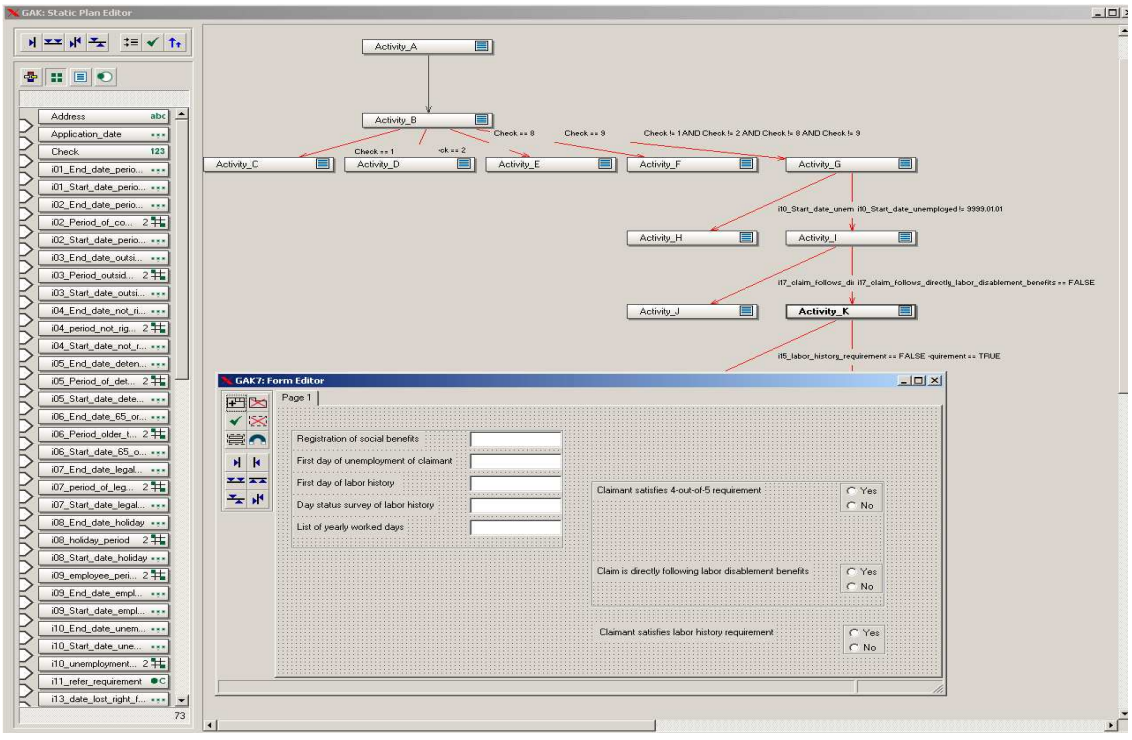

Fig. 7. Finally, the process model, containing the activities, is defined in the FLOWer Static Plan Editor. Each activity is linked to a form (for instance activity K is linked to form GAK7) and ...

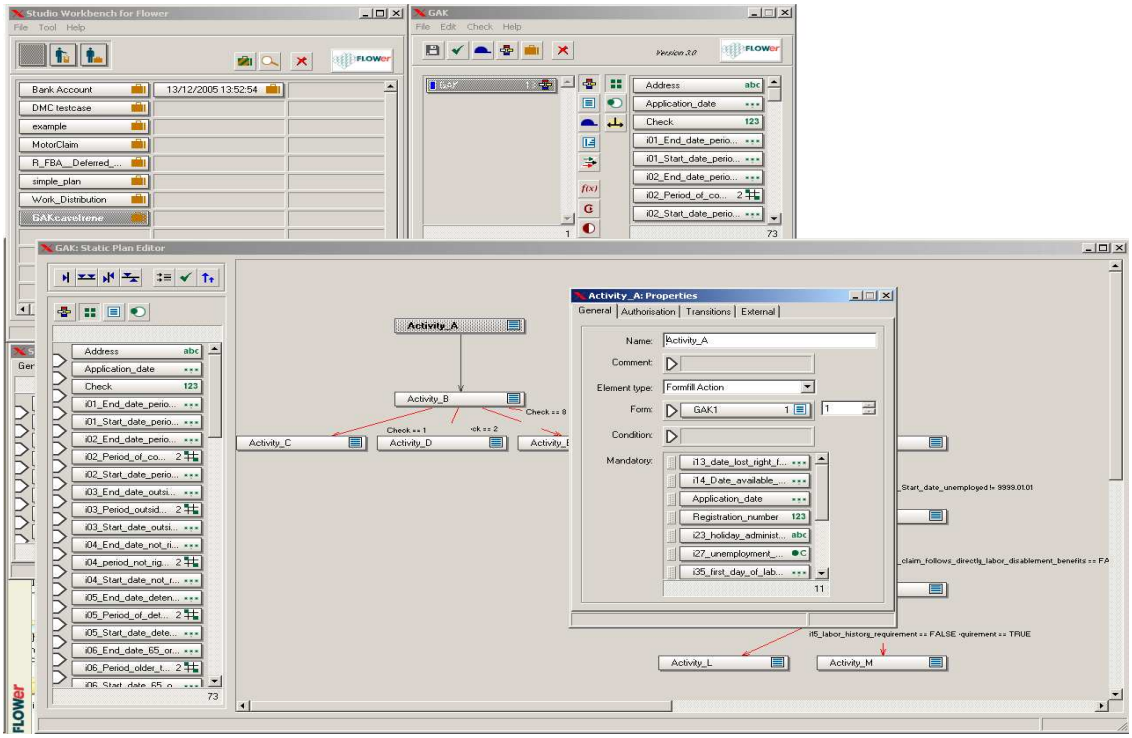

Fig. 8. ... mandatory data elements are defined for each activity.

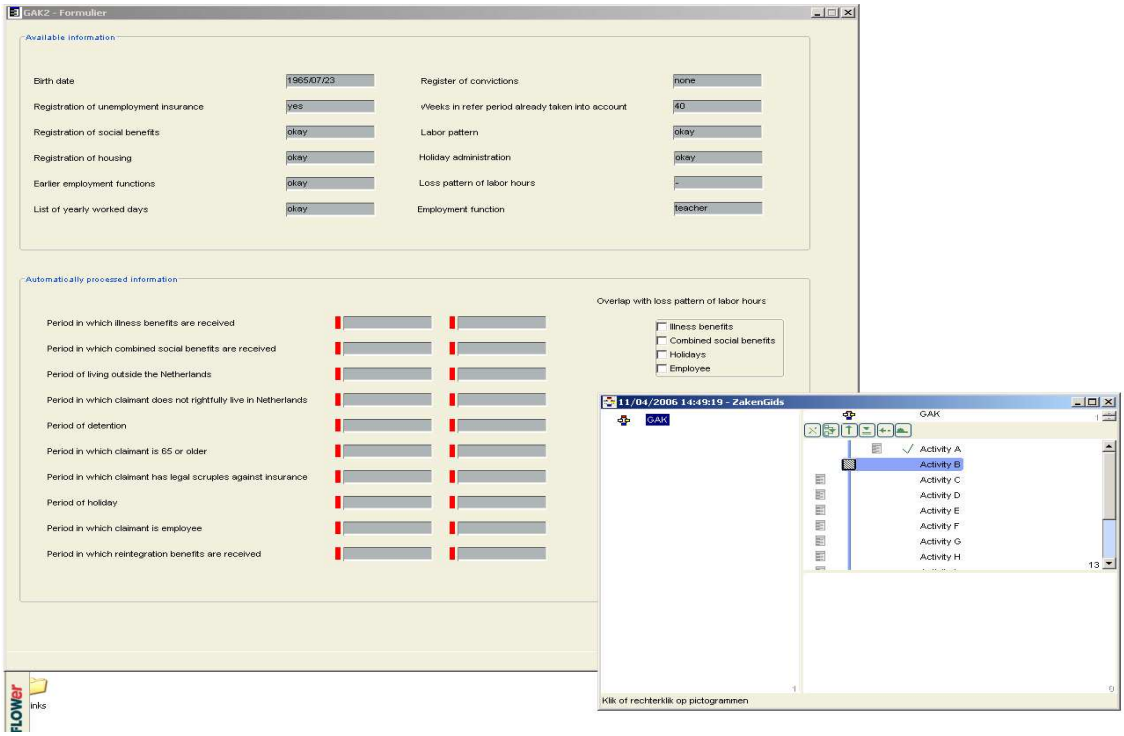

Fig. 9. After releasing the case type, the runtime environment can be started. When the form for activity A is filled out, activity B can be performed, as the time line shows. The form for activity B shows the available data elements (which come from activity A) and the mandatory data elements to be filled in in this activity (indicated by red bars).

2. The existing data elements are grouped onto forms. For each group of data elements a new form is made. See Figure 6 for an example.
3. An activity is defined for each form. Figure 7 partly shows the resulting process model for the GAK case together with the form for activity K.
4. Per activity the mandatory data elements[2] are indicated, i.e. all data elements that are on the form for this activity (Figure 8). Also, resource information is added in order to make the model executable in FLOWer's runtime environment (Figure 9).

Note that the final process model is designed according to the description and proposal for improvement in [25].

4.2 Activity Manager

BPi's Activity Manager is an "add-on" that can be used in combination with a workflow management system, such as COSA and Staffware [18]. For demonstration purposes also a stand-alone version can be used. In this research we used this stand-alone version because it is easier to manage as we could abstract from an actual workflow management system interacting with the Activity Manager. Activity Manager combines the structure and control of a workflow management system with the flexibility of case handling. Activity Manager imports the process model from the workflow management system via a database and provides a way to further define the activities in this model by elaborating the operations.

In this section, the running example is used again to investigate how Activity Manager can support product driven design. Activity Manager starts with some a-priori process model that needs to be imported before it can be used as a tool to further define operations. Therefore, it was *not* possible to reason only from the PDM to a process model as was described by the general strategy. To be able to work out the GAK case in this context we took the existing process description from [25] and used this as a starting point. The steps we performed are as follows, again illustrated with a series of screenshots:

1. The activities are defined via the database. See Figure 10.
2. The operations are added to the activities. Figures 11 and 12 illustrate the definition of an operation for activity B.
3. The data elements are defined as constants in the properties of every operation. Figure 12 shows how data element $i1$ is defined as a constant.
4. The conditions are defined for every operation (Figure 13). Using these conditions it is possible to implement the hierarchical dependencies between operations, i.e., if a certain operation needs the outcome of another operation as input, then the latter must be executed before the first one (Figure 14).

Note again that the final process model is designed according to the description and proposal for improvement in [25].

5 Discussion and conclusion

In this paper we have investigated to what extent current case handling systems are able to support PBWD by evaluating FLOWer and Activity Manager. As we have seen in the previous sections both systems still have quite a strong focus on the control-flow of the process, despite of their additional focus on data. However, in FLOWer we really can start reasoning from the PDM. As explained in Section 4.1, this provides an opportunity to focus on the grouping of data elements instead of defining activities directly. By putting groups of data elements on one form and playing around with these combinations it is possible to form activities based on the data and operations

[2] In FLOWer a mandatory data element is an element that has to be filled in when executing an activity. An activity with a mandatory data element can only be completed if a value for this element is given in the corresponding form. However, this value can be given in earlier activities.

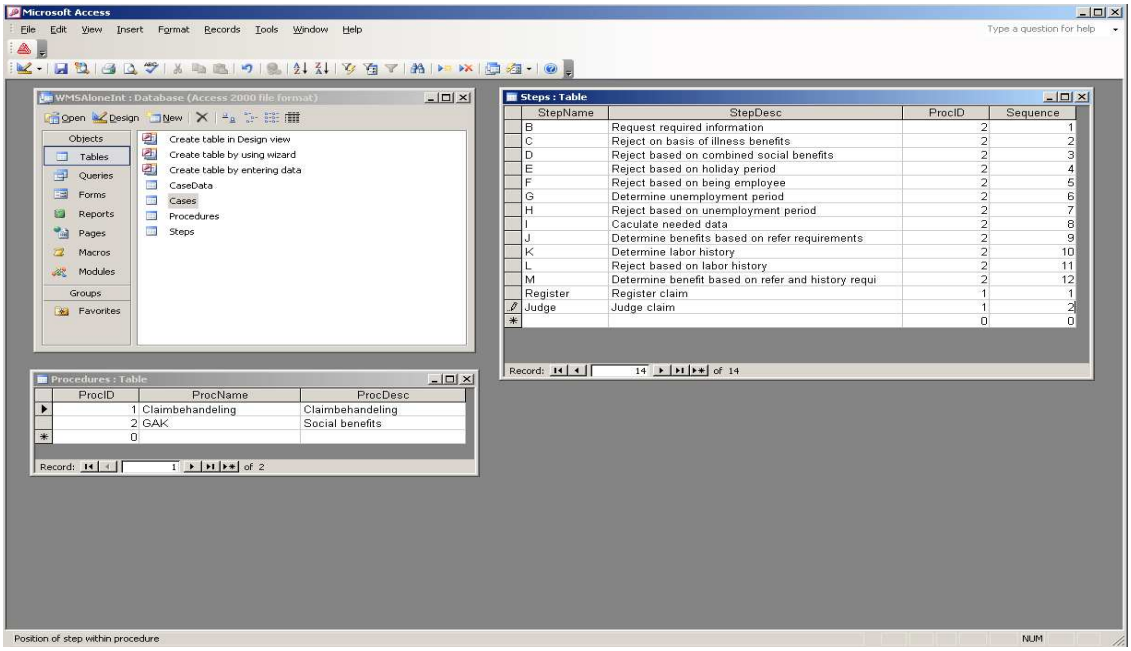

Fig. 10. The first step in the assessment of the stand-alone version of BPi Activity Manager is the definition of a process model as a Microsoft Access database of activities.

of the PDM instead of first defining the activities and afterwards determining what should be done in these activities. In contrast, BPi's Activity Manager is more process driven than data driven and starts from the definition of a process model. Of course, this follows from the fact that Activity Manager is "added on" to a workflow system, which only allows Activity Manager to further specify the already given process structure. Because of this, it is not possible to directly design a process model based on the PDM only in Activity Manager. The user needs to have a good understanding of how the activities are organized and what the content of each activity should be. This means that the process of designing a process model based on the PDM should then be done outside the tool, in such a way that the result (i.e. the activities including their operations) can be implemented in the system. Taking this design perspective we can remark that FLOWer offered the best assistance in creating a process model based on the product structure.

Looking at the evaluation from a conceptual viewpoint, we can conclude that both systems do not (yet) have a possibility to display the PDM as a hierarchical structure, which therefore would be a nice extension to use these systems as PBWD support tools. However, all concepts of the PDM and PBWD could be mapped to concepts in Activity Manager (see Table 1), while FLOWer is able to represent all concepts except for the operations.

This evaluation shows that current workflow technology, and even case handling systems, are not yet completely ready for PBWD. The research challenge now is to develop good support for applying this method in practice. The first contribution of this assessment is an overview of how existing systems can be improved to support PBWD. In close cooperation with suppliers of case handling systems we will further investigate the opportunities of using their systems. Secondly, we have learned some lessons for the development of specific tools for PBWD support. It seems to be important to (i) display and edit the PDM in the tool, and (ii) to somehow circumvent direct relations from activities to data elements. Finally, future work will focus on the discovery and collection of data elements (i.e. the derivation of a PDM). At this point in time, the ProM import framework for process mining [6] already supports mining based on data elements [14]. On a general level, this research shows that current workflow technology is *not* neutral towards the kind of process design. Even data-focused technology, such as case handling systems, still needs some control-flow information right from the start of the design process.

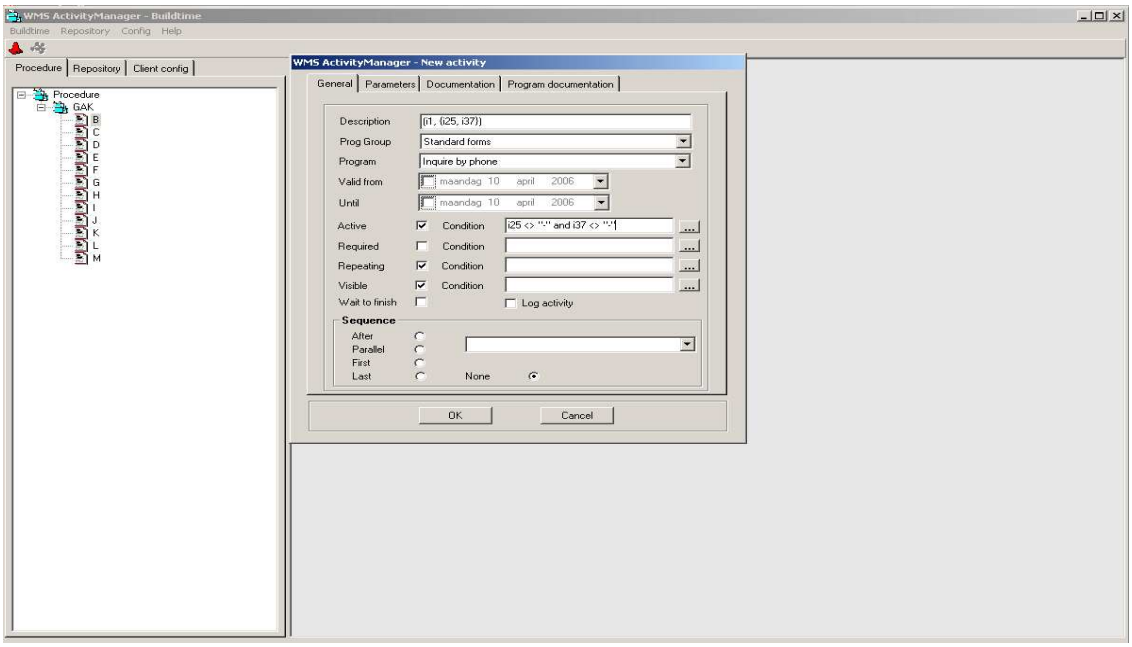

Fig. 11. Next, this database is imported in the build time component of Activity Manager, resulting in a list of activities on the left-hand side. The content of each activity can now be further specified. This screenshot shows the definition of a new operation for activity B. Note that an operation is called an activity in Activity Manager (see also Table 1).

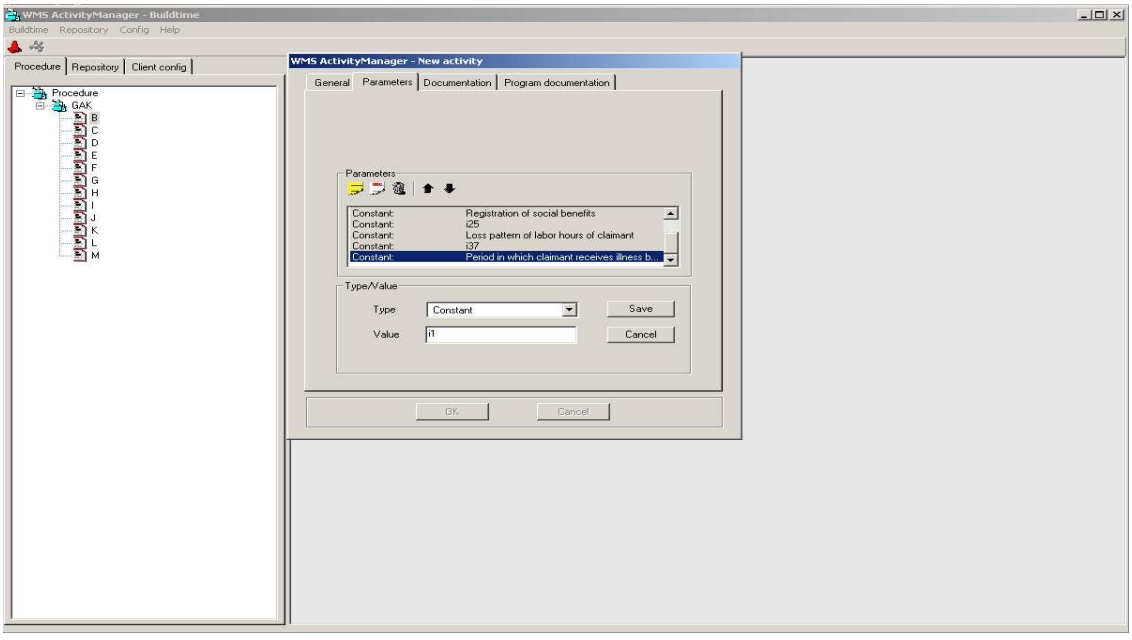

Fig. 12. Data elements are defined as constants in an operation, like $i1$ is defined here after the definition of $i25$ and $i37$.

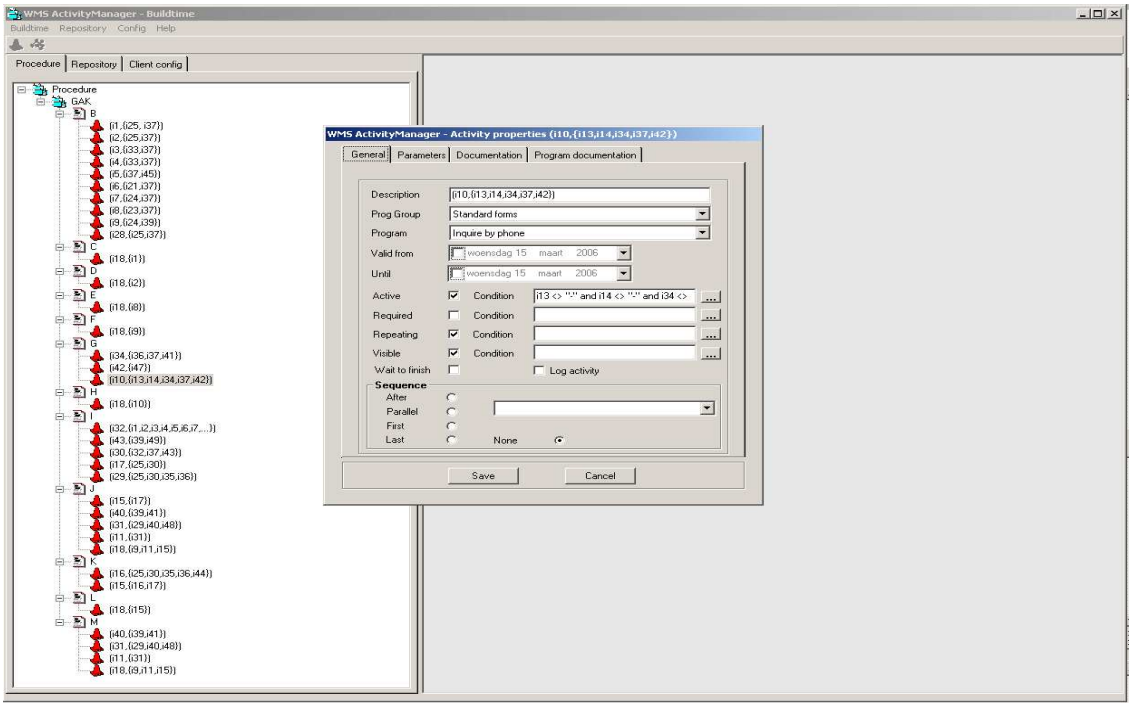

Fig. 13. After defining the operations for all activities from the process model (see left-hand side of figure), the conditions for execution can be added. Such a condition property makes it possible to give hierarchical dependencies to operations. In the case shown in this picture, the operation can only be executed when data elements $i13$, $i14$, $i34$, $i37$, and $i42$ are determined. To obtain $i34$ and $i42$ the other two operations in activity G have to be performed first. The other data elements are provided at the start of the case by the applicant.

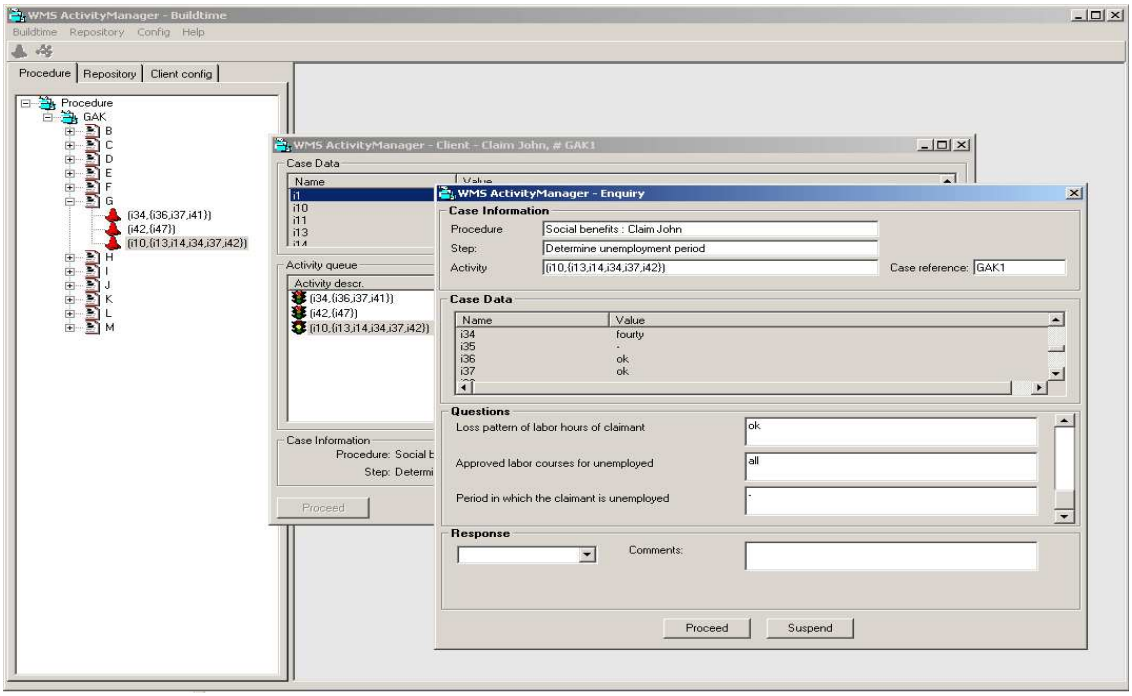

Fig. 14. In the runtime environment of Activity Manager, a form for every operation is shown to the user. This screenshot illustrates that the third operation of activity G only becomes available when the first and second operation are performed. This is according to the condition we defined in the previous step.

Acknowledgement

This research is supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Dutch Ministry of Economic Affairs. We gratefully acknowledge the technical assistance from Pallas Athena and Gyata BPI.

References

1. W.M.P. van der Aalst. On the automatic generation of workflow processes based on product structures. *Computers in Industry*, 39:97–111, 1999.
2. W.M.P. van der Aalst and P.J.S. Berens. Beyond Workflow Management: Product-Driven Case Handling. In S. Ellis, T. Rodden, and I. Zigurs, editors, *International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001)*, pages 42–51. ACM Press, New York, 2001.
3. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
4. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
5. W.M.P. van der Aalst and S. Jablonski. Dealing with Workflow Change: Identification of Issues and Solutions. *International Journal of Computer Systems, Science, and Engineering*, 15(5):267–276, 2000.
6. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
7. W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.
8. M. Adams, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Facilitating Flexibility and Dynamic Exception Handling in Workflows. In O. Belo, J. Eder, O. Pastor, and J. Falcao e Cunha, editors, *Proceedings of the CAiSE'05 Forum*, pages 45–50. FEUP, Porto, Portugal, 2005.
9. A. Agostini and G. De Michelis. Improving Flexibility of Workflow Management Systems. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 218–234. Springer-Verlag, Berlin, 2000.
10. F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow Evolution. In *Proceedings of ER '96*, pages 438–455, Cottubus, Germany, Oct 1996.
11. T.H. Davenport. *Process innovation : reengineering work through information technology*. Harvard Business School Press, Boston, 1993.
12. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.
13. C.A. Ellis and K. Keddara. A Workflow Change Is a Workflow. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 201–217. Springer-Verlag, Berlin, 2000.
14. C.W. Guenther and Wil M.P. van der Aalst. Mining Activity Clusters From Low-level Event Logs. BPM report, http://is.tm.tue.nl/staff/wvdaalst/BPMcenter/reports.htm, 2006.
15. GYATA BPi. GYATA BPi website. http://www.gyatabpi.com, accessed on November 18, 2005.
16. M. Hammer and J. Champy. *Reengineering the corporation*. Nicolas Brealey Publishing, London, 1993.
17. T. Herrmann, M. Hoffmann, K.U. Loser, and K. Moysich. Semistructured models are surprisingly useful for user-centered design. In G. De Michelis, A. Giboin, L. Karsenty, and R. Dieng, editors, *Designing Cooperative Systems (Coop 2000)*, pages 159–174. IOS Press, Amsterdam, 2000.
18. K. Kaan, H.A. Reijers, and P. van der Molen. Introducing Case Management: Opening Workflow Managements Black Box. In *Proceedings of the 4th International Conference on Business Process Management (BPM 2006)*, 2006.
19. M. Klein, C. Dellarocas, and A. Bernstein, editors. *Proc. of the CSCW-98 Workshop Towards Adaptive Workflow Systems*, Seattle, Washington, November 1998.
20. M. Klein, C. Dellarocas, and A. Bernstein, editors. *Adaptive Workflow Systems*, volume 9 of *Special issue of the journal of Computer Supported Cooperative Work*, 2000.
21. J.A. Orlicky. Structuring the bill of materials for mrp. *Production and Inventory Management*, pages 19–42, Dec 1972.

22. Pallas Athena. Pallas Athena website. http://www.pallasathena.nl, accessed on November 18, 2005.
23. E.A.H. Platier. *A logistical view on business processes: BPR and WFM concepts (in Dutch)*. PhD thesis, Eindhoven University of Technology, Eindhoven, 1996.
24. M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
25. H.A. Reijers. *Design and Control of Workflow Processes: Business Process Management for the Service Industry*, volume 2617 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2003.
26. H.A. Reijers, S. Limam, and W.M.P. van der Aalst. Product-based Workflow Design. *Journal of Management Information systems*, 20(1):229–262, 2003.
27. H.A. Reijers and I.T.P. Vanderfeesten. Cohesion and Coupling Metrics for Workflow Process Design. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 290–305. Springer-Verlag, Berlin, 2004.
28. S. Rinderle, M. Reichert, and P. Dadam. Correctness Criteria For Dynamic Changes in Workflow Systems: A Survey. *Data and Knowledge Engineering*, 50(1):9–34, 2004.
29. Staffware. *Staffware Process Suite Version 2 – White Paper*. Staffware PLC, Maidenhead, UK, 2003.
30. S.X. Sun and J.L. Zhao. A Data Flow Approach to Workflow Design. In *Proceedings of the 14th Workshop on Information Technology and Systems (WITS'04)*, pages 80–85, 2004.
31. I. Vanderfeesten, W.M.P. van der Aalst, and H.A. Reijers. Modelling a Product Based Workflow System in CPN tools. In K. Jensen, editor, *Proceedings of the Sixth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2005)*, volume 576 of *DAIMI*, pages 99–118, Aarhus, Denmark, October 2005. University of Aarhus.
32. M. Weske. Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System. In R. Sprague, editor, *Proceedings of the Thirty-Fourth Annual Hawaii International Conference on System Science (HICSS-34)*. IEEE Computer Society Press, Los Alamitos, California, 2001.