Adaptation Algorithm for Navigation Support in User Adaptive Enterprise Application

Inese Supulniece
Institute of Information Technology, Riga Technical University
Kalku 1, Riga, Latvia, LV-1658
Inese.Supulniece@rtu.lv

***Abstract*—User Adaptive Enterprise Application supports users in identification of more efficient variations of business process executions. It is the set of adaptive components to be added to standard enterprise application. Adaptive navigation support is one of identified components with the aim to help users execute routine activities faster, reduce amount of mistakes and support new users of the system. The paper presents a meta-model, architecture and adaptation algorithm behind the adaptive navigation support. Business process constraints are used to describe business rules and restrictions. Process execution patterns are used to discover characteristics and preferences of individual users. The proposed algorithm is evaluated using simplified sales process simulation in Microsoft Dynamics AX and task management process simulation. The results of the early evaluation show that adaptive navigation component supports business rules and individual variations of business process execution. It also indicated some limitations of applying business process constraints on user interface level.**

Keywords-user adaptive system; enterprise application; adaptation algorithm; recommendation.

I. INTRODUCTION

Today, in rapidly changing environment, business processes are dynamic [1]. The need to adapt a process has been a topic of interest in the recent years [1]. Enterprise applications are used to execute business processes. Usually, these are packaged applications providing standardized implementations of business processes. Users of enterprise applications either use predefined workflows or rely on user documentation and best practices to execute their business processes [2]. Besides these standard capabilities, in many cases, users also can use other functions provided by enterprise applications subject to their access rights. That means that users have possibilities to introduce their own variations in process execution. By considering these variations, users might come up with more efficient ways of executing business processes [3]. If an enterprise application supports users in identification of more efficient variations of business process execution and enables for continuous execution refinement it is referred as to as User Adaptive Enterprise Application (UAEA) [4].

There exist various approaches, on how to manage business process variants without violating organisational rules. One of them is description of business processes, using process constraints [5]. Business process constraints can express minimal restrictions on the selection and ordering of tasks of the targeted business process, thus providing a degree of flexibility in process execution. Constraint-based models are considered to be more flexible than traditional models because of their semantics: everything that does not violate constraints is allowed [5].

The objective of this paper is to present the meta-model, architecture and adaptation algorithm behind Adaptive Navigation Support (ANS) of UAEA. This component is using 1) business process constraints to keep main rules of the processes under control while allowing different business process execution variants; and 2) task execution patterns to manage individual user oriented process variants.

The rest of paper is structured as follows: Section 2 provides brief introduction to UAEA. Section 3 presents adaptation constraints for business process variants. The ANS component is explored in Section 4. The paper concludes with Section 5, where conclusion and further research are discussed.

II. USER ADAPTIVE ENTERPRISE APPLICATION

There are multiple ways the enterprise applications could be adapted, e.g. [1], [6], [7]. In the context of UAEA, the adaptation engine generates the user-oriented view of business processes in the enterprise application. Given that ERP systems are mainly used for repetitive tasks [8], the user-oriented process adaptation uses previously observed users' behavior to optimize performance of business activities. [6] and [7] discusses the same problems and similar approaches for solving them, however, architecture and logics differ per each research (also for this paper). Each of proposals has its own motivating business case, benefits and restrictions, thus it is rather impossible to compare their effectiveness. For example, the adaptation mechanism in [6] applies two data sets: the process model, which describes business rules and the sequence graph, which comprises nodes representing the individual process steps. An directed edge between two nodes A and B of the sequence graph describes a temporal sequence that process step B follows immediately after A and edge value represents the likehood of following a particular path through the process. Our adaptation mechanism uses business process constraints to describe business rules and an ordinary sequence to keep individual process execution variants. The choise between usage of the full business process model (as in [6]) or business process constraints (as in this paper) depends on the flexibility degree of the process.

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-219-6

The idea of the UAEA and main characteristics are described in [4], where the model of UAEA is elaborated. The overall goal of the UAEA is to identify possibilities of existing enterprise applications to raise performance efficiency (see Figure 1). Related operational goals are: optimization of routine activities, preventing mistakes, decreasing the learning time for new processes and for new employees. Technically, system should optimize routine activities, prevent mistakes and support non-routine activities. This is measured in process execution time and amount of mistakes.

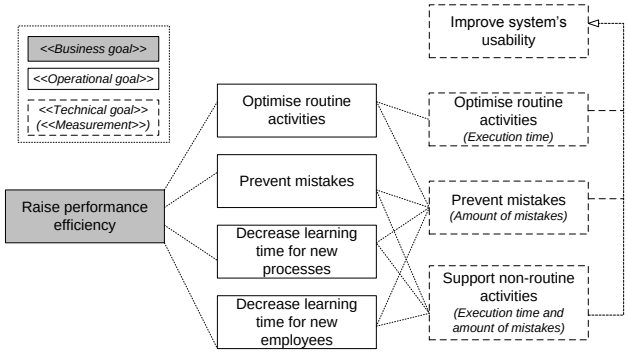


Figure 1. The goal model of the UAEA.

UAEA is the set of six adaptive components to be added to a standard enterprise application:

Adaptive process execution overview shows full process or part of the process, current activity and possible paths to finish the process. The aim of this recommendation is to provide local or global guidance for user, especially for non-routine activities.

Adaptive navigation support (ANS) presents recommendation block with recommended navigation item, mandatory and prohibited activities for particular process. The aim of this recommendation is to help user execute activities faster, to reduce amount of mistakes and support new users of the system.

Adaptive information support recommends related documents, systems or data based on local or global patterns.

Adaptive decision support recommends possible decisions based on local or global decision patterns.

Adaptive problem preventing presents most common problems and solutions related to current activity. It prevents possible mistakes for non-routine activities or new users.

Adaptive error and exception handling notifies user about incompleteness in process execution, e.g., missed activity or not finished process.

Idea of the ANS for the UAEA lies in the following observation [9]: users use enterprise application to accomplish their tasks, usually consisting of multiple steps; each user or user group has a preferred sequence of the steps (task execution patterns). UAEA attempts to exploit such usage patterns with the aim to improve performance efficiency.

This paper explores a meta-model, architecture and adaptation algorithm behind ANS component.

III. ADAPTATION CONSTRAINTS FOR BUSINESS PROCESS VARIANTS

In large enterprises, it can be observed that a common business processes exists in many variations across different parts of the organisation [10]. When supporting business processes there is a difficult trade-off to be made between control and flexibility [5].

Control is achieved with restrictions for the process adaptation, which are modeled as rules or constraints. Business process constraints are suitable for supporting flexible processes that allow many different executions [5]. Most theoretical process modeling languages, such as Petri Nets, process algebras, BPMN, UML and EPCs define direct causal relationships between activities in process models. Opposed to this, constraint-based languages are of a less procedural nature and use a more declarative style [5]. Declarative languages are more flexible by nature, and it is more likely that users working in such an environment need support, e.g. recommendations [7].

There have been proposed a number of constraint languages in various disciplines, e.g., ConDec [11], Object Constraint Language [12], MiniZinc [13]. These are extensive approaches; consequently they require specific knowledge and complex algorithms for run-time process adaptation based on available constraints.

Lu et al. [14] presents how task selection constraints can be specified at design time, through selection constraints. This approach was adapted for the ANS, because it is unsophisticated, efforts for managing and using the business process constraints should be kept minimal and seems to be promising approach for combining process constraints and task executions patterns in adaptation algorithm.

In [14], the following classes of selection constraints have been identified:

(1) Mandatory constraint *man* defines a set of tasks that *must be executed* in every process variant, in order to guarantee that intended process goals will be met.
(2) Prohibitive constraint *pro* defines a set of tasks that *should not be executed* in any process variant.
(3) Cardinality constraint specifies the minimal *minselect* and maximal *maxselect* cardinality for selection among the set of available tasks.
(4) Inclusion constraint *inc* expresses the dependency between two tasks T_x and T_y, such that the of T_x imposes restriction that T_y must also be included. Prerequisite constraint *pre* is the inverse of an inclusion constraint.
(5) Exclusion constraint *exc* prohibits T_y from being included in the process variant when the T_x is selected.
(6) Substitution constraint *sub* defines that if T_x is not selected, then T_y must be selected to compensate the absence of the former.
(7) Corequisite constraint *cor* expresses a stronger restriction in that either both T_x and T_y are selected, or none of them can be selected, i.e., it is not possible to select one task without the other.

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-219-6

(8) Exclusive-Choice constraint *xco* is also a more restrictive constraint on the selection of alternative tasks, which requires at most one task to be selected from a pair of tasks (T_x, T_y).

The mentioned classes of selection constraints are re-used in the ANS component.

IV. ADAPTIVE NAVIGATION SUPPORT

The main goal of the ANS component is to optimize routine activities, prevent mistakes and also support new users during non-routine activities. The changing object for this component is user and task. It means that adaptation result differs per user and takes into account situational aspects.

Figure 2 presents a meta-model of the ANS component, which illustrates the main concepts used by the adaptation algorithm. An enterprise application consists of *user interface (UI) elements*, which are mapped to the *activities* of the process. Capturing the activities, which are not related to the control UI elements of the application, is out of the scope of this research and proposed adaptation algorithm. Each UI element belongs to some UI *form* or *window*. *Constraint* consists of two activities. Constraints are defined separately for each form/window. *Process execution pattern* comprises activities representing the actually executed process steps. It consists of two or more activities and it is related to the *user*, who executed the particular pattern. Each pattern has the attribute – *frequency of execution* – how many times the pattern was executed. The set of *global patterns* include all execution patterns, despite the user, who created it. The set of *local patterns* include only those patterns, which were executed by the particular user.

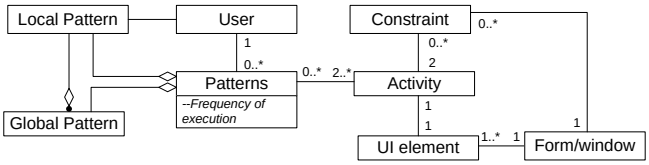


Figure 2. The meta-model of the ANS component.

The architecture of the ANS component is illustrated in Figure 3. It consists of data bases (event logs); repositories (users, constraints, activities, execution patterns); engine for the adaptation algorithm and the user interface of the ANS. Types of business rules or constraints are adapted from [14] and are available in the form:

<form/window>,<constraint_type>{Tx, Ty}.

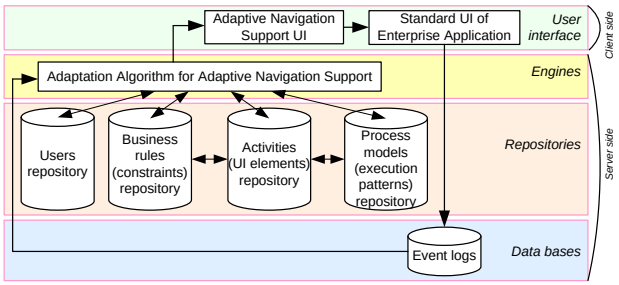


Figure 3. The architecture of the ANS component.

Process models or execution patterns are saved as the sequences of activities $a_1, a_2, \ldots a_n$. All activities executed by each individual user are perceived and stored as business process patterns.

A. Description of the adaptation algorithm

The current activity, process execution patterns (individual and global) and business process constraints forms the input to the adaptation algorithm (see Figure 4). The main output is recommended next step.

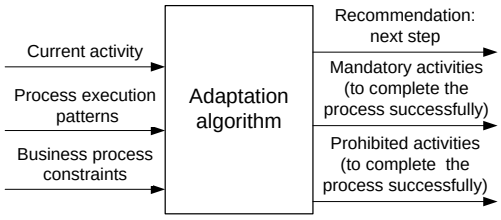


Figure 4. Input/ouput view of the adaptation algorithm.

To realize the adaptation algorithm of the ANS, the following data sets are introduced: *1) M* – consists of activities $M_1, M_2, \ldots, M_k$, which are mandatory; *2) E* – consists of activities $E_1, E_2, \ldots E_u$, which are prohibited; 3) *I* – consists of executed activities $I_1, I_2, \ldots I_p$. All mentioned data sets are sequences.

Figure 5 presents simplified view of the adaptation algorithm behind the ANS. Firstly, the system reads the activity *A* performed by the user, identifies the form F_o and selects all constraints, which include activity *A*. When the user executes any activity inside some form/window F_o, then all activities from the set of constraints $\{F_o, man\{T_i\}\}$ are automatically added to the set *M* and all activities from the

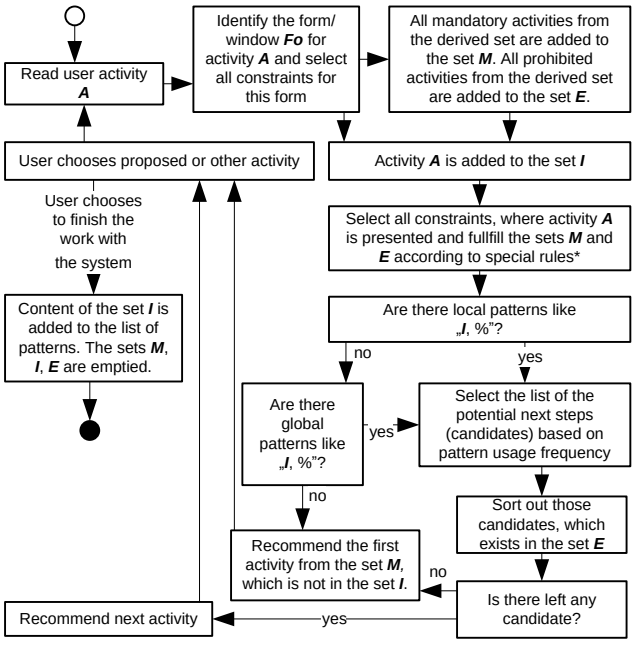


Figure 5. Simplified view of the adaptation algorithm behind the ANS.

TABLE I. THE LIST OF SPECIAL RULES

When any element H is added to the set M, then following rules should be verified:	
If there exists constraint	***Then execute following action***
$\{F_o, inc(H, Ty_i)\}$	Ty_i *is added to the set M*
$\{F_o, exc(H, Ty_i)\}$	Ty_i *is added to the set E*
$\{F_o, pre(Tx_i, H)\}$	Tx_i *is added to the set M*
$\{F_o, cor(H, Ty_i)\}$	Ty_i *is added to the set M*
When any element H is added to the set E, then following rules should be verified:	
If there exists constraint	***Then execute following action***
$\{F_o, sub(H, Ty_i)\}$	Ty_i *is added to the set M*
$\{F_o, cor(H, Ty_i)\}$	Ty_i *is added to the set E*

set $\{F_o, pro\{T_i\}\}$ are automatically added to the set *E*. But activity *A* is added to the set *I*.

Secondly, all constraints (including activity *A*) are reviewed by the system using special rules and the sets M and E are supplemented. For example, if there exists constraint $\{F_o, inc(A, Ty_i)\}$, then both activities *A* and Ty_i must be executed together. Consequently *Tyi* is added to the set of mandatory activities *M*. The special rules are listed in Table 1.

Further the list of local patterns is identified. If there are not local patterns, then system looks for global patterns. The list of the potential next steps is prepared according to the pattern usage frequency. If candidate exists in the set E, then it is removed from the list of the potential next steps. The system recommends the candidate with the highest pattern usage frequency index. If there are no candidates, then system recommends the first element from the set M, which is not executed yet. Next, it is up to user to utilize or ignore the recommendation.

B. Initial testing of the algorithm

The aim of initial testing was to prove: (1) if constraint types from [13] can be applied on user interface level and (2) if logic of the algorithm provides expected results.

	Recommendation (Adaptive Navigation Support)
STANDARD FUNCTIONALITY	Recommended step: Form: Activity (form will be opened) *Click here for all recommended steps*
	START NEW TASK/PROCESS CANCEL EXECUTED TASK/PROCESS
	Mandatory activities: Form: Activity (form will be opened) Form: Activity (form will be opened) Form: Activity (form will be opened) Form: Activity (form will be opened) ...
	Prohibited activities: Form: Activity (form will be opened) Form: Activity (form will be opened) Form: Activity (form will be opened) ...
	Executed activities: Form: Activity (form will be opened) Form: Activity (form will be opened) Form: Activity (form will be opened) ...

Figure 6. User interface prototype of the ANS.

Testing was performed as 1) simplified task management process simulation and 2) simplified sales process simulation in Microsoft Dynamics AX [15] Sales module and system proposed recommendations according to the user interface prototype, which is presented in Figure 6.

At the current stage of the research the main idea of testing was to perform initial validation of logics. Usability, performance and effectiveness testing is planned in nearest future.

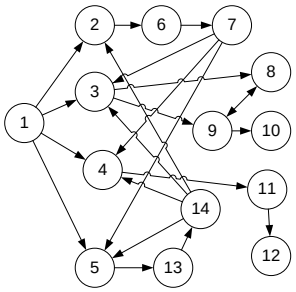
The results of the testing indicated limitations and problems of applying mentioned constraints on user interface level.

1) Task management process support

Before the testing, the following preparation works were done:

- Business process constraints were transferred to user interface level – see Table II.
- Four different process execution alternatives were stored in execution patterns repository – see Table III.

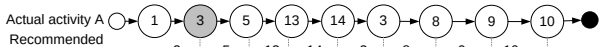
In the task management process, a user selects new or existing task. The task can be completed, forwarded, closed and/or supplemented with additional information. The possible process execution alternatives are illustrated in Figure 7.



No	Form	Activity
1	Main form	Proceed the task
2	New/existing task	Complete the task
3	New/existing task	Forward
4	New/existing task	Close the task
5	New/existing task	Additional information
6	Solution form	Add comment
7	Solution form	Save
8	Forwarding form	Select department
9	Forwarding form	Select employee
10	Forwarding form	Save
11	Closing form	Add comment
12	Closing form	Save
13	Additional information	Add comment
14	Additional information	Save

Figure 7. Task management: variations in process execution.

One variant of the process execution was simulated during the initial testing. The simulated process included nine activities. Seven activities corresponded to system recommendations and one recommendation failed. Testing report is available in Figure 8.



Recommended activity	2	5	13	14	3	8	9	10	-
M	1	1,5	1,5	1,5,13	1,5, 13,14	1,5,13, 14,3	1,5,13, 14,3,8	1,5,13, 14,3,8,9	1,5,13,14, 3,8,9,10
E	-	-	-	-	-	2,4	2,4	2,4	2,4
I	1	1	1,5	1,5,13	1,5, 13,14	1,5,13, 14,3	1,5,13, 14,3,8	1,5,13, 14,3,8,9	1,5,13,14, 3,8,9,10

Figure 8. Task management: testing results.

2) Sales process support

The three alternatives of the basic sales process activities were executed during the testing. The first alternative was linked to the user 1 – very careful person, who verifies the data before doing any action, e.g., prove the stock before adding the product to the offer. Second alternative was linked to the user 2 - person, who trusts the system and does only basic steps. Third alternative was linked to the user 3.

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-219-6

TABLE II. BUSINESS CONSTRAINTS

Type	*Form*	*Activities*
EXC	*New/existing task*	*3,2*
EXC	*New/existing task*	*3,4*
PRE	*New/existing task*	*5,3*
PRE	*Forwarding form*	*8,9*
PRE	*Additional information*	*13,14*
PRE	*Closing form*	*11,12*
PRE	*Solution form*	*6,7*

TABLE III. BUSINESS PROCESS PATTERNS

Pattern	*Usage frequency*	*User*
1,2,6,7	*5*	*1*
1,3,8,9,10	*2*	*1*
1,4,11,12	*3*	*1*
1,5,13,14,3,8,9,10	*1*	*1*

These alternatives were stored as process execution patterns and business process constraints were transferred to user interface level, e.g., delivery address and currency is mandatory information.

C. Limitations

The main problem is related to usage of constraints, because originally constraints in this form were developed for description of business processes. Constraints in current form define only relations between every 2 activities. For example, none of described constraints allows specifying the following rule: if *client is selected*, then afterwards it is mandatory to *Save* the form OR *Cancel* the form. One option would be to write this rule as *inc*{(client is selected), *xco*{Save, Cancel}}. But this requires more sophisticated algorithm, which might end with performance issues on real life system and data amount.

Another problem is related to user interface design of described component. How to track read-only fields; when user uses it; when they stop to be relevant to particular activity?

Consequently, currently design of Adaptive Navigation Support component recommends only next executable activity and opens the full form, where it is located.

V. CONCLUSION AND FUTURE RESEARCH

This paper presented a meta-model, architecture and adaptation algorithm behind adaptive navigation support component in user-adaptive enterprise application. Business process constraints are used to describe business rules and restrictions. Process execution patterns are used to discover characteristics and preferences of individual users.

Important problems are identified at current stage, e.g., limitations of existing form of defining the constraints. Now the aim is to develop an interactive prototype of the Adaptive Navigation Support component and test usability, effectiveness and performance by real users.

Also valuable ideas rose during the research, e.g. differentiation between mandatory and optional constraints as suggested by [5].

ACKNOWLEDGMENT

This work has been supported by the European Social Fund within the project «Support for the implementation of doctoral studies at Riga Technical University».

REFERENCES

[1] G. Hermosillo, L. Seinturier, L. Duchien, „Using Complex Event Processing for Dynamic Business Process Adaptation", In Proc. of the 7th IEEE 2010 International Conference on Services Computing, 2010. DOI : 10.1109/SCC.2010.48

[2] T.A. Curran, A. Ladd, "SAP R/3 Business Blueprint: Understanding Enterprise Supply Chain", Prentice Hall PTR, Upper Saddle River, 2000.

[3] H. Topi, W. Lucas, T. Babaian, "Identifying usability issues with an ERP implementation", In Proc. of ICEIS 2005, pp. 128-133.

[4] I.Supulniece, J.Grabis, "Modeling of user adaptive enterprise applications", In Proc. of ICEIS 2012, in press.

[5] M. Pesic, M.H. Schonenberg, N. Sidorova, W.M.P. van der Aalst, "Constraint based workflow models: Change made easy". In Proc. of OTM Confederated International Conferences 2007, 2007, pp. 77-94.

[6] C. Dorn, T. Burkhart, D. Werth, S. Dustdar, „Self-adjusting recommendations for peopledriven ad-hoc processes", In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, Springer, Heidelberg, 2010, pp. 327–342.

[7] B. Weber, B.F. van Dongen, M. Pesic, C.W. Guenther, W.M.P. van der Aalst, „Supporting flexible processes through recommendations based on history", Eindhoven University of Technology Eindhoven, BETA Working Paper Series, 2007. ISBN: 978-90-386-1038-2.

[8] H. Klaus, M. Rosemann, G.G. Gable, "What is ERP?" Information Systems Frontiers 2:2, 2000, pp. 141-162.

[9] I. Supulniece, J. Grabis, "Discovery of personalized information systems usage patterns", In Proceedings of ICIST, Kaunas, Lithuania, 2010, pp. 25-32.

[10] M. Weidlich and M. Weske. "Structural and behavioural commonalities of process variants". In Proc. of ZEUS'10, Berlin, Germany, CEUR vol.563, 2010, pp. 41-48, CEUR-WS.org

[11] M. Pesic and W.M.P. van der Aalst, "A declarative approach for flexible business processes". In J.Eder and S.Dustdar, (eds.), Business Process Management Workshops, Workshop on Dynamic Process Management (DPM 2006),LNCS, vol. 4103, Springer-Verlag, Berlin, 2006, pp. 169-180.

[12] Object Management Group, "Object constraint language specification version 2.3.1". OMG, 2012

[13] N. Nethercote, P.J. Stuckey, R. Becket, S. Brand, G.J. Duck and G. Tack, "MiniZinc: Towards a standard CP modelling language". In CP, LNCS 4741, 2007, pp. 529–543

[14] R. Lu, S. Sadiq, G. Governatori, X.Yang, "Defining adaptation constraints for business process variants", In Proc. of BIS, 2009, pp. 145-156

[15] Microsoft Dynamics AX, Retrieved from http://www.microsoft.com/en-us/dynamics/erp-ax-overview.aspx

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-219-6