Design and Simulation of Electronic Service Business Process

Peteris Stipravietis, Edzus Zeiris, Maris Ziema
ZZ Dats, SIA
41/43 Elizabetes street, Riga, Latvia, LV-1010
[peteris.stipravietis,edzus.zeiris,maris.ziema]@zzdats.lv

***Abstract* — The paper discusses the identification of common business process design-time problems using Yet Another Workflow Language (YAWL). The approach proposed by the authors is based on the creation of business process in the YAWL environment in order to simulate the process model which could resolve some of the design-time problems, i.e., possible bottlenecks, as well as provide hints on how to correct initial process. The simulation is done using process mining software "ProM Framework" and the Colored Petri nets simulation and analysis framework "CPN Tools". The process design with YAWL is done with respect to Business Process Execution Language (BPEL) requirements, thus later allowing the transformation from YAWL to BPEL via the intermediate structure. The examples show that it is possible to identify some of the possible faults of the process using the proposed approach.**

Keywords – Electronic service; YAWL; simulation.

I. INTRODUCTION

E-services are common in information society nowadays, and even though they tend to become more and more accessible and varied, the problems that occur during the design phase of the service remain the same. These problems include, for example, questions on how to facilitate the creation of business process to the user with no specific programming skills, how to define the process in a way that creates the process description abstract but accurate enough at the same time, how to check the created model – to determine the weaknesses, perform the measurements based tuning, and others.

The validation of the process is even more important when the process being changed is already deployed and used in production environment – one must make sure that the changes are implemented correctly, that the new instances of the process can run together with old instances already running. The implementation and validation of changed process also have to be simple and cost-effective enough – hence the conclusion that the solutions of these problems rely heavily on the choice of the language used to describe the process – does it provide the possibilities to validate and simulate the process.

Existing business process modeling languages can be divided in two groups. The languages of the first group are favored by the academic community, but rarely used in real-life solutions. These languages are based on Petri nets, process algebra; they have formal semantics, which allow the validation of the models described by these languages. The languages of the second group are used in real-life projects much more than in academic researches. Business Process Execution Language (BPEL) [1] and Web Services Flow Language (WSFL) [2] are among these languages. These, so called business languages, often lack proper semantics, which could lead to debate on how to interpret the business models described by these languages. The availability of different implementations of these languages from different vendors does not facilitate the situation either, yet they are used much more, compared to rarely used models described by academic languages. If a situation arises when business process model described by business language needs to be validated using Petri nets, one must either abandon the validation or transform the process model to another model, described in academic language, for example Yet Another Workflow Language (YAWL) [3][4][5]. The authors propose reverse approach – first, a process is created using academic language. The design problems of the process model can then be solved by mathematical means. Second, the verified and updated model is transformed to model described in business language. The advantages of the approach described follows:

- If a model is created using academic language, it is more readable and maintainable than the model, which is a transformation result itself. It is also easier to perform analysis of untransformed model, because the transformation could lose some design information.
- Model, transformed to business language, is already validated and ready to be executed. Of course, the model must be double-checked to make sure if it needs any corrections. The alternatives of the execution environment for the model are much more than the environments for academic languages; in addition to that, they have superior technical support.

The purpose of this paper is to examine the design and simulation stages of the aforementioned approach – can it be used during the design of simple e-service business project; and to check if it helps to identify and resolve most common design-time problems.

The rest of the paper is structured as follows – Section II provides an overview of the proposed design approach. Next, Section III defines the design-time restrictions of the YAWL workflow which must be observed. The simulation phase of the approach is discussed in Section IV, while Section V shows the practical example of the simulation. Finally, Section VI contains conclusion.

II. PROPOSED APPROACH

The approach proposed by authors consists of five consecutive steps – the design, the simulation, the transformation to 'protostructure' – simplified, yet fully descriptive notation of the business process control flow –, the optimization based on quality attributes of the service and the transformation to BPEL. Some of these steps may be omitted or repeated as necessary, as shown in Fig. 1 – solid lines show the most common path, dashed lines show alternative paths of execution, lighter boxes represent steps which may be omitted.

A similar approach is proposed by authors of [6], but their solution is based on straightforward conversion of YAWL workflow to BPEL process (straight to step 5 from step 1). While the straightforward transformation is more efficient in terms of development cost and time, the authors' approach includes simulation and optimization steps which should reduce the costs of maintenance later on.

The first step is the design of the business process using academic language. The initial business process model is created during this phase. Authors use the YAWL as the language of choice – it is based on Extended Workflow Nets (EWF), the workflows described in YAWL can be transformed to colored Petri nets to perform simulations and formal semantic validation. It also supports all workflow patterns [7] – parallel flow, branching, synchronizations and others. Although YAWL supports all the patterns, the business process model must take into account that the process will later be transformed to BPEL – as such, it may not contain patterns which do not translate to BPEL directly or using non-solution specific workarounds.

The second step of the approach is the simulation of the business process model. The analysis of the created business process is very important part of the design – one needs to find bottlenecks, when instance of the process or its part could use up all available resources, thus forcing other instances to wait for these resources; identify dead ends, which could lead to infinite loops and never ending process instances; find deadlocks, when process querying for the same resources effectively block each other; define fault handling and cancellation activities, which cancel all the work done by previous activities.

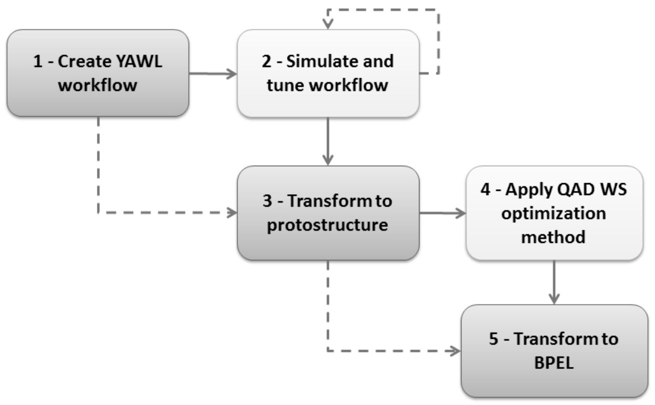
1 - Create YAWL workflow
2 - Simulate and tune workflow
3 - Transform to protostructure
4 - Apply QAD WS optimization method
5 - Transform to BPEL

Figure 1. Proposed approach, step by step.

During this step it is also possible to identify reusable structures, for example, audit log activities. Such challenges usually are solved with the simulation.

The third step provides the transformation to primitive structure. Primitive structure is simplified definition of business process control flow, although it can also be used to maintain the data flow. The primitive structure serves as an intermediate between academic and business languages and can be used to create processes described by multiple languages, not only BPEL. The primitive structure may be changed and improved during this phase to facilitate the transition to target language, i.e., restructure its control flow in a way that it becomes well-formed and contains only patterns supported by BPEL.

The fourth step provides the segmentation of primitive structure using the Quality Attributes Driven Web Services Design (QAD WS) method which offers the segmentation of business process, represented as oriented graph. The segmentation result depends on process quality attributes selected by designer and their respective values. The result of this method is Pareto optimality set – the method returns the most suitable segmentations from all possible considering the quality attributes given.

The business language selected by authors and used in their proposed approach is BPEL, and using of QAD WS method on primitive structure would provide the possible structures of BPEL process – which parts of the process would belong to orchestration and which ones would be implemented as web service calls. The work in [8] also discusses the partitioning of Web services into orchestrations based on their Quality of Service (QoS) values, although that approach do not use multicriterial optimization, but rather is based on Petri nets and usage of statistical data.

The QAD WS method perceives the business process as an oriented graph *G*, whose vertices corresponds to process activities, but edges between them represents the control flow. Using various quality attributes and the structure of graph *G*, QAD WS method solves multi-criteria optimization task, which results in the segmentation set of initial graph *G*: *G'=QAD(G)*. Criteria used by the method are:

- Costs of development *C*;
- Performance *T*;
- Maintenance costs *E*;
- Reusability *R*;
- Integrity *I* [9].

The segmentation set *G'* consists of N most optimal solutions designer can choose from – this method may greatly reduce possible solutions of process architecture, thus aiding the designer. For instance, if the main criterion for segmentation is performance, then the segmented graph should contain the Web service invokes as few as possible, because every invoke adds to total execution time. On the other hand, the reusability will lead to much more segmented graph to allow the components to be reused. Preliminary tests show that if a certain graph G consists of 11 vertices, then, taking into account the segmentation restrictions (mainly to preserve control- flow), one ends up with 1015 possible solutions. Applying the QAD WS method reduces the count of possible solutions to 8 solutions. The total solution count varies depending on the structure of the graph, and resulting set solution count varies depending on the used criteria and their respective weights.

The result of the 3rd step of authors proposed approach is primitive structure – oriented graph *P* that corresponds to initial YAWL workflow, which could be used as an input graph *G* for QAD WS. Authors also note that *P* is more complicated and restrictive as *G* – in addition to process activities and links between them it also contains the process control flow.

The last phase of proposed approach is the transformation of primitive structure to business language process which results in the business process defined in business language. This process is not ready to be executed, but its structure corresponds to initial process model described by academic language and maintains its process flow.

III. THE DESIGN OF THE BUSINESS PROCESS

To be able to transform the YAWL workflow successfully, the workflow must conform to some requirements. Firstly, it should not contain patterns, which have no analog constructions in BPEL, for example, the passing of process control to an activity residing outside the synchronized block, i.e. – goto-like construction. BPEL directly supports 13 patterns out of 20 [10], discussed in [11]. The parallel flow with runtime-only knowledge is also not supported.

Secondly, the incoming and outgoing messages are associated with specific process instance using correlation sets. YAWL lacks concept of correlation sets, because each workflow instance (case) is started by its clients (users), thus creating an instance in execution environment. This environment manages the workflows and offers to users corresponding options, based on the state of instance and its specification [4]. The variable which could be used as a correlation set variable must be created in the workflow or during the transformation and finally added to each defined data type used in BPEL messages.

Thirdly, support of human tasks – all BPEL activities related to exchange of information with process partners are perceived as web service operations, i.e., BPEL has no concept of "Human interaction". To fill this gap several BPEL extensions are proposed, for example, BPEL4People [12][13] – Organization for the Advancement of Structured Information Standards (OASIS) is working on standardizing this extension.

Last but not least, workflow definition must correctly define all the branching conditions to avoid cases when transformed BPEL process' While, Repeat/Until and If blocks contain incorrect values.

IV. THE SIMULATION

As already mentioned before, the developers and architects may encounter the same kind of problems during the development of the business process of electronic service – bottlenecks, dead ends, and fault handling and cancellation activities. Part of these problems may be identified using simulation.

The authors of the paper [14] propose simulation which uses process design data, historical data about executed process instances from audit logs and state data of the running process instances from the execution environment. Data from all three sources are combined to create simulation model – design data are used to define the structure of the simulation model, historical data define simulation parameters, state data are used to initialize the simulation model.

Altering the simulation model allows to simulate different situations, for example, to omit certain activities or divert the process flow to other execution channels. Taking into account the state data of running process instances, it is possible to render the state of the system in near future and use the information to make decisions regarding the underlying business process.

The simulation of the workflow is carried out using process data mining framework ProM [15]. To create simulation model, following steps are performed:

- Workflow design, organizational and audit log data are imported from execution environment;
- According to imported data a new YAWL workflow model is created and state data are added;
- The new model is converted to Petri net;
- Resulting Petri net is exported to simulation execution environment CPN Tools [16] as a colored Petri net.

CPN Tools environment provides the process simulation possibilities both in long and short-term, using the state data of chosen process instance. This technique differs from others with its degree of realism. For instance, the work [18] shows the so called mediator approach based on Discrete Event System Specification (DEVS) models, while the work [19] exploits event-based approach on Service-oriented Architecture (SOA) Both of these approaches, however, do not use the statistical data of already finished instances to take into account the availability of the resources; yet this method creates artificial delays based on historical data from audit logs and organizational model.

Returning to design-time problems, mentioned before – the authors now will evaluate the simulation approach, focusing on its capabilities to identify these problems.

A common mistake is to propose that the user of process will provide all data when prompted, for example, fill out all fields in a web form. Some fields could be left blank because user is not interested in sharing particular information (because of privacy concerns, etc.) or other reasons. If a process needs such information to continue, but user does not provide it, it enters in a waiting state and theoretically may never be finished. As an example, a simple YAWL workflow is provided, which expects the input of e-mail address and phone number from the user. The workflow contains AND-flow – it is finished when both branches are finished – see Fig. 2.

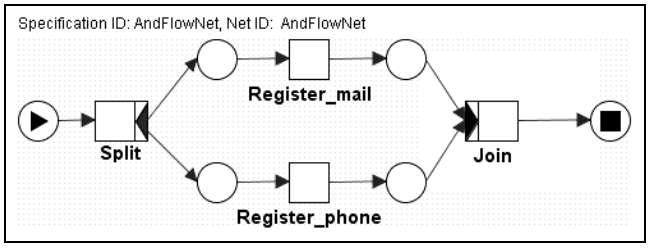

Figure 2. Simple YAWL workflow – parallel flow

The solution is to use OR-flow instead of AND, but the simulation approach discussed is not applicable – Petri nets does not support OR-flows; also the ProM Framework generates syntactically incorrect Standard ML (SML) file, if the net output variables have not been assigned value during the flow – however, it is not possible to assign initial values to them during the design. However, the problem can be identified by analyzing the process execution log, using, for example, ‘Basic Log Analysis’ module from ProM toolset. Fig. 3 clearly shows the difference between activities executed. Naturally, the question arises – why are there so many processes in waiting state and why do the users register their phone numbers far less than their e-mail addresses.

The simulation will not be helpful to identify the problem even after the editing of the SML file – the generated Petri net will contain AND-flow and the short-term simulation will direct the token through both branches. It is not reasonable to fix the net – the problem would be already identified and there would not be need to carry out the simulation once more. As mentioned before, such problem could be fixed introducing the OR-flow, however, both Petri net and BPEL lack the concept of OR-flow – it could be replaced by subsequent XOR-flows (IF-THEN branching).

Another task is to identify possible situations which could lead to infinite looping. The reason behind the looping mostly is incorrectly defined loop exit condition or loop variable does not have correct value assigned. This case can also be identified using ProM tool ‘Basic Log Analysis’ (activities within loop would be far more than others), but the simulation technique discussed can be used too.

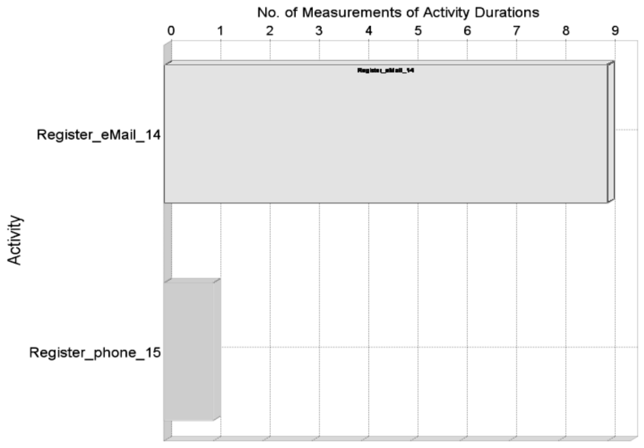

Figure 3. Difference of activities

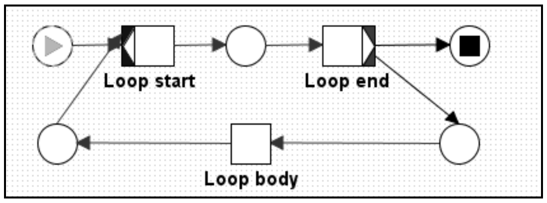

Figure 4. Simple YAWL workflow – possible infinite loop

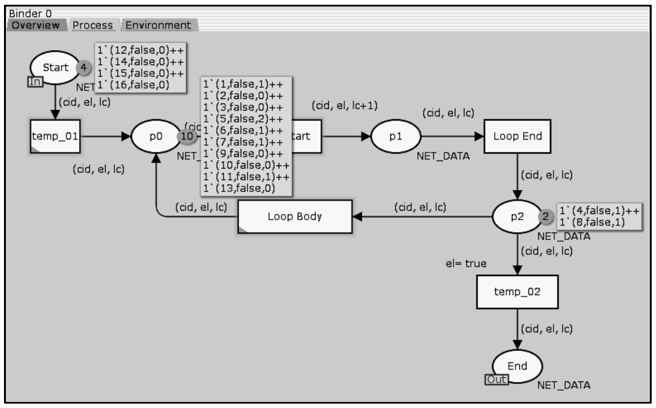

Figure 5. Colored Petri net – infinite loop

Fig. 4 shows the example of loop. The workflow has one local variable – “exitLoop” of Boolean data type, but its value intentionally is not being changed. Fig. 5 shows resulting Petri net – variables “loopCount” and “caseId” were added for demonstrative purposes (token colset NET_DATA is defined as product INT*BOOL*INT). Examining the net simulation, one observes that incoming tokens never leave loop.

The simulation performed to identify the bottlenecks produces similar results. YAWL has mechanism to distribute activities to resources defined in its organizational model – these resources are tokens of different colorset in a colored Petri net. A transition in Petri net may fire if all the places leading to transition have tokens – if a ‘resource’ place has no tokens, transition never fires, and all ‘data’ tokens accumulate in ‘data’ place until resources are freed and transition may fire again.

V. EXAMPLE

The example for simulation is quite simple process – the user is prompted to enter both phone number and email. The request is processed and notification to user is sent about the availability of result. If the user does not retrieve the result within some amount of time, the notification is sent again. “Wait/Check” activity sleeps for some predefined time and then checks the value of the element “exitLoop” of each token to determine if it is possible to exit the loop and end the process instance or does the instance send the reminder once more. The example is not too complex because it contains the possible simulation problems discussed previously and there is no need to make it more complicated.

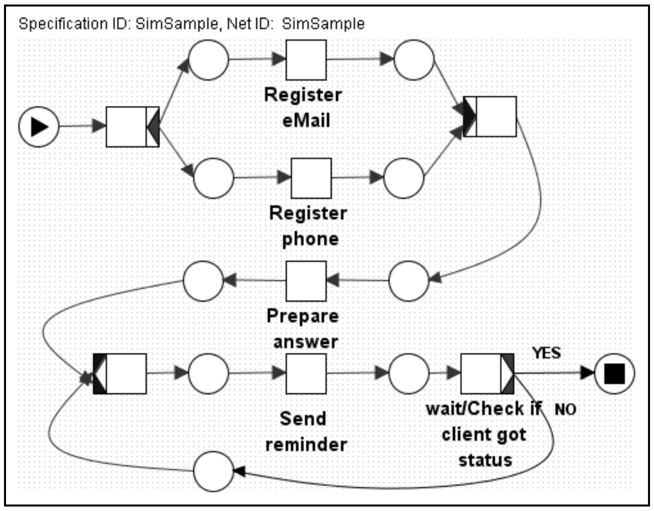
Specification ID: SimSample, Net ID: SimSample
Register eMail
Register phone
Prepare answer
Send reminder
wait/Check if client got status
YES
NO

Figure 6. YAWL workflow - the simulation example

The example consists of three blocks – AND-flow, proposed ‘bottleneck’ and possible infinite loop – Fig. 6. The user of e-service is prompted to enter his/hers email address and phone number – AND-flow. Then some worker/service checks the data and prepares answer – possible ‘bottleneck’. Finally the reminder to user is sent until he/she turns up for the answer – possible infinite loop. The activity ‘Prepare answer’ is assigned to YAWL resource “User1”.

There were 12 process instances created – in 6 of them both e-mail address and phone number were provided, in other 6 just the e-mail address. As mentioned before, CPN Tools cannot simulate OR-flows – the only way to diagnose waiting processes is using ‘Basic Log Analysis’ from ProM toolset. Fig. 7 illustrates the measurements of the execution count of each activity – the activity “Register_eMail_14” has been executed in all 12 instances, while the activity “Register_phone_15” has been executed only 6 times.

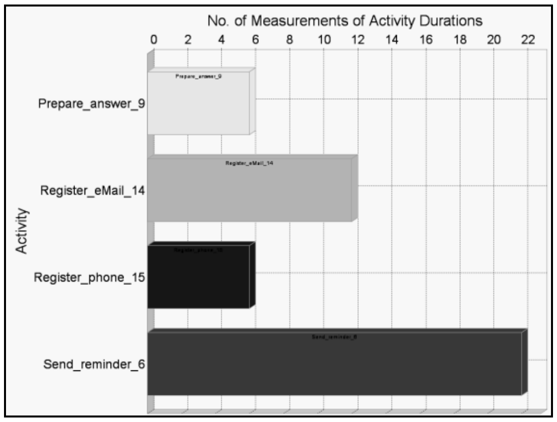

Figure 7. Basic Log Analysis – activity execution total count

After the tweaking the workflow – changing the OR-flow to AND-flow, all instances of the process could complete both activities. Fig. 8 shows the initial Petri net which corresponds to YAWL workflow in example. It has one token in place “Resources”, which means that all the concurrent instances will be processed in order by the same user, as seen in Fig. 9. The availability of only one resource token forces a wait in other processes, illustrated by 5 tokens in place “p4”, waiting for the resource “USER 1” to become available again. Transition “Timeout” was created to add artificial delay, which simulates the processing of the application.

Of course, it could be implemented using timed data types for all colsets used in net, but for the demonstrative purposes timed net is not used.

Fig. 9 also shows 6 instances moving through loop block – possible candidate for an infinite loop. The loop exit variable “exitLoop” in each of the tokens holds value “FALSE”, so until the transition “t3” can accept the token, it is stuck in the loop.

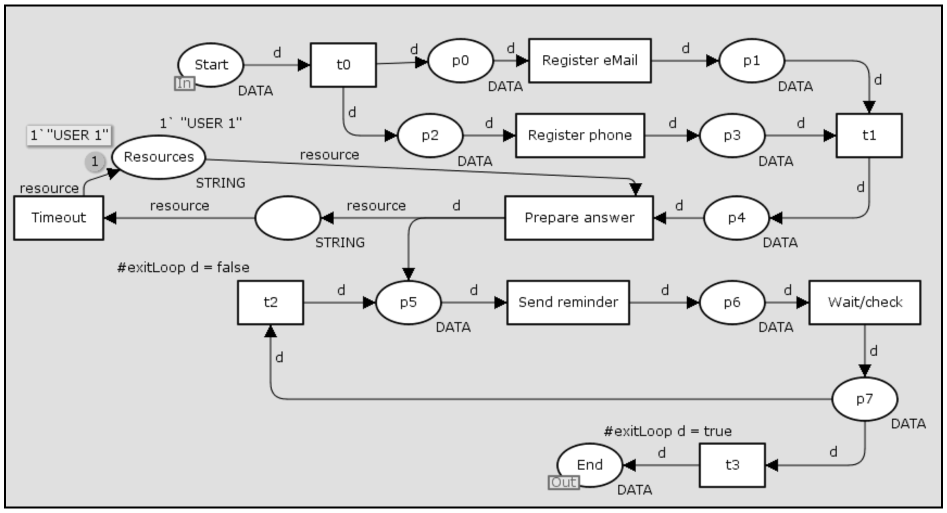

Figure 8. The Petri net corresponding to the example

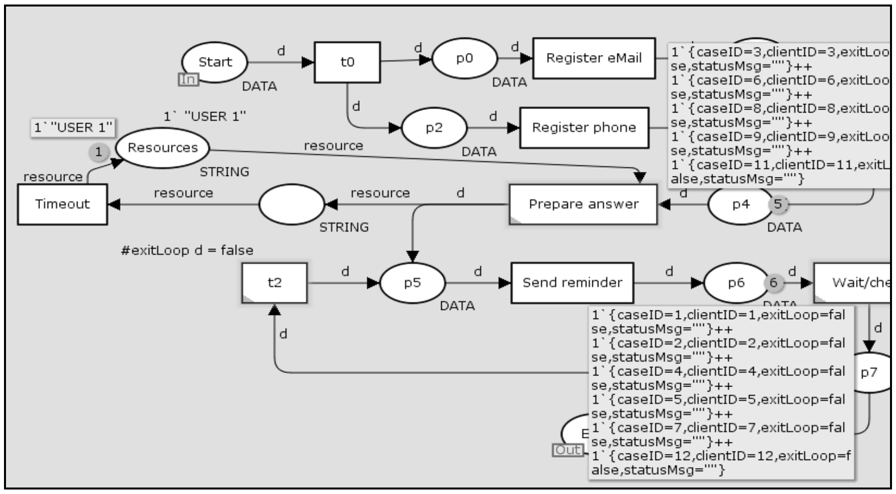

Figure 9. The bottleneck and the infinite loop in the Petri net

Examining the net in Fig. 9, it is clear that the process lacks an activity where the applicant can receive the results, thereby changing the value of variable “exitLoop” in the corresponding token.

VI. CONCLUSION

The proposed approach of business process modeling, when initial business process model is created using academic language and then transformed to the process described in business language, is quite successful. The main benefit of this approach is creation of primitive structure, which would later allow the transformation from YAWL workflow to any other hierarchical language (not only BPEL), both academic and business.

After examining and simulating simple workflow containing three possible problems, it is clear that the approach can identify simpler design problems; for example, the bottleneck and dead end identification can be achieved with this approach. However, more complex problems, such as deadlock identification or the operation of cancellation region could not be resolved. The problem lies with Petri nets, used in the simulation model, because they lack support of multiple simultaneous process instances or cancellation regions. One possible solution that could detect the deadlocks in the process model would be to use XML Process Definition Language (XPDL). Unfortunately, the XPDL supports only 9 out of 20 workflow patterns, while YAWL supports 19 [3][17].

On the other hand, the deadlock identification may be not as important since the deadlock situations would not arise in pure BPEL orchestration. The restrictions imposed upon YAWL workflow would also prohibit the use of OR-flow – another pattern which cannot be simulated. Taking into account the restrictions, authors conclude that the proposed simulation approach may be successfully applied during the development of electronic services.

ACKNOWLEDGMENT

The research leading to these results has received funding from the research project “Information and Communication Technology Competence Center” of EU Structural funds, contract nr. L-KC-11-0003, signed between ICT Competence Centre and Investment and Development Agency of Latvia, Research No. 1.21 ”Research on effective transformation of business processes to the architecture conforming to cloud computing”.

REFERENCES

[1] Web Services Business Process Execution Language Version 2.0, Online. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, accessed on the 19th of February, 2014

[2] Web Services Flow Language Version 1.0, Online. Available: http://cin.ufpe.br/~redis/intranet/bibliography/standards/leymann-wsfl01.pdf, accessed on the 19th of February, 2014

[3] W. M. P. van der Aalst and A. H. M. ter Hofstede, „YAWL: Yet Another Workflow Language” Information Systems, vol. 30(4), 2005, pp. 245–275.

[4] W. M. P. van der Aalst, L. Aldred, M. Dumas, and T. A. H. M. Hofstede, „Design and implementation of the YAWL system”, Proc. of the 16th International Conference on Advanced Information Systems Engineering (CAiSE 04), LNCS, vol. 3084, 2004, pp. 142–159.

[5] M. Weske, "Business process management", Springer, 2007, p. 169.

[6] S. Pornudomthap and W. Vatanawood, “Transforming YAWL workflow to BPEL skeleton”, Proc. of the IEEE Software Engineering and Service Science (ICSESS 11), Beijing, China, July 2011, pp. 434-437.

[7] N. Russell, A. H. M. ter Hofstede, W. M. P. van der Aalst, and N. Mulyar, „Workflow control-flow patterns: A revised view”, BPM Center Report BPM-06-22, BPMcenter.org, 2006.

[8] S. Rosario, A. Benveniste, S. Haar and C. Jard, “Foundations for web services orchestrations: functional and QoS aspects,

jointly", Proc. of the 2nd International Symposium, Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 06), 2006, pp. 309-316

[9] E. Zeiris and M. Ziema, "E-Service architecture selection based on multi-criteria optimization", Proc. of the 8th International Conference PROFES 2007, Riga, Latvia, July 2007, pp. 345-357.

[10] M. Havey, „Essential business process modeling", O'Reilly 2005, p. 141.

[11] P. Wohed, W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede, "Pattern based analysis of BPEL4WS", FIT Technical Report, FIT-TR-2002-04, Queensland University of Technology, Brisbane, 2002.

[12] OASIS WS-BPEL Extension for People, Online. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=bpel4people, accessed on the 18th of February, 2014

[13] T. Holmes, M. Vasko, and S. Dustdar, "VieBOP: Extending BPEL engines with BPEL4People", Proc. of the 16th Euromicro International Conference on Parallel, Distributed and network-based Processing, February 2008, pp. 547-555.

[14] A. Rozinat, M. T. Wynn, W. M. P. van der Aalst, A. H. M. ter Hofstede, and C. J. Fidge, "Workflow simulation for operational decision support using design, historic and state information", Proc. of the 6th International Conference on Business Process Management (BPM 08), Milan, Italy, Springer, 2008, LNCS, vol. 5240, pp. 196 – 211.

[15] W.M.P. van der Aalst et. al., "ProM 4.0: Comprehensive support for real process analysis" In J. Kleijn and A. Yakovlev, editors, Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007), Springer, 2007, LNCS, vol. 4546, pp. 484-494.

[16] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri nets and CPN Tools for modelling and validation of concurrent systems", International Journal on Software Tools for Technology Transfer, vol. 9(3-4), 2007, pp. 213-254.

[17] WFMC, Workflow Management Coalition Workflow Standard, Workflow Process Definition Interface – XML Process Definition Language (XPDL)(WFMC-TC-1025), Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA, August 2012.

[18] D. Lee, H. Shin, and B. K. Choi, "Mediator approach to direct workflow simulation", Simulation Modelling Practice and Theory, vol. 18(5), May 2010, pp. 650-662.

[19] Y. Zheng, Y. Fan, and W. Tan, "Towards workflow simulation in service-oriented architecture: an event-based approach", The 1st International Workshop on Workflow Systems in Grid Environments (WSGE '06), vol. 20(4), March 2008, pp. 315-330.