Class 23

Learning Objectives

- Be able to *use* process modelling notations other than YAWL
- Be able to *translate* process models between YAWL and other notation
- Be able to *evaluate* other modelling notations

Readings

After the very hands-on classes involving WoPeD and YAWL, you now have a very good understanding of the capabilities of the YAWL notation and the YAWL software. This class and the next three provide a little bit of context around what you have learned. Obviously, YAWL is not the only game in town for business process modelling and implementation. In fact, some would argue that YAWL is the outlier in this area. However, we believe it was instructive to look at process modelling from a vendor-independent perspective first. Remember that YAWL was developed based on research, not to make money, so it might arguably provide a perspective on process modelling without trying to sell you something. With this background, you should be well equipped now to make sense of other languages and products in the business process area. This class will look at BPMN (Business Process Modelling Notation). BPMN is a result of an industry standardization process and is a standard developed by the Object Management Group. Historically, one of its inputs was the BPEL (Business Process Execution Language, we will discuss this in class 20) which does not have a graphical, easy-to-use notation. Hence, BPMN was to be the graphical "front-end" to BPEL. Consequently, the translation from BPMN to BPEL is important. However, as we will see in class 20, this translation is a little problematic as the two languages are somewhat mismatched. Despite this, BPMN is in fact the industry standard for workflow management and process modelling today with many vendors providing BPMN modelling products.

In this class, you will get to know BPMN from the perspective of the YAWL language. As you read Chapters 13, you should focus on identifying things that you can express in YAWL but not in BPMN and things that you can express in BPMN but not in YAWL. Sometimes, these may be real shortcomings of the language, other times there may good reasons why something is not possible on a language.

Chapter 13: The Business Process Modeling Notation

The chapter is relatively easy to read, so we will provide only limited comments and notes on this. On page 352, you will see the expression "concrete syntax". This terminology is frequently used together with the term "abstract syntax" to differentiate between, e.g. the model and the way the model is written. For the data example on page 352, the data could be specified in many other forms, and XML plays the role of the concrete syntax here. In this case there is no generic abstract syntax, but you could for example think of a simple list of the data items as an abstract syntax specification.

On page 354 you can see that BPMN provides exception handling using "throw" and "catch". This idea is borrowed from modern programming languages. "Throwing" an error or an exception simply means that the process signals this error and stops executing. The (workflow) software then looks into the surrounding or higher-level process to see whether there is an way to handle this error or exception. If there is, that higher-level process then "catches" the error. Otherwise, the error is raised even higher (to

the surrounding process of the surrounding process) and so on, until a handler is found to “catch” that error.

As you read this chapter, you may want to reflect on the following questions:

- The introduction to Chapter 13 suggests that “the requirements for process modeling notations at these two levels of abstraction [modelling and implementation] are significantly different” (pg. 347). Do you believe this to be true? If so, why do you think YAWL tries to do both? Does it do it well? Or, given your experience with YAWL, do you think there is good reason to separate the two? What are some of the pro's and con's of either way?
- Why do you think BPMN makes a distinction between data objects and properties? What possible use could this have? Or do you think this should best be combined or that one or the other should be used (and why)?
- Given the comparisons in Figure 13.7 and Figure 13.12, and the discussion of Events on page 358, do you think it would be sensible to include explicit message and timer triggers in YAWL? How might you do this?
- While the chapter tries to translate between BPMN and YAWL and thereby also makes comparisons, what *criteria* do you think we should apply to *evaluate* process modelling languages? In other words, when you want to choose a language for your business, *how* would you choose? Are all the criteria you thought of equally important? Applying these criteria to BPMN, EPC and YAWL, which language do you prefer?

Review Questions

After this class, you should be able to answer the following review questions:

- List and describe the main modelling elements of BPMN
- Identify equivalent YAWL modelling elements for each of the BPMN tasks, events, and gateways
- Explain how BPMN exceptions (error and timer) can be translated to YAWL models
- Describe how YAWL cancellation regions can be translated to BPMN.

Review Exercises

- Chapter 13, Exercises 1-7