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

Event-driven Process Chains (EPC) were one of the first process modelling languages, and developed in Germany for the SAP Enterprise Resource System. The SAP R/3 system was arguably the first enterprise software developed with a process focus. Hence, it was important to not only document the processes for SAP customers but also provide a way for customers to model processes that are then implemented in the R/3 software. Beginning in the 1980s, a long time before other languages, EPC were developed primarily by Dr. August-Wilhelm Scheer at the University of Saarbrücken, who also took an early role in the development of the SAP systems. Since SAP is the market leader in enterprise software with a market share ranging between 40% and 60%, the EPC modelling language became quite influential in practice, even if it's specification and theoretical properties are not very rigorous. In fact, the early emphasis for EPC was on modelling and understanding of processes, rather than on process execution and precise semantics.

In this class, you will get to know EPC from the perspective of the YAWL language. As you read Chapter 14, you should focus on identifying things that you can express in YAWL but not in EPC and things that you can express in EPC 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 14: Event-driven Process Chains

This chapter deals with the concept of EPC and, after introducing the EPC modelling elements, presents translations from YAWL to EPC and from EPC to YAWL. **Do not read Section 14.6.** Again, the chapter is fairly easy to understand on its own, so we will provide only a few notes and clarifications.

On reading this chapter, it should become apparent that EPC are strongly based on the idea of Petri Nets with their alternating places and transitions that are similar to alternating events and functions in EPC. While this was the general idea behind EPC, EPC were never intended to have a precise meaning and be executable. Hence, the book describes “the informal (or intended) semantics of an EPC” (pg. 370) because there is no formal semantics from the developer of the EPC. This lack of precise specification makes it difficult to compare and translate EPC to other languages, as there can be much ambiguity in the meaning of an EPC. Given this, much depends on the interpretation of the person or company who will implement an EPC process in (workflow) software. One such ambiguity is discussed at the end of Section 14.3 with respect to XOR and OR joins.

On page 370, the term “transitive predecessor nodes” is used. This simply means the set of predecessors, the predecessors of predecessors, the predecessors of predecessors of predecessors, and so on, all the way to the beginning of the process branch.

The example presented in Figure 14.4 and discussed on page 375 is important to understanding the idea of free-choice nets. You should look at this example and its discussion carefully and fully understand it.

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

- There is some duplication in Figure 14.1. For example, the decision on delivery appointments and delivery arrangements is duplicated in two process branches. Can you think of ways in which this can be avoided? How might you wish to change EPC to help with these situations?
- Given that EPC events have no equivalent in YAWL (Section 14.4), how important to you think these are in practice? Should they or could they be left out of EPCs? What are the pro's and con's of having them?
- Do you think the development of yEPC (Yet another EPC) is wise, given that we already have YAWL which supports all the features missing from EPC? Why might yEPC still be a good idea?

Review Questions

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

- List and describe the main modelling elements of EPC
- Identify equivalent YAWL modelling elements for each of the EPC elements
- How are branching conditions for an XOR split indicated in EPC? Why is it bad practice to follow an XOR split with two (or more) functions and one should use events instead?
- Describe the idea of a “free choice net” and how places with multiple outgoing arcs may or may not represent choices

Review Exercises

- Chapter 14, Exercises 1-3 (Note for Exercise 1: A “trace” is a sequence of function executions)