Westfälische Wilhelms-Universität Münster

Topic:

Prototypical Development of a Docker-based Workflow Management System

Masterthesis
in the subject
at the Department of Information Systems — Practical Computer Science

Supervisor:	Prof. Dr. Herbert Kuchen
Tutors:	MScIS Vincent von Hof

Submitted by: Lars Greiving
Dettenstraße 4
48147 Münster

+49-176 704 253 17
l_grei02@uni-muenster.de

Deadline: 2016-02-24

Contents

List of Figures

List of Tables

Abbreviations

WfMS	Workflow Management System
WFMC	Workflow Management Coalition

Symbolverzeichnis

a_0	Anschaffungsauszahlung in $t = 0$
C	Kapitalwert
dt	Einzahlungsüberschuss in bezug auf t
i	Kalkulationszinsfuß
n	Nutzungsdauer
q	Zinsfaktor $1 + i$
r_s	Abstand der Stufe s in cm vom Seitenrand
s	Stufenindex
t	Periodenindex

1 Introduction and Motivation

1.1 Research Methodology

1.2 Related Work

2 Workflow Management Systems

In this chapter, the concepts of workflows and workflow management systems will be briefly introduced and related to each other. There is a plethora of term definitions and deviating understandings of workflows and the concepts related to them [3]. In large parts, the concepts presented here thus rely on specifications published by the Workflow Management Coalition (WFMC), a consortium of workflow management software vendors, researchers in the field of workflow management and Workflow Management System (WfMS) users, as they represent some form of consensus.

The identified use cases and properties will be used in **??** to identify objectives for the architecture. Also, they will be the reference to which the final architecture developed in this thesis is compared against.

2.1 Concepts

2.1.1 Workflow

In order to achieve their business goals, organizations perform temporal and logical sequences of tasks that help to interact with business relevant entities. These sequences are known as *business processes*. If the logic that controls the processes is performed in an automated way, e.g. by an information system, one refers to the processes as *workflows* [2, 6]. The WFMC defines workflows as the computerized facilitation or automation of a business process, in whole or part [6].

Process activities are the atomic steps that processes consist of. The WFMC differentiates between *manual activities* and *workflow activities*. The former are activities that involve user interaction in order to be completed, while the latter are automated and require no interaction [6]. As the term "workflow activity" might be misunderstood as "any activity belonging to a workflow", in the following the term *automated activity* will be used instead.

2.1.2 Process Definition

In order to be able to execute workflows, the underlying business processes must be machine processable and thus have to be formalized from real world to an abstracted model [6]. This model is usually called *process definition* and stored in form of some high-level programming language construct [6, 8]. The process definitions typically consist of a collection of activities with additional metadata such as associated applications or participants, and a set of rules which determine the execution order of these activities [6]. They further may contain references to other processes, which are treated as a single activity in the process definition [6, 3].

- usually directed grpah - how stored?

2.1.3 Process Instance

A *process instance* is an enactment of a process definition. A process definition may be instantiated multiple times, even at the same time. [3]. If only the automated parts of such an instance are meant, the WFMC advocates for the term *workflow instance* [6].

Process instances have several states. When they are created, they are in the *initiated* state. In this state, all relevant data has been provided, but the execution has not yet begun, e.g. because not all requirements are met. When the process is started, it enters the *running* state and it's activities may be started according to the process definition. If it has one or more instanciated activities, a process instance is in the *active* state. Process instances may be suspended, i.e. they enter the *suspended* state and no activities are instanciated until they leave it again. There are two states that a stopped process instance can be in. Either the completion requirements are met and the stopped process instance is in the *completed* state. Or the process instance stopped before its regular end, i.e. because of an error or manual interruption. In this case the process instance is in the *terminated* state [6].

A graphical representation of the state transitions described above can be seen in figure **??**. In this depiction, the allowed transitions between the different states are easy to grasp.

2.1.4 Activity Instance

Just like processes, activities are instanciated during workflow execution and have a set states that they may be in.

When an activity instance is created, it is in the *inactive* state. From this state, it may enter the *suspended* state, in which it will neither be activated nor assigned a worklist item. If the activity instance is not suspended, it is activated once its entry conditions are fulfilled. It then is in the *active* state. When the execution of the activity has finished, it finally enters the *completed* state [6].

The possible transitions between the activity instance's states can be seen in Figure **??**.

2.1.5 Workflow Data

In a WfMS, several forms of data may occur at diverse occasions. The WFMC differentiates between three types of data: workflow relevant data, workflow application data, and workflow control data [6].

WfMSs use *workflow relevant data* to determine a process instance's status and the next activity to be executed. It is normally available to the WfMS and both process- and activity instances [6].
Applications that are part of an workflow may work on domain specific data, which is called *workflow application data*. In most cases, the WfMS does not interact with this data other that providing it to the respective applications and limit access to it according to some authorization rules [6, 3].
Data that is internally managed by a WfMS is refered to as *workflow control data*. This data usually comprises the states of process- and activity instances and other internal statuses and is per se not interchanged in its default form [6, 3].

2.1.6 Workflow Participant and Worklist

There are workflows that contain activities which require user interaction. A WfMS thus provides the functionality to assign workflows and activities to workflow participants. This assignment can either be a specific one, targeting one single person, or be more general, targeting a set of users from which whe WfMS may choose during execution time. These sets are usually based on an organizational structure that manifests itself in roles, of which an user may have one or more [6, 3].

Each user has a so called *worklist* that consist of activities to which he is assigned to and which are scheduled for execution. Depending on the actual implementation, activites may appear on multiple users' worklists until one of them signals that he/she will work on it [6, 3].

2.2 Typical Architecture

With a growing number of workflows in an organization, the need arises to manage their creation, distribution and execution in a structured manner. An information system is called WfMS, if it is able to define, create and manage the execution of workflows by using software that runs on one or more workflow engines, is able to interpret process definitions, can interact with involved participants, and may invoke external applications [7]. According to the WFMC, a workflow management system is “a system that defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic” [6].

In the following, the typical foundations of WfMSs architectures identified by the WFMC are presented and related to the concepts introduced in Section 3.1.

2.2.1 Functional Areas

The WFMC divides the reponsibilities of a WfMS in three functional areas: *build-time* functions, *run-time process control* functions and *run-time activity interaction* functions [6, 1].

The *build-time* functionalities are concerned with the abstraction of workflows, i.e. the creation of process definitions.
The *run-time process control* functionalities of a WfMS are dealing with instanciating and controlling processes, coordinating the execution of activities within a process instance, initiating (but not performing) both participant interaction and application invocation [6].
Some activites require users to enter data or applications to perform a specific task. The *run-time activity interaction* functions of a WfMS provide the possibilities to do so. They make forms available to users, instruct other applications, and collect the respective outcome [6].

2.2.2 System Components

The WFMC identified four software components that most WfMSs have in common: *Process Definition Tools, Administration and Monitoring Tools, Workflow Client Applications,* and *Workflow Enactment Service* [6].

Process Definition Tool

Process definition tools are responsible for analysis, modelling, description and documentation of business proceses. The output of process definition tools – process definitions – can be interpreted by workflow engines in order to enact the respective workflow.

The WFMC notes, that process definition tools do not necessarily have to be part of a WfMS, since the definition may take place in another tool as long as it is passed along in a standardized format [6].

Administration & Monitoring Tools

The administration and monitoring tools are responsible for high level monitoring and control of the system. Their functionalities may include user management, role management, logging, performance auditing, resource control, and supervision over running processes.

Workflow Client Applications

The core function of the workflow client applications is to let the user retrieve worklist items that were assigned to him/her. In the WFMC reference model they are thus sometimes referred to as *worklist handlers* [6].
Yet, the WFMC stresses that their functionality may be much broader, e.g. letting him/her enter data that is associated to one worklist item, allow him/her to alter the worklist, signing in or off, or control the processes' statuses. The WFMC thus advocates for the term *workflow client applications* [6]. The user interface may be part of the workflow client applications or exist as a separate software component.

Workflow Engine

Workflow engines provide the runtime control environment for the execution of workflow instances, that is, they interpret the process definition, manage the instances' status, update worklists, determine participants, and invoke external applications. They further manage the storage and flow of workflow control data and workflow relevant data [6].

Workflow Enactment Service

The Workflow Enactment Service groups one or more workflow engines into one logical component that exposes a single coherent external interface to other software [6].

2.2.3 WFMS Implementation Structure

According to the WFMC, the components described in 2.2.2 interlock in order to provide the overall functionality of a WfMS. As visible in **??**, the workflow enactment service plays a central role in wiring the components together.

3 Docker

When multiple applications or application instances shall be run on one machine, they are usually isolated from each other in terms of execution environments and provided with a controllable share of system resources [5]. These goals can be fulfilled by both virtual machines and containers.

** here be dragons **

Docker is a tool, that simplifies container creation and management. In Section 3.1 the underlying concepts will be presented. Based on that, the functionality that Docker provides will be explained in Section 3.2. Finally, the Docker ecosystem, i.e. the set of tools that enhance the core docker tool, is introduced in Section 3.3.

3.1 Concepts

3.1.1 Application Containers

3.1.2 Docker Images

- image layering [4]

3.1.3 Docker Containers

- runtime instance of a docker image - last layer on top of the other images - *lifecycle of a docker container here*

3.1.4 Data Volumes

3.1.5 Dockerfiles

3.1.6 Registries and Repositories

3.2 Functionality

3.3 Docker Ecosystem

3.3.1 Docker Swarm

3.3.2 Docker Machine

3.3.3 Docker Compose

3.3.4 Docker Hub

4 Application Design

4.1 Determination of Objectives

4.1.1 Functional Requirements

Infrastructure and Infrastructure Management

Workflow Modeling

Workflow Distribution

Workflow Execution

Integration of Third Party Containers

4.1.2 Intangible Requirements

"It should be easier to use"

4.1.3 Derived Objectives

4.2 Architecture

4.2.1 Application Architecture

4.2.2 Workflow Execution Environment

4.2.3 Workflow Management Environment

4.2.4 Developer Client

4.2.5 User Client

5 Prototypical Implementation

5.1 Toolchain

5.2 Realization of the Architecture

Any Compromises here?

6 Evaluation and Discussion

7 Conclusion

Bibliography

[1] Alonso, G.; Agrawal, D.; Abbadi, A. E.; Mohan, C.: Functionality and Limitations of Current Workflow Management Systems. In: IEEE Expert, 12 (1997).

[2] Becker, J.; Uthmann, C.V.; Muhlen, M. zur; Rosemann, M.: Identifying the workflow potential of business processes. In: Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences, 1999. HICSS-32. Vol. Track5, 1999, pp. 10 pp.–.

[3] Casati, Fabio; Ceri, Stefano; Paraboschi, Stefano; Pozzi, Guiseppe: Specification and implementation of exceptions in workflow management systems. In: ACM Transactions on Database Systems, 24 (1999) 3, pp. 405–451.

[4] Docker, Inc.: Docker Documentation. https://docs.docker.com/.

[5] Felter, Wes; Ferreira, Re; Rajamony, Ram; Rubio, Juan; Felter, Wes; Ferreira, Alexandre; Rajamony, Ram; Rubio, Juan: An Updated Performance Comparison of Virtual Machines and Linux Containers. 2014.

[6] Hollingsworth, David: WfMC: Workflow Reference Model. In: Specification Workflow Management Coalition, http://www.wfmc.org/standards/docs/tc003v11.pdf 1995.

[7] Lawrence, Peter: Workflow Handbook 1997. New York, NY, USA: John Wiley & Sons, Inc. 1997.

[8] Wutke, Daniel; Martin, Daniel; Leymann, Frank: Model and Infrastructure for Decentralized Workflow Enactment. In: Proceedings of the 2008 ACM Symposium on Applied Computing. New York, NY, USA: ACM 2008 (SAC ’08), pp. 90–94. – http://doi.acm.org/10.1145/1363686.1363712. – ISBN 978-1-59593-753-7.