title
Whitepaper

Benjamin Leiding[1], Will Vorobev[1], Peter Zverkov[1], and Lena Cherry[1]

Chorus Technology

Abstract. I am an abstract - pet me.

DON'T FORGET THE PLAGIARISM CHECK

Keywords: keywords

1 Introduction

Work-In-Progress

WE ARE A LIBRARY!

The remainder of this paper is structured as follows: Section 2 introduces supplementary literature and related work. Section 3 then outlines the vision of Chorus Technology as well as different use-cases. Afterwards, Section 4 analyses the requirements of the our system and outlines the resulting system architecture that we derive from the requirements. Afterwards, Section 5 expands on the system-engagement processes for the stakeholders, followed by Section 6 that introduces the Chorus prototype as well as feasibility evaluation. Section 7 provides an discussion and an analysis of related projects. Finally, Section 8 concludes this work and provides an outlook on future work.

2 Technical Background and Related Works

The following section provides background information and describes related works regarding previous ideas and concepts that focus on a blockchain-based VANET platforms. First, Section 2.1 introduces the general concepts of blockchain technology, terms and frameworks. Afterwards, Section 2.2 and Section 2.3 focus on the fundamentals of autonomous vehicles as well as vehicular ad-hoc networks. Finally, Section 2.4 focuses on related work.

2.1 Blockchain Technology

Work-In-Progress

maybe already cite unchained and whisperkey paper?

2.2 Autonomous Vehicles

Work-In-Progress

2.3 Vehicular Ad-Hoc Networks - VANETs

Work-In-Progress

2.4 Related Work

Work-In-Progress

3 The Chorus Vision

Intro

3.1 Use Cases

Work-In-Progress

3.2 Human to Human

3.3 Human to Vehicle

3.4 Vehicle to Vehicle

3.5 Vehicle to Infrastructure

4 System Design and Architecture

The vision of Chorus outlined in Section 3 is now analyzed from technical perspective as part of the following section. In order to identify, structure and formalize the critical requirements and stakeholders on an abstract level, we use one part of an Agent-Oriented Modeling (AOM) method [10], i.e., goal models. Section 4.1 introduces AOM goal models and the Chorus specific goal model. The produced goal model is used in subsequent Section 4.2 to derive the Chrous system architecture . The resulting system architecture and specifications serve as implementation guidelines. Finally, Section 4.3 provides some details on the smart contract lifecycle management within our solution, whereas Section 4.4 outlines a general outlook on the APIs and how to integrate the Chorus library into external applications.

4.1 Functional Goals, Quality Goals, Stakeholders and Requirements

In system development and software engieerng, good requirements follow certain characteristics. According to [2][4] requirements address one issue only and are completely specified without missing information. Moreover, they have to be consistent and do not contradict itself, or in correlation with other requirements. Finally, a requirement must also be atomic and without conjunctions [6].

The AOM methodology is a socio-technical requirements-engineering approach used to model complex systems that consist of humans, devices, and software agents. An AOM goal model enables both, technical- and non-technical stakeholders, to capture and understand the functional- and non-functional requirements of a complex system. Figure 1 depicts the three main elements that an AOM goal model comprises in order to capture the system requirements and goals. Roles of involved entities are represented in form of sticky men, whereas functional requirements are depicted as parallelograms. Functional requirements are also referred to as goals. Non-functional requirements are depicted as clouds and refer to quality goals of the modeled software system. The AOM goal model follows a tree-like hierarchy with the root value proposition of the modeled system at the top. Subsequently, this main goal is decomposed into sub-goals where each sub-goal represents an aspect for achieving its parent goal [5]. The goals are further decomposed into multi-layered sub-goals until the lowest atomic level is reached. Additionally, roles and quality goal may be assigned to goals and are inherited to lower-level goals.

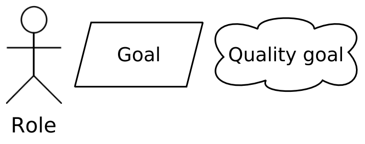

Fig. 1. Selection of AOM notation elements.

4.1.1 Top-Level AOM Goal Model

Fig. 2. Caption.

4.1.2 Refined AOM Goal Model

Fig. 3. Caption.

4.2 Component Diagrams

The abstract system- and business architecture is derived from the functional- and non-function requirements of the AOM goal model presented earlier. The services are powered by a service-oriented architecture (SOA) that is comprised of different designated components. Each of these components is self-contained, well-defined components and provides a specific set of services [3][7]. Dedicated services and components may also consist of other underlying sub-services [8].

In the following, a technology-agnostic UML-component-diagram representation is used to illustrate the system architecture [1][9]. The UML notation elements used to model the architecture are presented in Figure 4. In UML, components are represented as rectangular boxes and labeled either with the keyword *component*, or with the component icon in the right-hand upper corner. A component may consists of further sub-components and is implemented by one, or more classes, or objects. Moreover, components are reusable and communicate via two types of interfaces as illustrated in Figure 4. Small squares depict ports that are attached to the border of components and expose required and provided interfaces. Ports may also specify inputs and outputs as they operate uni-, or bi-directionally [1][9]. Once more, sticky men are used to depict entities and their interactions with the system.

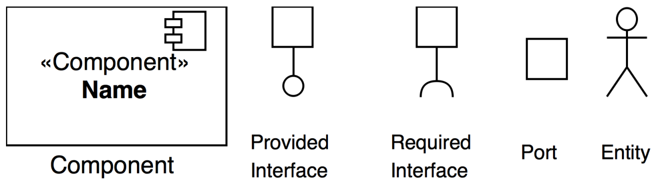

Fig. 4. UML-component diagram notation elements.

The remainder of this section first introduces an abstract high-level overview of the system architecture and components. Afterwards, further illustration present selected sub-components of our architecture.

4.2.1 High-Level Architecture

4.3 Smart Contract Lifecycle Management

Work-In-Progress

4.4 Library / API

Work-In-Progress

5 System Engagement Processes

Intro

5.1 Sequence Diagrams — or BPMN representation of Processes

Work-In-Progress

Fig. 5. Caption.

5.2 Auction Algorithm

Work-In-Progress

Fig. 6. Caption.

5.3 Token Economics and Value Proposition

Work-In-Progress

6 Prototype and Feasibility Study

Intro

6.1 Prototype

Work-In-Progress

6.2 Evaluation

Work-In-Progress

7 Discussion

Intro

7.1 Critical Analysis

Work-In-Progress

7.2 Related Work

Work-In-Progress

8 Conclusion and Future Work

We conclude ...
Work-In-Progress

References

1. Booch, G., Jacobson, I., Rumbaugh, J., et al.: The Unified Modeling Language. Unix Review 14(13), 5 (1996)
2. Davis, A.M.: Software Requirements: Objects, Functions, and States. Prentice-Hall, Inc. (1993)
3. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Pearson Education India (2005)
4. IEEE Computer Society. Software Engineering Technology Committee and Institute of Electrical and Electronics Engineers: IEEE Recommended Practice for Software Requirements Specifications. IEEE Std, Institute of Electrical and Electronics Engineers (1994)
5. Marshall, J.: Agent-Based Modelling of Emotional Goals in Digital Media Design Projects. International Journal of People-Oriented Programming (IJPOP) 3(1), 44–59 (2014)
6. Norta, A., Grefen, P., Narendra, N.C.: A Reference Architecture for Managing Dynamic Inter-Organizational Business Processes. Data & Knowledge Engineering 91, 52–89 (2014)
7. Perrey, R., Lycett, M.: Service-Oriented Architecture. In: Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on. pp. 116–119. IEEE (2003)
8. Rosen, M., Lublinsky, B., Smith, K.T., Balcer, M.J.: Applied SOA: Service-Oriented Architecture and Design Strategies. John Wiley & Sons (2012)
9. Specification, O.A.: OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2. Object Management Group (2007)
10. Sterling, L., Taveter, K.: The Art of Agent-Oriented Modeling. MIT Press (2009)