High resolution and transparent production informatics

Key enabling information technologies for supplier collaboration management (SCM) systems

PhD Thesis

Dávid Karnok

Supervisor:
László Monostori, academician

Budapest, Hungary, 2017

Alulírott **Karnok Dávid** kijelentem, hogy ezt a doktori értekezést magam készítettem és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával jelöltem.[1]

Budapest, 2017. október 30.

_____

Karnok Dávid

[1] See http://www.gpk.bme.hu/PhD/images/gepeszkar/doku/PhD-beadasi-urlapok/Nyilatkozat_onallo_munkarol.doc

”If the ancient societies had understood the concept behind the simple child-play of potato-printing, we could have had printed books for at least 4,000 years now.”

– unknown

”Ha az ősi civilizációk megértették volna az egyszerű krumpli-nyomda gyerekjáték mögötti elveket, akkor már legalább 4000 éve lehetnének nyomtatott könyveink.

– ismeretlen

Acknowledgements

Thanks

First of all, I would like to thank my mother *Anikó*, my father *József* and my sister *Noémi*, who where patient during the last 12 years; not being nagged about when I am going to finish my PhD (or when will I get married) helped ease the discomfort of the ever present pressure.

Second, I would like to thank my thesis advisor, *László Monostori* for his patience of my – probably record breaking – time it took to compose the solid results into a dissertation. His high level view of the problem domain in general helped greatly in shaping the words of this dissertation.

Third, I would like to thank my closest colleagues who endured my endless complaints about research & development over the years:

Zsolt Kemény, who helped me in situations requiring scientific thinking and made suggestions how to write about certain facts in a politically correct fashion.

János Csempesz, with whom I could talk a lot about computer games and advanced information technology whenever I needed to clear my head.

József Váncza, whom without I most likely wouldn't have started actually writing this dissertation; it's astonishing what weekly consultancy can bring about.

Finally, thanks to the rest of my colleagues at the *Research Laboratory on Engineering & Management Intelligence* at SZTAKI.

Grants

This research has been supported by the Hungarian Scientific Research Fund (OTKA), Grant No. 113038.

This research has been supported by the GINOP-2.3.2-15-2016-00002 grant on an "Industry 4.0 research and innovation center of excellence".

Abstract

Today, we are at the brink of a new industrial revolution, the fourth to be precise, in which information technologies and computer science results are to become first class citizens of the so-called cyber-physical production- and logistics systems movement. The ubiquitous presence of communication networks, smart- and embedded devices and the so-called cloud allow the real-time tight coupling of the material and information flow of production- and logistics processes and enable formerly hard-to-achieve goals, approaches and novel technologies.

However, there are some caveats remaining from the previous industrial revolution in both the physical- and the cyber-world. First, it appears the third industrial revolution revolved around mostly the programmable logic controller (PLC) that helped automate processes but even though (personal) computers and communication networks were in their infancy, the lack of understanding and unable to see their potential, companies didn't push these technologies in time, potentially delaying the evolution by decades. Second, as computers and networks became more accessible, software supporting the manufacturing- and logistics operations where developed, but these couldn't really interchange information with each other neither within nor outside the organization borders; therefore, many processes kept relying on paper-based or phone call-based recording and data-exchange protocols. Just a small step above these, the use of a (usually free-form) spreadsheet for planning and coordination left many information sources untapped. Third, the need for standards in communication and data formats were eventually realized, however, many of them were designed by 'comittees' with limited sight and expertise and were forced upon implementors and companies in the name of conformance (i.e., see Common Object Request Broker Architecture[2] and Electronic Data Interchange[3]). Not to mention, many competing standards were introduced and employed over the years[4].

In the new millennium, the availability of various information technologies to all, mostly competing companies doesn't give the expected 'competitive' edge in itself anymore. Companies are looking at buzzworlds such as Big Data, the Cloud and the aforementioned cyber-

[2] http://www.omg.org/spec/CORBA/3.3/

[3] http://www.nist.gov/itl/, although hard to locate

[4] Summarized in the famous XKCD strip: https://xkcd.com/927/

physical duality, however, there is a significant technological dept accumulated in both the physical and information processes and technologies over the last decade. Many business process re-engineering efforts didn't finish and many information technologies weren't adjusted to accommodate the new era. For example, the Enterprise Resource Planning (ERP) can be found at many companies but the core technologies of such software hasn't been significantly evolved in the past 20 years, making them unsuited for the so-called Extended Enterprise functionality. Over the years, their strictly closed architectures and communication capabilities gave rise to many (custom-developed) satellite software solutions which utilized much more recent (although still 3–5 year old) technologies and approaches. Naturally, maintaining such systems becomes costly over time, therefore, software- and hardware virtualization have become a trend in recent years, peaking in the Cloud-based solutions as the ultimate cost-saving measure.

Moving the various software into the Cloud, just by itself, doesn't yield benefits outright; certain classical problems are now replaced with new problems: (1) the unavoidable departure from the classical paper & phone processes, (2) security, secrecy and information sharing, (3) scaling with the data amount and/or the demand, (4) becoming responsive and resilient and (5) exploiting knowledge accumulated in various data sources inside the company. In other words, companies have to become more transparent, more reactive and more knowledgeable than ever before. The dissertation details the research effort into these problems and provides novel so-called key-enabling technologies to solve (partially or fully) the challenges. Novel approaches are presented for each of the main problem areas mentioned above: (1) methodologies and models for real-time production and logistics information management, (2) methodologies and protocols for sharing information through the organization borders and (3) methodologies, programming paradigms and algorithms for scalable, resilient and responsive information processing.

Absztrakt

Napjainkban egy új ipari forradalom – szám szerint a negyedik – kiteljesedése előtt állunk, amiben az információs technológiák és a számítástudomány eredményei egyenrangúvá válnak a kiber-fizikai gyártó- és logisztikai rendszerek világában. A bárhol elérhető kommunikációs hálózatok, okos és beágyazott eszközök és az ún. Felhő lehetővé teszik a gyártó és logisztikai folyamatok anyag- és információs áramának valósidejű összerendelését, elősegítve ezáltal korábban csak nehezen elérhető célok, megközelítések és újszerű technológiák alkalmazását. Az előző ipari forradalomból azonban hátramaradt néhány buktató a fizikai- és kiber-világot egyaránt érintve.

Először is, a harmadik ipari forradalom, látszólag, főleg az ún. programozható logikai vezérlők (PLC) körül forgott, amely segített a folyamatok automatizálásában. Habár a (személyi) számítógépek és kommunikációs hálózatok még gyerekcipőben jártak, a megértésük hiánya és a bennük rejlő lehetőségek felismerésének elmaradása miatt a vállalatok nem fektettek elég hangsúlyt ezekre a technológiákra, így akár évtizedekig is késleltetve a fejlődésüket.

Másodszor, ahogy a számítógépek és hálózatok hozzáférhetőbbé váltak, gyártási- és logisztikai-folyamatokat támogató szoftverek lettek kifejlesztve. Sajnos ezek a különféle cégeknél kifejlesztett szoftverek gyakran nem voltak képesek egymással (kellő minőségben) kommunikálni – akár cégen belül, akár a cégek között –, emiatt nagyon sok folyamat megmaradt a papír- és telefon-alapú információ-csere protokollok szintjén. Ennél egy kicsit fejlettebb megoldás a (szabad formájú) táblázatkezelők űrlapjainak használata a tervezésben és koordinálásban, amely továbbra is számtalan információforrást hagyott kiaknázatlanul.

Harmadszor, a kommunikációra és adatformátumokra vonatkozó szabványok ugyan létrejöttek, azonban a legtöbbjüket bizottságok terveztek korlátozott látókörrel és technikai tapasztalatok mentén, majd azt a konformancia nevében, függetlenül annak alkalmazhatóságától az adott célterületen, minden fejlesztőre és vállalatra rákényszerítették (pl., lásd Common Object Request Broker Architecture, CORBA[2] és Electronic Data Interchange, EDI[3]). Azokról az esetekről nem is beszélve, amikor éveken át egymással versenyző szabványok tömege[4] volt használatban anélkül, hogy bármelyik szabványt is egyértelmű nyertesnek nevezhetnénk.

Az új évezredben a különféle információs technológiák elérhetősége bármely vállalat számára már nem biztosít versenyelőnyt önmagában. A vállalatok jellemzően olyan címszavakra (buzzwords) koncentrálnak, mint pl. a Nagymennyiségű Adatok (Big Data), a Felhő (Cloud) és a korábban említett kiber-fizikai dualitás, viszont legtöbbjüknél az elmúlt évtizedekben jelentős technológiai adósság halmozódott fel a fizikai- és információs folyamataikban egyaránt. Számtalan üzleti folyamat újratervezési projekt nem fejeződött be és számtalan meglévő információs technológia nem lett az új korszak igényeihez igazítva. A legtöbb vállalatnál megtalálható egy központi vállalatirányítási rendszer (Enterprise Resource Planning, ERP), de azok alapvető építőelemei és technológiái érdemben nem fejlődtek az elmúlt 20 évben, ezáltal nehezen vagy egyáltalán nem tudnak részt venni az ún. bővített vállalati funkcionalitás megvalósításában jelen formájukban. Az évek során a zárt architektúrájuk és kommunikációs hiányosságaik miatt számtalan, egyedi fejlesztésű, ún. szatelit (a központi rendszertől különálló, de vállalaton belüli) szoftver készült el, amelyek már korszerűbb (de így is 3–5 éves) technológiákat és módszereket alkalmaznak. Természetesen az ilyen külsős rendszerek karbantartása idővel megnövekedett költséget jelent, emiatt a számítástechnikai hardverek és szoftverek virtualizációja általi erőforrás-összevonások jellegzetes trenddé váltak az elmúlt években. Ezen irányvonal csúcsát, és ezáltal a végső költségmegtakarítást maguk a Felhő-alapú megoldások jelentik. A különféle (meglévő) szoftverek számítási felhőbe mozgatása önmagában nem biztosít azonnali előnyöket, mert néhány korábbi problémát új problémák váltanak fel: (1) a klasszikus papír és telefon-alapú folyamatoktól való elszakadás, (2) biztonság, titkosság és információ megosztás, (3) az adatmennyiséggel és/vagy igénnyel párhuzamos skálázhatóság, (4) időben válaszoló és hibatűrő megoldások szükségessége és (5) a vállalatoknál felhalmozódott tudás és adatforrások jobb kiaknázása. Más szóval, a vállalatoknak átláthatóbbá, reaktívabbá és nagyobb tudásúvá kell válniuk, mint azelőtt. A disszertáció ezen problémák mentén végzett kutatások eredményeit, **új információs kulcstechnológiákat** ismertet, amelyek hozzájárulnak a fentebb vázolt problémák kezeléséhez: (1) módszerek és modellek a valósidejű gyártási és logisztikai információmenedzsmentben, (2) információcsere módszerek és protokollok a szervezetek között húzódó határokon keresztül és (3) módszerek, programozási paradigmák és algoritmusok skálázható, hibatűrő és időben válaszoló információfeldolgozás céljából.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

Today, we are at the brink of a new industrial revolution process often referred to as **Industry 4.0** [56] and **Cyber-Physical Production Systems** (CPPS) [73]. Allowing computer science results to become first class citizens in manufacturing and logistics systems and -processes is a long lasting dream which can now be achieved.

Even though computers, software and algorithms were part of these manufacturing and logistics systems for over a decade, their second class nature meant the physical processes were either completely detached from the information processes or there were significant latencies and quality differences between the two processes. The ubiquitous nature of the Internet, the ability to communicate wirelessly and to put computational power into components and products directly and more economically enable the cyber and physical systems to operate hand-in-hand allowing greater flexibility, better response times and overall improved key performance indicators [72].

In the past decade, the amount and frequency of data available in manufacturing and logistics environment have been increasing almost exponentially. The so-called big-data revolution prompts for **high resolution** [87] information technologies and control solutions on both the offline and the online – (near-)real-time – processing of data and events.

Several strategies and key areas have been identified to allow shifting from the classical industrial production and logistics to the new ways of the CPPS [56] by **end-to-end digital integration**, **management of complex systems** and **resource efficiency**, to name some of the relevant points in regard to the dissertation. In addition, I propose putting emphasis on the requirement of these systems and components to be built in reactive fashion (in the sense of data- and computation-reactiveness) as being **elastic and message driven** in addition to being **resilient and responsive** on par with the Reactive Manifesto[5] for software in general.

However, for most companies, joining the CPPS world cannot just happen over-night as such leap requires design, planning and rigorous change management. Many companies in the field of manufacturing and logistics I worked with, although use up-to-date hardware, have legacy

[5] http://www.reactivemanifesto.org

systems built with outdated software technologies and usually have accumulated a technological debt of 5 to 10 years (i.e., still relying on spreadsheets, programs written in programming languages several versions before and using old and non-standards-compliant web browsers).

What are Cyber-Physical Production Systems really? From a conceptional standpoint, these are systems that attempt to draw benefits from the recent evolution of IT: namely the Cloud, Big Data, Wireless networks, Smart devices/sensors, modern algorithms and paradigms, machine learning, Internet of Things and service orientation [91]. These are also called core-enabling technologies of CPPS, but many of them were developed independently and without a common focus. CPPS aims to bring these technologies under a common umbrella [25, 34]. This involves a multi-axis integration: (1) horizontal integration across the value chain (such as digital integration of suppliers and customers), (2) vertical integration across the manufacturing systems (such as digital integration from the sensor up to the Enterprise Resource Planning level) and (3) engineering integration (such as CAD/CAM, factory- and machine-design, workflow design, etc.) [56, pp. 31–32]. Targeting the Cloud as the platform for the software technologies backing the CPPS is a common theme and the main cost-saving factor. The so-called Cloud-Manufacturing [91, 97] and its engineering extension, the Cloud-based Design and Manufacturing [106, 107] aim to exploit the general benefits of cloud computing, namely: (1) self-service, (2) ubiquitous network access, (3) rapid elasticity, (4) pay-per-use and (5) location-independent resource pooling [105]. The vertical digital integration, consequently, brings in a torrent of information from sensors, product-tracking solutions (RFID), machines, inventory tracking etc. Dealing with and extracting valueable information from the data is no longer feasible on a small scale IT level, therefore, one has to move to the so-called Big Data world with distributed computing, vast amount of storage and advanced data processing- and mining-algorithms [59, 61, 59, 60].

From a practical perspective, however, CPPS systems appear to be nothing new, at least on component level. One can argue, that *"if you've written a web-enabled application in the past 15 years, you have essentially a CPPS system and you only need it to move to the Cloud by virtualizing its current running environment"*. Clearly, the movement of Manufacturing Execution Systems (MES) [47] and Enterprise Resource Plan-

ners (ERP)[6] appear to reinforce this view. Whether or not CPPS is new, it offers benefits and features over the previous web-based and agent-based approaches and its open nature is more suited to cross-company interoperation: (1) encourages sharing, (2) better real-time operation, (3) multi-tenancy software[7] and (4) "software-izing" hardware through approaches such as Platform-as-a-Service (Paas) or Infrastructure-as-a-Service (IaaS)[106]. In short, the Cloud is someone else's hardware and software which one can borrow for a well defined (and thus plannable) costs per time and usage level.

Moving into the Cloud-era, unfortunately, has its pitfalls mostly unaddressed or bypassed (i.e., by using a private Cloud, which defeats most of the cost benefits) in the literature.

The first issue centers around safety, secrecy and security [88]. Software, hardware and the network may fail and if *resilience* is not built into the solution, it can cause major disruptions which, due to the integration dimensions, may affect large proportion of the value chain. In addition, CPPS systems are to rely on the real-time nature of the physical world that the disruption/delay in the information flow, i.e., the lack of *responsiveness* can cause similar problems. In classical logistics networks, secrecy is one of the utmost important factors of cooperation.

Having partners to share business-critical data with each other is still a difficult task to achieve, but making such data cloud-enabled is even more difficult to make happen: sharing with a third party (the Cloud provider) is of great concern. There are no physical barriers such providers (or adversaries) to gain access to the unencrypted data (remember, one can store the data encrypted, but for processing, it has to be decrypted somewhere).

Perhaps the security aspects of Cloud is the most worrying due to the increased frequency of hackings, security vulnerabilities and various forms of espionage recently. An adversary, exploiting vulnerabilities or poor software implementations[8] can gain access to business critical data, steal it or even manipulate it. Defending the data with tight access control is only one part of the solution, organizing the frequent patching of software component across the vertical integration (from sensor through network equipment up to the ERP) has to be designed and implemented

[6] SAP Insider: Cloud Computing and SAP, 2010, link, last accessed: September, 2010

[7] Instead of buying the software, the software is leased to many users, sharing the costs.

[8] See SQL injection.

into the CPPS solutions.

The second issue is with the existing ERP systems, many of which is already web-enabled but haven't really evolved conceptionally in the past two decades [53] and carry significant technological debts themselves, such as lack of (truly) standard communication tools, limited modelling capabilities regarding multiple company sites or extended enterprise functionality [21]. As of today, ERPs appear to be only a modularized platform and not much of an off-the-shelf software solution: almost every company that wants its ERP to handle more than employee-, payment- and warehouse- management (which, by the way, don't require such a diverse model), needs to spend significant time and resources to implement the highly company- and product-specific information management [94].

The third issue is the need for new contracts or even new legislations regarding contracts [56, pp. 58–60]. Such contracts need to establish the minimum set and service-level of information sharing between partners and a clear description of who is responsible for what (i.e., cost, integrity, reliability and availability). This may involve (monetary) compensation schemes and intellectual property (IP) related issues that most legal systems today doesn't allow or can resolve effectively.

The fourth issue is with the increased amount and the effects of Big Data on human comprehension. By putting sensors and tags onto everything, the sheer amount of raw data becomes a maintenance issue and decision making based on these is hardly possible or even close to optimal. The main trend is to employ distributed data processing and analytics and present the decision maker with distilled information. This implies that all data is processed which due to the cost-structure of cloud can become expensive. As an analogue to the situation, the most extreme case for the situation can be found at the European Organization for Nuclear Research (CERN)[29]: a single experiment run yields a daily 1 petabyte[9] of raw sensor data. Storing and processing such amount of data is infeasible today, and due to the increase of experimental energy levels (~ 13 TeV[10], tera-electron volts) and sensor density, it is likely to stay ahead of the well known Moore's law. Interestingly, the solution was mainly twofold: move the data processing as close to the sensors as possible (which implies custom hardware) and use early *filtering* of the data to reduce the

[9] that is $\sim 10^{15}$ bytes or ~ 1000 terabytes

[10] http://www.bbc.co.uk/news/science-environment-32976838, last accessed: July, 2015

analysis (and storage) space for likely interesting information patterns.

As a final thought, it might not be possible or feasible to upgrade some older companies and factories to fully embrace the CPPS benefits due to its highly disruptive nature. The level of ”hard-coded” workflows, information systems and the physical nature of manufacturing may prevent proper transition to the Cloud, especially if there was no successful business process redesign (BPR) beforehand. Dealing with disruptive technologies, sometimes, requires the ”reboot” of the factory and even the supply network to some extend [33].

1.2 Motivation

In the past 10 years, I have been working on various academia–industrial research and development projects and many different companies involved in these projects had very similar issues, regardless of their size, location and field. These boil down to the following cases:

- the (current) status of the shop floor or logistics network is not available in a preprocessed and/or timely manner,
- the existing Enterprise Resource Planning (ERP) implementation is too limiting, the extension of the ERP system is very expensive,
- the software is slow or inadequate (i.e., spreadsheet) or the information flow mostly relies on paper- and phone-based processes, and
- the production informatics is unable to exploit the available data due to privacy concerns (i.e., limiting the types of data sources exposed versus the ability to utilize more data sources for extracting new kinds of information).

Interestingly, these research issues appear to be already solved by either academic research and/or Information Technology (IT) advancements. However, a thorough review of the information sources usually reveals that the existing approaches are either (1) defined too broadly and leave a significant gap for the implementations or (2) were applied to some nearby area or a company with very specific properties that have no analog in the current target settings. The former case can be attributed to the generally top-down approach of academic research; even if the original work was based on some concrete results at a certain company, the generalization caused information loss, and the usual lack of deeper concepts (and examples) makes it difficult to apply those results. The second

case, usually, gives enough details, but again, lacks the deeper concepts. In addition, the availability of IT advancements is adapted slowly and there seems to be a 5 to 10 years IT lag; new IT projects finish within 1–2 years yet they employ 5-year old technologies and approaches. The lag cannot really be attributed to hardware requirements but is rather likely due to budget reasons, conservative management or not keeping the staff up-to-date with trends and modern technology at these companies. Besides, as known in change management, there is usually modest to significant resistance to new technologies and approaches that aim to improve processes, performance and transparency; the fear and/or actuality of downsizing when such IT improvements go live is a significant contributing factor. In contrast, experience shows, in accordance with the social aspects of Industry 4.0 [56, p. 55], that human operators are needed more than ever. One can invent all sorts of clever algorithms but data and information have to be presented to these algorithms in order to be useful. Many of these data and information, for example, process structure, parameters and master data may only exist in the heads of the operators. Thus the former human data-processing is rather turning into a job where the employees are one of the important sources of information and are designers and supervisors of these advanced IT-backed processes.

Therefore, one has to consider these factors and apply the solution in an incremental way and build it from a bottom-up approach. The solutions presented in the dissertation were invented, designed and implemented in this fashion.

The scientific motivation behind the dissertation comes from he need for novel methodologies, methods, models and algorithms that extend or outright replace their former variants because, in the world of (almost) always connected, communicating and computing entities, they will not be good enough or simply will not work anymore at all. The dissertation aims at presenting novelties for a handful of the key indispensable elements in a manufacturing- and logistics IT environment. And certainly, armed with the contributions, solving problems and implementing IT solutions in the future should become easier and the results be more transparent.

Interestingly, my research and contributions presented in the dissertation (and the IT sector in general) have been working towards CPPS in the past 10 years, sometimes knowingly, sometimes unknowingly. The introduction of the Cyber-Physical Systems (CPS) [58] paradigm into the

manufacturing- and logistics environment did not take a sudden move but years of evolution, and the recognition of this being something new. In my opinion, coming up with CPPS-compatible concepts, models, approaches and algorithms are the main tasks of the foreseeable future. Contributing to and filling in the "proper" details of how CPPS should be instantiated is of high importance, especially if one would like to avoid the all too common dead-end solutions IT is so riddled with.

1.3 Structure of the Dissertation

The novel results presented in the dissertation are grouped into two sections which are conceptually organized along a higher-level (share information between human users) to lower-lever (coordinate information sharing between IT systems).

Section 2 – containing **Thesis 1** – focuses on novel, minimal information-model and -exchange protocols between a focal manufacturing company (customer) and its suppliers and one simple objective for this network: there should be no (unplanned) material shortage at the focal customer. Aiming for a wider applicability base, the solution presented in the section defines a small set of expectations towards such logistics setups:

- low amount of already digitalized processes,
- no dependency on any existing, cross-company ERP integrations or data exchanges,
- high importance of ensuring privacy between suppliers that are competitors in other areas, and
- trust through transparent information exchange between the focal customer and each individual supplier.

The novel results were implemented by the author in a new software solution called **Logistics Platform** as part of an academia-industry research and development proj-ect involving a Hungarian manufacturer of incandescent light bulbs and its few dozen component suppliers (providing the packaging material, glass, filament, cap and wires[11]). In exchange for the direct access to high quality component demand and end-product forecast of the focal customer, the suppliers were expected to provide a daily or more frequent information about their delivery and internal

[11] https://en.wikipedia.org/wiki/Incandescent_light_bulb#Construction

production schedule to match said demand. In addition, the platform gave the option to the suppliers to monitor the inventory and consumption levels of their supplied material and take initiative, **react** to events and changes at the focal manufacturer by sending more in case of a projected shortage or withhold shipments in case of an oversupply. The developed software went live in November 2007 and was in operation until the company stopped manufacturing light bulbs and turned into a logistics/redistribution center around 2015.

Although the solution in the previous paragraph was mainly aimed at human operators, the sheer amount of information regarding thousands of components, millions of records and tens of millions of events per day put a high stress on the information systems involved in the logistics processes. Although existing technologies can be employed to handle the problem at the given scale, the associated cost becomes quite visible in the Cloud-era and the mandatory company growth will increase these costs as well. If the cost of such operations are in the millions of US dollars, a 10% performance improvement is quite a saving from the view of a typical IT department.

Researchers at Microsoft (around 2009) and an unrelated industrial collaboration (around 2015) shifted the viewpoint from which such high volume information flows can be built, run, analyzed and by revitalizing the **reactive programming paradigm** along with **functional-** and **declarative programming** the benefits of systems constructed along these approaches are now available to a wider audience starting from mobile app developers up to cloud-solution providers.

Section 3 utilizes this new viewpoint and applies its concepts in a novel framework to support high volume near-real-time information exchange regarding logistics events in a so-called hub-and-spoke network. However, shifting and/or rewriting software based on the reactive paradigm is not enough if the partners participating in the exchange are part of a heterogeneous and dynamic network of companies. Since one can't expect that there is a dominant player who can force its IT solutions on the others, properties of such information exchanges, such as

- the meaning and structure of the data,
- the correctness of the services working with or providing the data,
- the timely, economic and efficient exchange and processing of the data, and

- the entering and leaving of entire organizations and the effects of such large scale change on the information processes of the network.

have to be managed on a higher level than the programming languages and existing frameworks the IT solutions usually employ. To address these complications, a novel, three part solution has been invented and presented in the section: **Thesis 2** describes a novel canonical representation and algorithms about analyzing and describing data structures which allows matching the so-called data types coming from the heterogeneous sources and/or partners; **Thesis 3** allows specifying and verifying generic and specific data processing networks supporting the services of the logistics network's IT solutions; and **Thesis 4** organizes the previous two solutions and an underlying reactive programming library (of which one implementation has been developed) into a framework and runtime environment to facilitate execution and re-definition of some or all parts of the processing network in order to match an support the logistics network it accompanies. The three parts were designed to be able to work relatively independently of the other parts and define a clear service interface towards each other. For example, the type-inference algorithm of Thesis 3 can work with a non-structural type system service provider instead of the canonical and structural type system of Thesis 3. The underlying reactive processing and communications library supporting Thesis 4 can be also replaced with a more modern (and since then industry standard) library solutions developed by other projects.

The three elements of the proposed novel approach have been implemented by the author in a new software solution called **ADVANCE Flow Engine** as part of an EU research and development project (2010–2013). The host for the pilot system was a nation-wide UK logistics company working as the central re-distributor in the aforementioned hub-and-spoke style network with tens of thousands of parcels per day and millions of parcel- and transportation tracking events coming from within its main warehouses and from the almost weekly changing set of hundreds of independent transportation companies (depots) servicing various postal regions of the UK island. The ability to process and display information more transparently and in near-real-time in comparison to the previously existing paper–phone–legacy software solutions has been estimated to save £500,000 per year on IT costs, avoiding transportation vehicle over- and under-utilization (by using prediction algorithms driven by the data coordinated by the Flow Engine) and faster unloading

and loading such vehicles in the warehouse by showing current and predicted parcel amounts in the various storage areas inside the warehouses. (As far as the author knows, the software solution has not been put into live usage after the project ended and the focal company has instead purchased an undisclosed off-the-shelf complete management solution.)

Finally, **Section 4** lays out further research possibilities and shows some work done into these directions.

2 Coordinating consumer–supplier planning via channels with minimal backing data model

2.1 Introduction

The global behavior of production networks emerges from the interaction of local intentions and actions of the partners. Supply planning is considered in a production network as a distributed effort for matching future demand and supply under continuously changing conditions. Even though decisions are made locally in an autonomous manner, partners in a production network should act in a concerted way.

Given their business goals, market and the production environment that all evolve in time, partners have to reason over their future courses of actions by considering to some extent also the others' situations and intentions. The problem boils down to distributed planning: network members would like to exercise control over some future events by relying on all kinds of information they have at hand. Some of this information can be considered certain, such as those related to products, production technologies, resource capabilities, or sales histories. An essential part of the accessible information is, however, incomplete and uncertain, such as those items capturing forecasted demand, or expected resource and material availability. In addition, further difficulty arises when the information is outdated, late or contains errors due to the way it was recorded in the system in the first place.

Industry strategists and academics – while sketching alternative trajectories for technological and organizational developments – agree alike that resolving incompleteness and uncertainty by proper information exchange is a matter of survival for any production network that is to operate under volatile market conditions.

However, uncertainty and the lack of information is only one side of the coin; different, though equally hard problems ensue from the plethora of information. When preparing the ground for informed planning decisions, an enormous amount of behavior related – i.e., dynamic – data must be handled, synchronized, cleared, filtered, aggregated and archived. The decision complexity of planning processes can but grow with the extension of input data, which is in sharp conflict with the requirement of giving timely, almost instant responses to queries during interactive planning sessions.

Production Informatics has well-proven approaches to handle uncertainty and structural complexity. Aggregation merges detailed information on products, orders, demand forecasts, production processes, resource capacities, and time. Various planning problems – such as production scheduling, production planning or master planning – are formulated by merging more and more details on longer and longer horizons [83]. Solutions are generated in a hierarchical planning process where higher-level solutions provide constraints to lower-level problems.

Hence, according to their horizon and detail, plans can have strategic, tactical or operational dimensions. At the same level, decomposition separates planning problems into easier-to-solve subproblems. This is the case when, e.g., on the tactical level production planning is separated from supply or distribution planning.

The evolution of planning functions in production management resulted in a generic hierarchical planning matrix [44, 90]. Figure 1 shows typical planning functions on the strategic (long-term), tactical (medium-term) and operational (short-term) level organized along the main flow of information and materials. These functions are more or less common at each node of a production network, though, of course, manifest themselves in different forms and complexity.

Within an enterprise, the coordination of segregated planning modules is a grave problem in itself [44, 67]. However, this issue is even more critical in a production network where proper information exchange is the primary precondition of the collaboration between the partners [65]. Usually, the type and amount of such interactions can be described on three levels. **Coordination** is when the partners work towards a shared goal but they also retain most of their autonomy. During a **cooperation**, the partners work more closely and often share their resources as well. Finally, in **collaboration**, the partners decide on a strategic level that they will work together very closely towards the shared or overlapping goals.

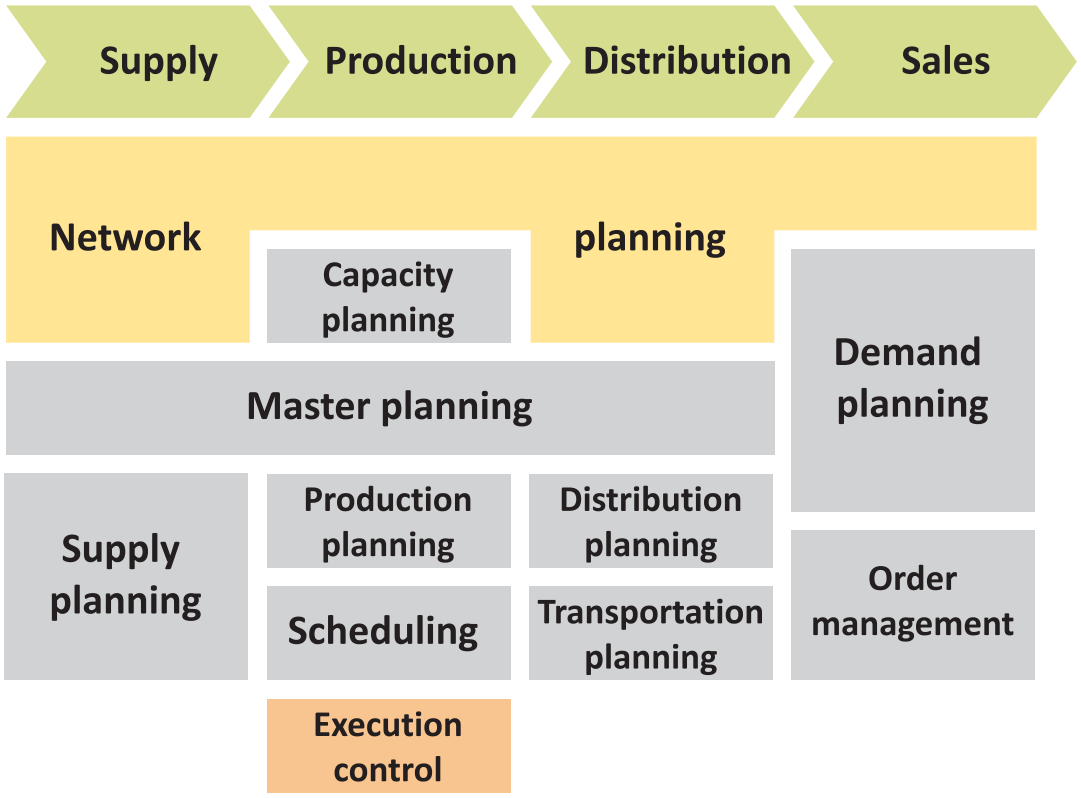


Figure 1: Matrix of strategic, tactical and operational planning functions [4] p. 298 after [44].

While the theoretical aspects of coordination (i.e., personnel and information systems working well together) and cooperation (personnel and information systems exercising common efforts towards a shared goal) in production networks have been investigated extensively for a long time, these studies have paid due attention neither to the differentiation of planning functions, nor to the underlying causes. Here, one can but remark that the surprisingly low number of deployed manufacturing applications of agent technologies even on the 'ideal' field of supply network management [71] – is also a symptom of this lack of focus.

The motivation here is in bridging the gap between the theory and practice of coordinated planning in production networks. The particular background to this work was a national industry–academia research and development (R&D) project that was aimed at improving the performance of a network that produces customized mass products [74, 95].

2.2 Problem statement

There are two relevant problems to be investigated by the dissertation in such consumer-supplier linked networks:

- shortage due to lack of one or more material components from one or more suppliers: the manufacturing process is stalled once the inventory runs out at the focal manufacturer, causing delays for shipping the end product further along the chain, and
- excessive component (safety) stock retention, again, at the focal manufacturer (customer), which increases the overall cost of the manufacturing and opens it to risks such as product runouts (components cannot be reused for other products) and limited shelf-life of the components themselves.

The trivial high level solutions are, for example, to perform (a) centralized planning with little-to-no autonomy left to the suppliers, or (b) coordinated planning with shared, transparent information distributed between the partners. Depending on the particular set of companies, option (a) usually requires a dominant company – often the focal manufacturer – to express its powers and enforce process and IT solutions on its suppliers. Option (b) requires sharing information more gratuitously among partners with arbitrary local IT supporting the process itself of which requires way more design effort than the classical periodic batched ordering of components.

The CPPS paradigm also favors option (b), however, new complications arise when the participating companies have not or have only partially invested in digitalizing their existing processes. In addition, digital information interchange between companies today is still a technological and IT-specific challenge requiring process- and software-designers at both sides of a supply line to work out a common solution. This coordination-, digitalization- and interoperation-enhancement process should be performed with all the other suppliers as well.

The third problem – experienced by the author in practice – is the general distrust in digitalized and automated processes at companies that otherwise were performing tasks related to the logistics activities towards the focal manufacturer via manual, offline (paper) or local IT solutions (speadsheet, local database). For them, any complicated solution would trigger increased resistance and depending on the supplier company's

scale, the cost of adapting of-the-shelf solutions would be a significant burden. If there was no prior IT system, the introduction of any new solution is expected to be as non-invasive as possible without compromising its functionality and the common goal of resolving the shortage/inventory problem mentioned before. The secondary source of distrust is the fear of losing privacy towards other suppliers that may be competitors in other networks.

The fourth problem is that without established digital and transparent data interoperation between the participants, modern planning algorithms cannot produce quality results as they have to rely on some forecasts, often derived from other forecasts, based on historical data from a previous stage of the supply chain, which increases the uncertainty of the end result – the timely and quantitatively successful delivery of the end product. A more reliable information source is the exact production schedule of the customer from which the component demands are generated and submitted to each supplier.

In summary, the problem of shortage/excess inventory at the focal manufacturer can be addressed by sharing detailed production information between the customer and each individual supplier, so that:

- information remains private between suppliers,
- both side share more information, such as their detailed production schedule to the other side
- information is symmetrically accessible to both sides
- the information sharing happens digitally,
- complexity of the system facilitating the information remains low, the applied (data) model starts out simple, enabling easier entry for digital newcommers,
- protocol(s) are established to supply and handle the shared information and act mainly according it, and
- there is a feedback loop, in the form of key-performance-indicators or other performance evaluations, also digitally presented and clearly linked to the original information exchanges, that show how well **both sides** performed their duties.

2.3 Literature review

Research of coordinated supply planning goes back to inventory management where the main questions are when to order and how much to order. Typically, centralized channel coordination models have been developed where one of the partners had all the information available to make optimal decisions. The **channel** in this context and in the dissertation represents the logical and physical flow of the material and information of a specific component, which is handled largely independently from the rest of the channels by the partners. The centralized approach is, however, hardly realizable in practice, owing to the supply chain partners being separate, autonomous business and legal entities. Instead, upstream planning is the most widespread form of collaboration at the moment [22]. Since in reality, no partner can control the chain, let alone a complete network, there is an increased interest in decentralized control, both in deterministic and stochastic settings [27].

Actual investigations take mostly the approaches of game theory and economics with asymmetric information between the participants. In a real network there is always an information gap between the partners: the supplier is familiar with the production and setup cost for the components, while the end product manufacturer (in the customer's role) can estimate better the finished goods demand. This demand is distorted by the internal planning processes: normally, master plans are generated which are further refined into production plans and schedules.

When lot sizing decisions are made and parallel component demands are aggregated, the resulting actual component demand forecast can hardly be related to the original finished-goods forecast (see Figure 2). Although, in the age of electronic information exchange, this gap could be bridged easily, partners do not have incentives for sharing private business information [41, 95].

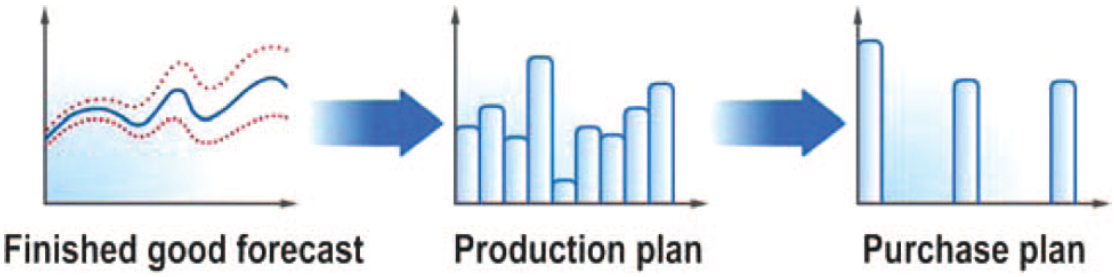


Figure 2: Transition of demand forecasts [4] p. 299.

On the practical side, there exist various information sharing solutions that support order processing in production networks. One example is the centralized SupplyOn platform for automotive and manufacturing industries, developed by several European automotive part suppliers[12]. It uses Electronic Data Interchange (EDI) as a basic format but also supports WebEDI, which technically requires only Internet access [66]. A similar approach called myOpenFactory [86] proposes a centralized information sharing agency.

The past performance of channels is evaluated from both perspectives. Results can be aggregated and presented for any past period. The traditional, order-based supply has standard performance evaluation techniques that measure the fulfillment of orders on the one hand, and the inventory or overall logistics costs on the other hand [48]. In cooperative supply however, the responsibilities are more complex because the decisions of any partner may propagate throughout the network. Certain kinds of the uncertainties that emerge from an unpredictable market environment or from the system's properties are hard to avoid (for a categorization, see [75]).

However, in a production network the other members' operation is also a source of uncertainty which is often charged by factors that are, in fact, unnecessary. For example, when the supply planners of the customer are measured by the material shortage, then they tend to inflate the demand and forward too optimistic plans towards suppliers.

On the contrary, if the planners are rewarded for over-performing their plans, then they deliberately underestimate the demand and share too pessimistic plans with the suppliers. The suppliers can be aware of the biases and may not accept the demand information without critique. They rather tinker the forecasts based on their own past experiences, but this cannot completely restore the quality of the forecast. This way the network as a whole operates far away from its main objectives: while the service level is corrupted (and is to be restored time and again by urgent orders at the price of increased system nervousness and additional costs), large, even obsolete inventories may also accumulate. All in all, the selfish distortion of information necessarily decreases the performance of the network. Hence, the performance of all partners has to be measured with a kind of deviation between their promised and actual activities [4, p. 303–305][30, 45].

[12] http://www.supplyon.com/, last accessed: July, 2015

As of lately, various paths to coordinated – and, eventually, even to cooperative – planning in production networks are subject to extensive negotiation. Together with related research [62], the following general roadmap was suggested for such studies [74, 95, 96]:

1. Development of processes and establishment of media for sharing information about the actual and expected situations, as well as of the future intentions (i.e., plans) of autonomous network partners.
2. At each level of aggregation, development of powerful local planners and schedulers which are able to solve the richer, extended models.
3. Design and set up of incentive mechanisms that facilitate truthful information exchange, the sharing of risks and benefits, and cooperative behavior, after all.

The solution presented in the dissertation aims at providing a foundation for enabling step (1), on which the subsequent steps listed can be built as extensions, in face of the requirement for simple structure, protocols, bounded exposure of private information, transparency, and low-bar entry for less IT focused suppliers.

2.4 Solution approach

The solution consists of four major components to be implemented in software **platform** accessible over the network (local or via virtual private network):

1. a **data model**, which allows storing the information provided by both sides over time and with a limited set of entity types,
2. a **set of protocols** describing the information to be transmitted between and the actions to be taken by each side,
3. a set of **filtering tools** to let the users work on channels that need attention based on user-defined criteria, and
4. **data population and evaluation algorithms** that can automatically fill in details (such as production and delivery timing and quantities) and calculate the past performance of both sides (i.e., supplier's service level vs. customer's demand/forecast stability).

2.4.1 Minimal model required for the coordination protocols

When designing and implementing software that will require persistent storage of data, usually the first step is to specify the entity model of the underlying structure of the data to be handled. A decade ago, this meant declaring the relational database table model, but as of lately, there exist a trend for designing for non-relational persistent storage or even designing for service-oriented storage where accessing data of an entity (table) requires only a parameterized web-service call where the actual storage details are hidden (and usually irrelevant). For the sake of describing the solution in a concise fashion (and because the conventional database terminology is generally more widely known), the entities will be referred to as **tables** in this section.

To recap the problems and requirements for coordinated planning from section 2.2, the underlying data model for any solution should be as small and compact as possible yet still enabling the sharing and evaluating of the relevant information. In addition, limiting the complexity allows the participating companies' own IT staff to evaluate and audit the security aspects of the solution and match the properties with their own privacy/risk matrices.

The data model that stores the collected data (over time) from each participating company and certain related settings (relatively constant) that forms the information base for the protocols (2.4.2) can be seen on Figure 3.

Channels The core table in the minimal data model is `Channels` (Figure 3, central block with yellow background) which represents (virtual) flow of a concrete material component between the focal manufacturer and each supplier involved. It defines an unique identifier (`ID`) that uniquely represents a specific material component (`ComponentID`) between the focal manufacturer (`CustomerID`) and (one of) the supplier (`SupplierID`) of that component. Since the product mix of the focal manufacturer can change over time, certain channels have to be closed an marked inactive (`Active`), therefore, it no longer participates in the day-to-day coordination process and evaluations. As the platform has to support certain level of automation and evaluation of each channel, the suppliers are expected to share their detailed production schedule and local inventory levels. The platform then can consider the many types of

inventories and given the customer's detailed production schedule, the changes in the inventor ylevels can be projected over time (for example, see Figure 12). If the projected shortage only manifests at the customer and the supplier has some of all of the required component amount in its local inventory, the platform can consider the time it takes to deliver that amount (`DeliveryLeadTime`) over to the customer and include it in the projection Hovever, if not the supplier has the required component in stock at that time, it has to start its own production process to match the demand. This naturally takes time and the platform requires a hint about how much time this production alone (on average, per unit quantity) takes (`ProductionLeadTime`).

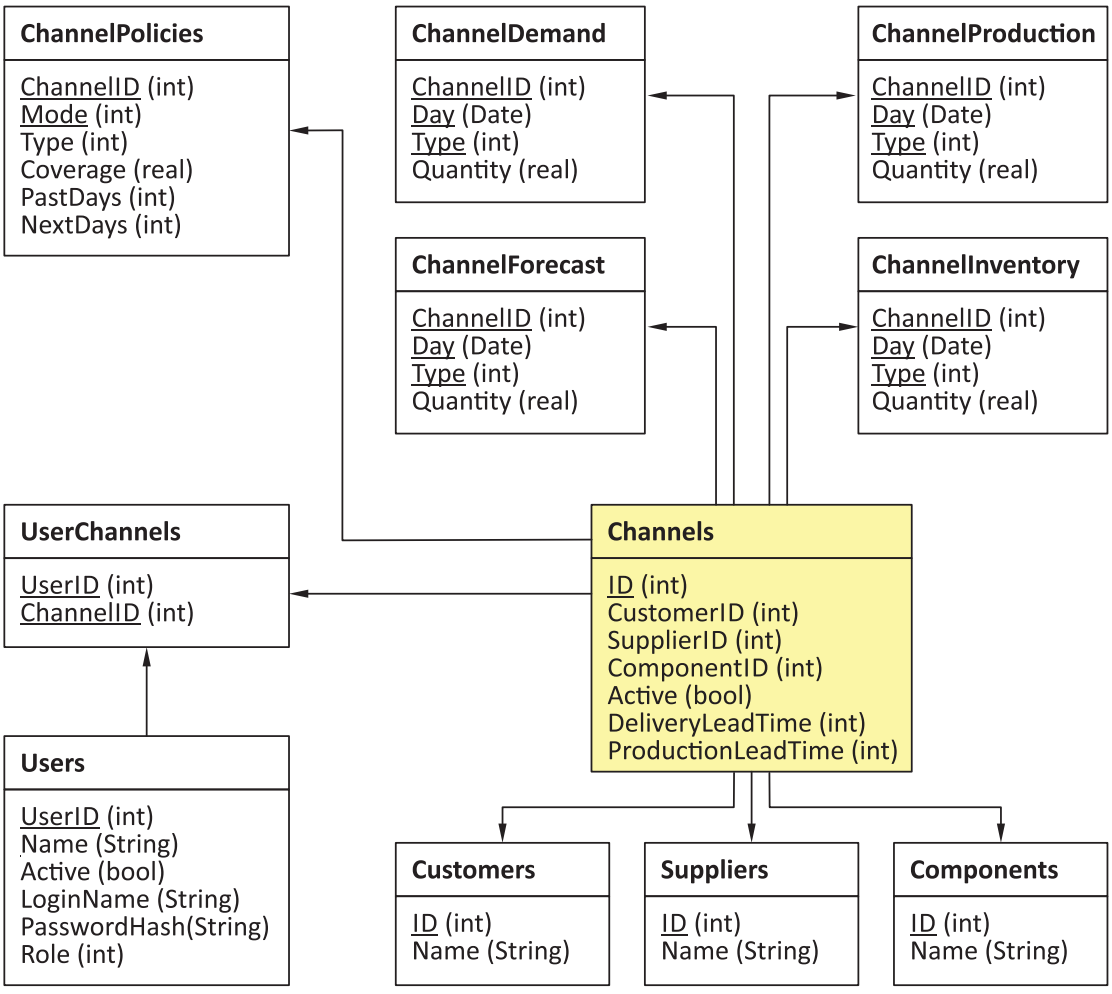


Figure 3: The data model and relationships between the entities of the platform

Orthogonal to this, a channel may represent operaton several modes (manual, automatic) and on several horizons (short-term, medium-term), each four combinations having separate set of demand, forecast, produc-

tion and inventory details, selected by the `Type` attribute in their respective tables (see the paragraphs later in this section).

This type of organization of identifying a channel, in concept, allows describing networks that involve multiple customers to and multiple suppliers of the same material component. In addition, it can represent the suppliers as being the consumers of their input material components forming a graph of the material flow between the participants. In practice, such networks consists of that many competing companies that unlike the setup with only one focal manufacturer, there needs to be a third party that hosts the platform, which has to be trusted by all and which should uphold the privacy requirements of not sharing information across suppliers. Thus, `CustomerID` can be considered either constant or may represent destination sites of the same company where the material components should be delivered.

Customers, Suppliers, Components The tables `Customers`, `Suppliers` and `Components` hold the human-readable names (textual descriptions) of the participants and the material component a channel references via its `CustomerID`, `SupplierID` and `ComponentID` respectively. In practice, these tables may hold additional information beyond the aforementioned textual descriptions, such as global properties for said entities, identifiers used in other information systems, etc. In addition, this allows the set of suppliers and components (and the extra properties) to change over time without affecting the main `Channels` table.

Users, UserChannels The platform is used by humans which have to login into the platform. This limits which particular channels they can see and interact with and thus ensures and enforces the privacy requirements mentioned before. For this, each user has an unique identifier (`UserID`) and a (convenience) name (`Name`) to display on screen (i.e., who modified a particular channel's details last time) and in the application/audit logs, a login name (`LoginName`) that identifies the user on the platforms login page.

This standalone login and user identification process is provided because although most companies have their own authentication and authorization services (such as Single-Sign-On, SSO; Lightweight-Directory Access Protocol, LDAP), those are often not shared or open to external

partners. Therefore, the platform provides its own authentication via the traditional `LoginName` and `PasswordHash`. In theory, more complex authentication solutions could be supported (such as certificate-based, OpenID-based, Trusted Platform Module-based), however, that would significantly raise the bar for entry into the network due to additional IT costs of setup at the supplier. The proposed solution only requires a modern web-browser for example. The `PasswordHash` is suggested to be calculated via bcrypt[13] with at least 65000 rounds.

The authorization is established via the `Active` flag that allows disabling user accounts, the `Role` flag that determines the user exclusively represents a customer, a supplier or an administrator. Both the customer and supplier user role requires a set of channels to be associated with the users via the `UserChannels` table. The platform ensures a customer user may interact with channels from multiple suppliers in general but a supplier user cannot interact with channels associated with other suppliers. The role of the administrator is to add new users and establish the correct association. The platform provides safety measures so the administrator cannot associate a supplier user with channels that belong to other suppliers.

ChannelDemand The main driving information of the platform is the component demand on each channel as provided (truthfully) by the consumer. In practice, this is the aggregated material component demand across all products of the customer that use said component. In case there are multiple channels and thus multiple suppliers for that same material component, the demand is split and spread across those channels. There are various ways to distribute the quantities which, by default, is not in the scope of the platform and the dissertation. In practice, the rules are company and/or material component dependent and require custom development. Needless to say, any such algorithm may require information about the supplier's status which is dutifully available and provided, as part of the transparent and symmetric information sharing nature, by the platform in the other tables listed further down in this chapter.

The default resolution supported by the platform, and thus batching window, is a calendar day. The table hosts the daily (`Day`) demanded component quantity (`Quantity`) of each channel (`ChannelID`).

[13] https://en.wikipedia.org/wiki/Bcrypt

Since the channel can represent two operation modes (manual, automatic) and two horizons (short, medium), the `Type` flag selects to which the particular entry belongs to. In practice, the medium term demand usually represents a combination of the aggregated daily resolution component demand followed by the aggregated demand over the subsequent weeks based on planned (but not forecast) production of the end-products. The platform may allow the changing of the quantities manually by the consumer users, but it is expected the data is provided to the platform in an automated way from a production scheduling solution (such as [5]). The supplier user associated with the channel can read this information but not modify it.

ChannelForecast The `ChannelForecast` has the same structure as the `ChannelDemand` and is provided by the customer as well, however, the `Quantity` represents the aggregated component demand derived directly from the forecast of the end-product that requires it. The forecast itself may consists of predicted amounts and quantities of known end-product demands which don't have their due date of confirmed for example. The generation of such quantities are beyond the scope of the platform and the dissertation and is usually the role of the Enterprise Resource Planner (ERP) of the customer. The supplier user associated with the channel can read this information but not modify it.

ChannelProduction The `ChannelProduction` table holds information about the supplier's plans on producing a material component in certain quantity in case the customer and the supplier both run out of the material component in their inventory. Structurally, the table is similar to the previously mentioned `ChannelDemand` and `ChannelForecast` tables. It is expected to be provided on a daily basis by the supplier either via manual data entry or (automatically) uploaded through a service endpoint provided by the platform. The consumer user associated with the channel can read this information but not modify it.

ChannelInventory The `ChannelInventory` table represents the available quantity (`Quantity`) of the inventory at various places at a given day for a given channel. The location of an inventory indicates

how soon the material component is accessible to the customer and supports the following types (`Type`) by default:

- **Customer inventory on hand**: represents how much is available right away at the customer's manufacturing site.
- **Supplier inventory on hand**: represents how much is available right away at the supplier's which can be called in by the customer within the delivery lead time (`DeliveryLeadTime`).
- **In transit**: the amount already in transit and is expected to become available the next day.

The inventory information is shared among the channel's operation modes and form the initial quantity amount of the material component when the inventory level calculation projects its consumption and replenishment over the given horizon (i.e., days for short-term, weeks for medium-term).

In practice, the companies and tracking of the customer's and the supplier's inventories as well as the in transit amounts may be subdivided to other types, such as demonstrated in the top-right table section Figures 12 and 13:

- **Carried forward**, which tells indicates the non-consumed material component amount from the previous day, relevant to the consumer; and
- **Consignment supplier**, which represents the amount available to the customer from the warehouse at the customer's site which warehouse technically belongs to and is managed by the customer.

Similar to the information provided to the platform for the `ChannelDemand` table, the inventory amounts of the customer, since that information is required for its scheduler system already, is expected to be automatically shared and updated by the customer's other information systems.

The supplier-specific inventory level information can be manually entered by the user or automatically retrieved from the supplier's system if possible.

Note in general, that such automated access to data sources at the suppliers require case-by-case development work on the platform (i.e.,

establishing a webservice endpoint at the platform or at the supplier and let one call the other with the data). An alternative option, although more complicated, is to use the reactive dataflow engine solution from Thesis 4 for the data interoperation layer of the platform.

ChannelPolicies The `ChannelPolicies` table specifies the per-channel (`ChannelID`), and a composite flag `Mode` representing the operation mode (manual, automatic), the horizon (short-term, medium-term) and the bound type (lower, upper) of the channel's projected inventory over time.

The platform supports several ways to determine the inventory quantity limits (`Type`):

- **Fixed quantity**, the traditional exact amount that the inventory level on a given day should go below or exceed it. The amount is specified via the `Coverage` field.
- **Past coverage**, given a number of past (working) days, specified by `PastDays`, the inventory level should not go below or exceed the average demand quantity over this period of time.
- **Forward coverage**, given a number of next (working) days, specified by `NextDays`, the inventory level should not go below or exceed the the average demand of those subsequent days.

By interpreting the `Type` flag as a binary string that enables each type of coverage calculation mode, composite policies can be defined where the final evaluation on whether an inventory level on a particular day goes out of the bounds is determined by being within the bounds computed by each of active methods listed above.

2.4.2 Coordination protocols for two automation types and two horizon sizes

The platform is hierarchical (see Figure 4) and the high level idea behind it aims at the so-called Vendor Managed Inventory concept [36]. Instead of the customer issuing orders for a particular material component, the suppliers are provided with detailed plans for consuming said material component, and are expected to act proactively by monitoring the inventory levels themselves and issuing deliveries or local production on

their own whenever there is a shortage detected by the platform. In practice, the legal requirements are that the customer still has to administer a regular component order (but this can be in response from the supplier notifying about the upcoming shortage). In a future scenario, such fill-ins may be governed by special aggregate contracts, avoiding the need for issuing individual orders on a daily basis.

The platform operates on two types of planning horizons. On the **short-term horizon**, the supplier meets the exact, short-term component demand of the customer. This demand is generated from the actual daily production schedule of the customer in form of call-offs and can be satisfied only by direct, just-in-time delivery from an inventory. This short-term demand of the customer is responded by the supplier's actual delivery schedule. Decisions are made on a daily basis, on a horizon of 1 to 2 weeks. With this short look-ahead, demand uncertainty is hedged by safety stocks.

On the medium-term horizon, the supplier has to make preparations for satisfying the short-term demand of the customer. Hence, the supplier receives medium-term demand forecasts of components from the customer, together with some information about the reliability of forecasts. Managing inventories, deciding about the periods and optimal lot sizes of production is the supplier's responsibility. The demand forecast of the customer should be acknowledged by the supplier, either by simply accepting/rejecting it or by responding with an appropriate production plan. On this level, decisions can be made in a longer (weekly) cycle.

For each channel, there is a complex inventory composed of the in-transit as well as of the on-hand inventories at the supplier and the customer. The platform keeps track of the inventory items on a daily basis. However, as Figure 4 shows, inventory is a passive element whose level is influenced by the local decisions of the supplier (who builds up the inventory) and the customer (who consumes the inventory). The platform collects, presents, analyses and aggregates all relevant information concerning the future, present and past of the channels. Hence, each channel has various dynamic future-related information, such as forecasted demand, open orders, scheduled demand (generated by the customer), production plan and delivery schedule (generated by the supplier). Departing from the actual inventories, projected inventories are calculated both in the medium and short term, and inventory statuses are evaluated

from both partners' point of view.

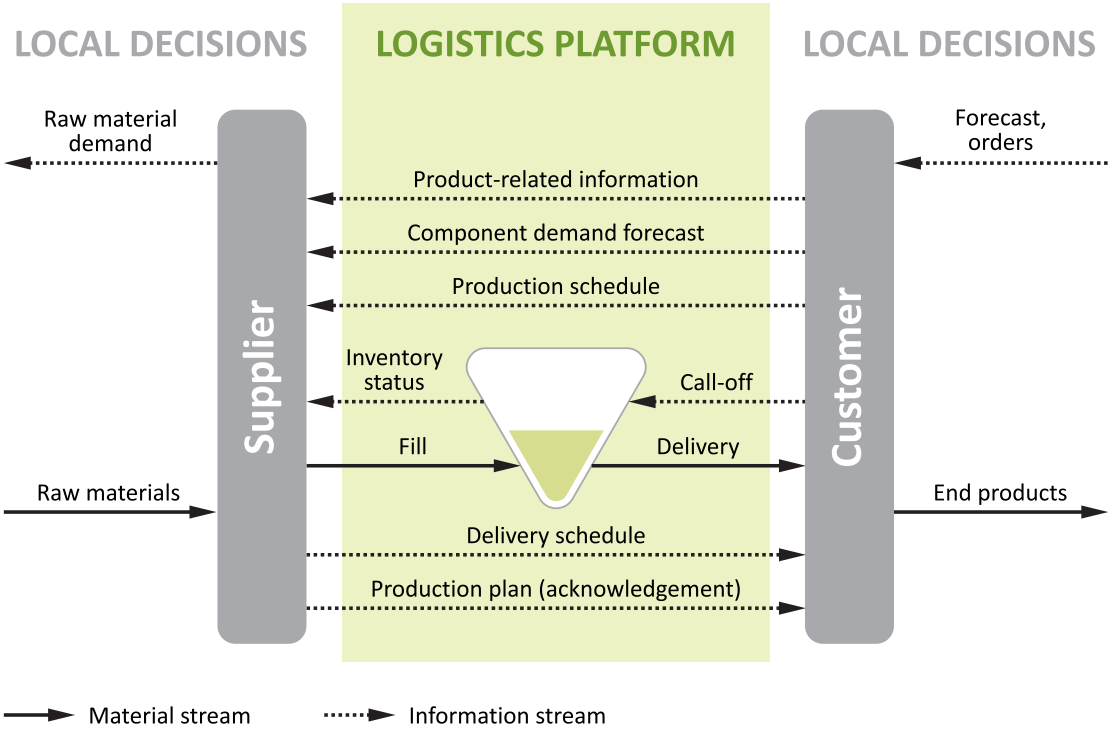


Figure 4: Information flow through a logistics platform [4, p. 299].

The following paragraphs detail the interaction protocosl between the customer user (or backing IT system), the platform and the supplier user (or backing IT system) for the combinations of two planning horizon lengths (short-term, medium-term) and two automation levels (manual, automatic).

Protocol for exchanging information in short-term manually controlled channels The protocol, see Figure 5, is exercised on a daily basis and has the aim to inform the supplier about the exact component demand for the upcoming few dozen workdays so the supplier's user can respond with confirmation, manually entered into the platform, about its ability to deliver said component or make preparations on its end to produce more. The following steps are executed:

1. The planning/coordination starts at the beginning of a work day (i.e., 9 am) after the detailed production schedule for the current and upcoming days have been generated by an external system.
2. The daily demand, the current day's customer inventory and the new policies for the particular channel (if changed) are uploaded into the platform.

3. The platform stores the data from step (2), calculates how the inventory amounts will change over time, then notifies the supplier user about the availability of the information.
4. If the supplier's on-hand inventory doesn't cover the demand, the supplier will perform its own short-term production for the component.
5. Given the available inventory and the plan for components to be produced fresh by the supplier, the short term delivery schedule is generated by the supplier.
6. The supplier uploads its own daily component production schedule, its on-hand inventory at the start of the day and the quantity of components that should arrive at the customer on the current day (in-transit).
7. Given the updated plans and inventories of the two sides, the platform recalculates the inventory amounts over the horizon and applies the channel policies set by the customer to identify shortage or excess of the material component.
8. The evaluation of the results and policy violations on particular days are presented to the supplier.
9. In case of the detected policy violations (shortage or excess), the supplier can adjust either its own production schedule – step 4 – and/or adjust the delivery schedule – step 5 – according to the quantity discrepancy detected by the platform.
10. Otherwise, the supplier plans are marked as committed (and final for the day) by the supplier. Note that depending on the circumstances, some policy violations may still be present at this point.
11. Given the committed quantities of the supplier, the consumer is notified about the inventory status and the planned production of the channel's component at the supplier. Given this component production schedule, inventories and lead times specified by the channel, the scheduling and/or planning system at the customer can be updated. In case of any remaining unresolvable shortage, these systems can then reschedule the production of the end-product if necessary.
12. The customer then either accepts the outcome or initiates exception handling, which is outside the scope of the platform.

13. The supplier receives the acknowledgment or enters an exception handling mode as well. This mode, in practice, involves exchanging emails and/or phone calls trying to resolve the (usually shortage related) problem.

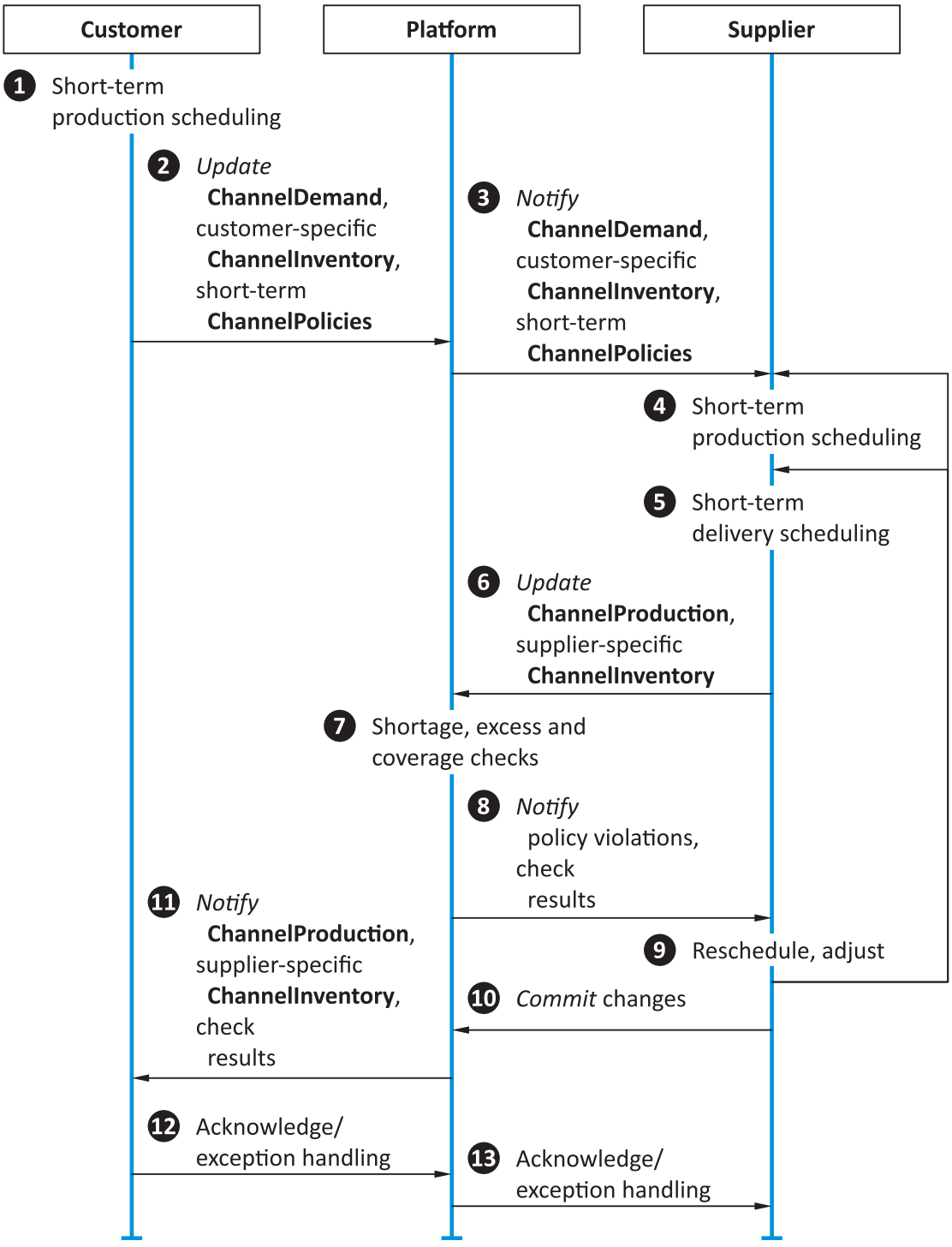


Figure 5: Protocol for exchanging short-term information regarding a manually controlled channel.

Protocol for exchanging information in medium-term manually controlled channels The protocol, see Figure 6, is exercised on a weekly basis and has the aim to inform the supplier about component forecast derived from the customer's primary forecast for the end-product it is used, for the length of several weeks in advance. The supplier can respond with its own plans for producing the material component in questions aggregated up to weekly levels. The following steps are executed:

1. The customer performs its weekly, medium term planning (via MRP or by other means) and generates the aggregate material component quantity (forecast) for each week.
2. The customer uploads the component forecast, its inventory status at the beginning of the week and adjusts the channel policies if necessary.
3. The platform notifies the supplier about the new forecast quantity, inventory and policy changes of the channel.
4. The supplier performs its medium-term production planning based on the provided material component forecast in the previous step.
5. the channel updates the medium-term production plan for the material component inside the platform.
6. Given the updated plans and inventories of the two sides, the platform recalculates the inventory amounts over the horizon and applies the channel policies set by the customer to identify shortage or excess of the material component.
7. The evaluation of the results and policy violations on particular weeks are presented to the supplier.
8. In case of the detected policy violations (shortage or excess), the supplier can adjust either its own production schedule by going back to step 4.
9. Otherwise, the supplier plans are marked as committed (and final for the week) by the supplier. Note that depending on the circumstances, some policy violations may still be present at this point.
10. Given the committed quantities of the supplier, the consumer is notified about the planned production of the channel's component at the supplier.
11. The customer then either accepts the outcome or initiates exception handling, which is outside the scope of the platform.

12. The supplier receives the acknowledgment or enters an exception handling mode as well. This mode, in practice, involves exchanging emails and/or phone calls trying to resolve the (usually shortage related) problem.

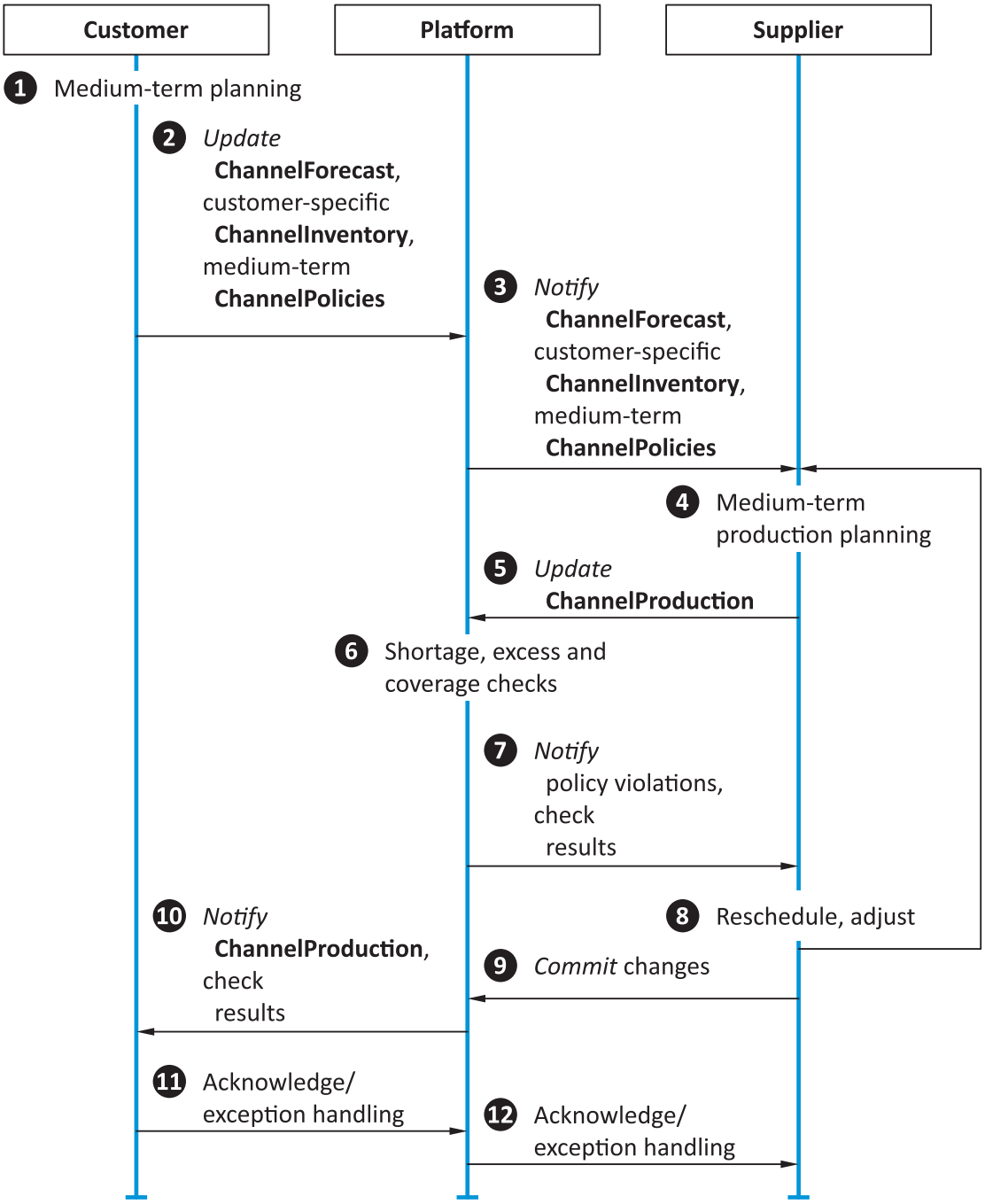


Figure 6: Protocol for exchanging medium-term information regarding a manually controlled channel.

The important difference between the short-term and medium-term protocols is that there is no delivery planning required by the supplier at this latter level.

Protocol for exchanging information in short-term automatically managed channels The protocol, see Figure 7, is exercised on a daily basis and has the aim to inform a planning **IT system at the supplier**, that is capable of scheduling production and delivery autonomously, about the exact component demand for the upcoming few dozen workdays. The main difference between the manual and automatic channel management is that the supplier's response is delivered via digital information exchange rather than manual user input. The second difference is that the automated operation requires a legal framework that requires the supplier to respond to the exact component demand and inventory evolution over time proactively (supported by a contract with coarse-grained quantity bounds over a longer period) instead of waiting for component orders from the customer. The following steps are executed:

1. The planning/coordination starts at the beginning of a work day (i.e., 9 am) after the detailed production schedule for the current and upcoming days have been generated by an external system.
2. The daily demand, the current day's customer inventory and the new policies for the particular channel (if changed) are uploaded into the platform.
3. The platform stores the data from step (2), calculates how the inventory amounts will change over time, then notifies the supplier user about the availability of the information.
4. If the supplier's on-hand inventory doesn't cover the demand, the supplier will perform its own short-term production for the component.
5. Given the available inventory and the plan for components to be produced fresh by the supplier, the short term delivery schedule is generated by the supplier's system.
6. The supplier's system uploads its own daily component production schedule, its on-hand inventory at the start of the day and the quantity of components that should arrive at the customer on the current day (in-transit).
7. Given the updated plans and inventories of the two sides, the platform recalculates the inventory amounts over the horizon and applies the channel policies set by the customer to identify shortage or excess of the material component.

8. The evaluation of the results and policy violations on particular days are presented to the supplier's system.
9. In case of the detected policy violations (shortage or excess), the supplier's system can adjust either its own production schedule – step 4 – and/or adjust the delivery schedule – step 5 – according to the quantity discrepancy detected by the platform. However, the looping should be limited to a reasonable amount.
10. If the policy violations cannot be reasonably resolved in an automated fashion, the process should switch to the manual short-term protocol (steps 4, 5 or 13 of that protocol) requiring human intervention.
11. Otherwise, the supplier plans are marked as committed (and final for the day) by the supplier. Note that depending on the circumstances, some policy violations may still be present at this point.
12. Given the committed quantities of the supplier, the consumer is notified about the inventory status and the planned production of the channel's component at the supplier. Given this component production schedule, inventories and lead times specified by the channel, the scheduling and/or planning system at the customer can be updated. In case of any remaining unresolvable shortage, these systems can then reschedule the production of the end-product if necessary.
13. The customer then either accepts the outcome or initiates exception handling, which is outside the scope of the platform.
14. The supplier receives the acknowledgment and/or enters an exception handling mode as necessary.
15. In exception handling mode, a human operator from the supplier gets involved, who can either redo the planning steps and/or provide different quantities to the platform or tries to resolve the situation in an offline manner. In practice, involves exchanging emails and/or phone calls trying to resolve the (usually shortage related) problem.
16. Depending on performance evaluations (see 2.4.4), higher level customer systems (such as an ERP) and/or the legal contract framework can be adjusted over time.

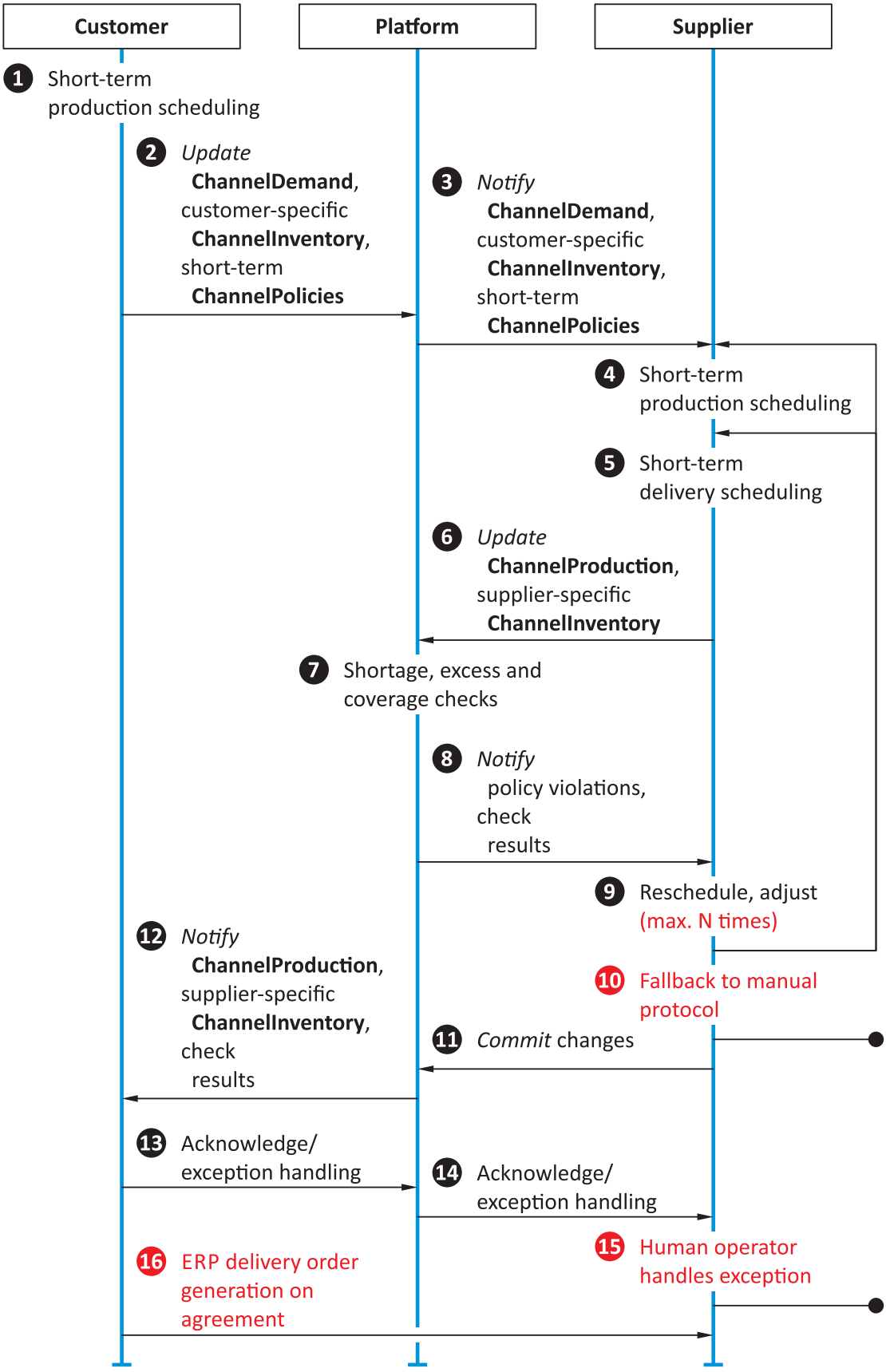


Figure 7: Protocol for exchanging short-term information regarding an automatically managed channel.

Protocol for exchanging information in medium-term automatically managed channels The protocol, see Figure 8, is exercised on a weekly basis and has the aim to inform a planning **IT system at the supplier** about component forecast derived from the customer's primary forecast for the end-product it is used in, for the length of several weeks in advance. The supplier's system can respond with its own plans for producing the material component in questions aggregated up to weekly levels. The following steps are executed:

1. The customer performs its weekly, medium term planning (via MRP or by other means) and generates the aggregate material component quantity (forecast) for each week.
2. The customer uploads the component forecast, its inventory status at the beginning of the week and adjusts the channel policies if necessary.
3. The platform notifies the supplier's system about the new forecast quantity, inventory and policy changes of the channel.
4. The supplier performs its medium-term production planning based on the provided material component forecast in the previous step.
5. the channel updates the medium-term production plan for the material component inside the platform.
6. Given the updated plans and inventories of the two sides, the platform recalculates the inventory amounts over the horizon and applies the channel policies set by the customer to identify shortage or excess of the material component.
7. The evaluation of the results and policy violations on particular weeks are presented to the supplier.
8. In case of the detected policy violations (shortage or excess), the supplier can adjust either its own production schedule by going back to step 4.However, the looping should be limited to a reasonable amount.
9. If the policy violations cannot be reasonably resolved in an automated fashion, the process should switch to the manual medium-term protocol (steps 4 or 12 of that protocol) requiring human intervention.
10. Otherwise, the supplier plans are marked as committed (and final

for the week) by the supplier. Note that depending on the circumstances, some policy violations may still be present at this point.

11. Given the committed quantities of the supplier, the consumer is notified about the planned production of the channel's component at the supplier.
12. The customer then either accepts the outcome or initiates exception handling, which is outside the scope of the platform.
13. The supplier receives the acknowledgment or enters an exception handling mode.
14. In exception handling mode, a human operator from the supplier gets involved, who can either redo the planning steps and/or provide different quantities to the platform or tries to resolve the situation in an offline manner. In practice, involves exchanging emails and/or phone calls trying to resolve the (usually shortage related) problem.
15. Depending on performance evaluations (see 2.4.4), higher level customer systems (such as an ERP) and the legal contract framework can be adjusted over time.

Again, the important difference between the short-term and medium-term automatic protocols is that there is no delivery planning required by the supplier's system at this latter level.

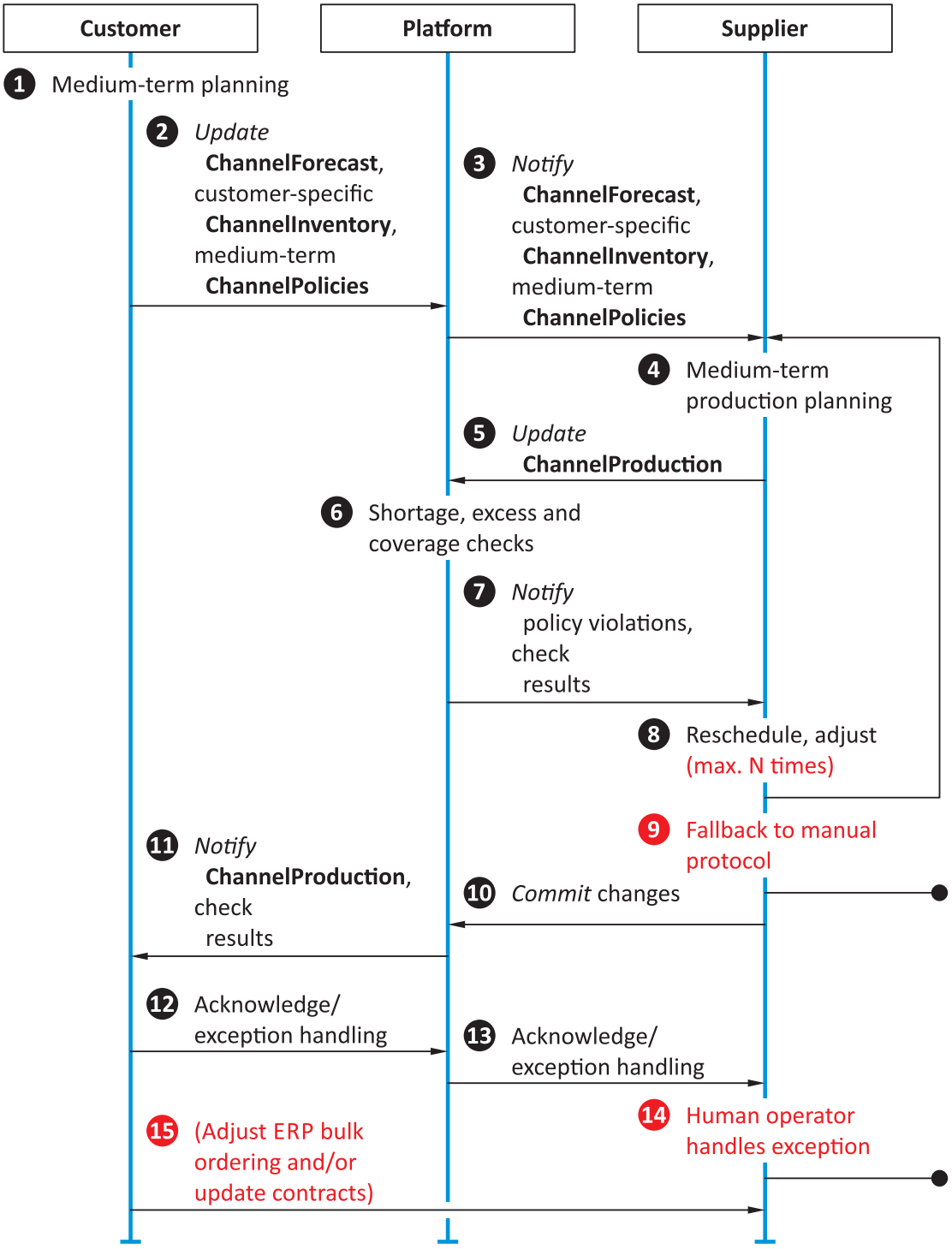


Figure 8: Protocol for exchanging medium-term information regarding an automatically managed channel.

2.4.3 Tools for filtering channels

If the number of material components between a customer and supplier exceeds a certain number, managing the associated channels via the man-

ual protocols described in section 2.4.2 can become tiresome for a human operator. When the operations quality improves over time, thanks to the platform, most channels evaluated during the information exchange protocol should show no problems, therefore, the human operators can concentrate on the problematic channels first.

Since each channel does have a dozen direct (e.g., names) and possibly another dozen aggregate or dynamic properties (status, quantities at a given time), hard-coding filtering options for such channels is generally infeasible. In addition, general experience shows, the users like to set up their own critera for filtering for components they have to work on.

Therefore, the platform features filter model that allows the users (both on customer and supplier side) to assemble their own filtering rules based on a rich set of choices regarding the aforementioned direct, aggregate and/or dynamic properties of a channel. The data model for storing such filters are shown in Figure 9.

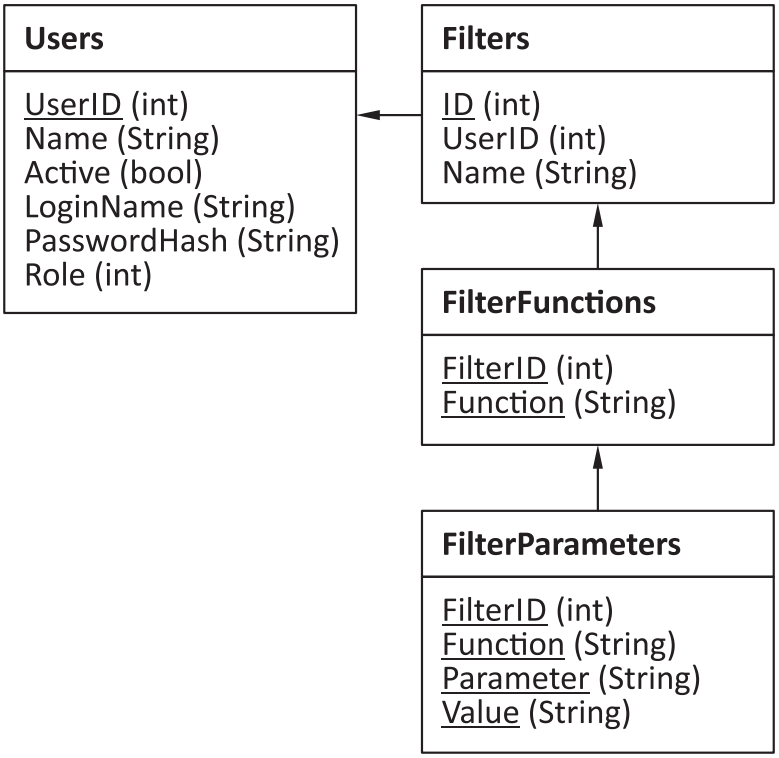


Figure 9: Data model for storing user specified and generic channel filtering definition.

Each user has its own personal set of filters, which consist of a set of filter functions, which in turn can have multiple parameters and finally multiple values for each parameter.

Filters The `Filter` table holds the per user (`UserID`) container for the specific filter definitions. The users can name their filters (`Name`) for easier selection.

FilterFunctions Each filter consists of one or more filter functions (`Function`). The set of available generic functions are defined by the platform based on the specific requirements of the logistics environment it is deployed for. Unfortunately, adding more functions requires software development on the platform itself. This limitation is a trade-off that enables having a concise user interface (see Figure 14 for example) for assembling the predefined set of filters. The alternative, as found in other systems requiring data filtering, is usually a more complicated mini programming language (i.e., regular expressions or any other Domain Specific Language, DSL) the users have to learn. Although more powerful, such feature would raise the bar for using the platform by logistics people who are generally not programmers by trait. (On a side note, such DSLs can likely be transformed into a representation that can be stored/applied the same way as the model shown in Figure 9.)

By default, when a filter contains multiple functions, those functions are considered in a logical AND conjunction. The structure supports composite filters via dedicated functions that take filter identifiers as parameters themselves. This allows adding other logical operators, such as OR and NOT to a filter by requiring the user to first assemble the secondary filters and then reference them from the main filter via these compositor functions. Table 1 lists the typical functions and their parameter types supported by the platform.

Function	**Parameter(s)**
Channel_Name_Equals	Set of Strings
Channel_Name_Not_Equals	Set of Strings
Channel_Name_Contains	Set of Strings
Channel_Name_Doesnt_Contain	Set of Strings
Customers	Set of customer ids
Customer_Name_Equals	Set of Strings
Customer_Name_Not_Equals	Set of Strings
Customer_Name_Contains	Set of Strings
Customer_Name_Doesnt_Contain	Set of Strings
Suppliers	Set of supplier ids

Supplier_Name_Equals	Set of Strings
Supplier_Name_Not_Equals	Set of Strings
Supplier_Name_Contains	Set of Strings
Supplier_Name_Doesnt_Contain	Set of Strings
Channel_Active	N/A
Channel_Inactive	N/A
Channel_Any_Problems	N/A
Channel_No_Problems	N/A
Channel_Demand_Present	N/A
Channel_No_Demand	N/A
Channel_Shortage_Within_Days	Number of days
Channel_Excess_Within_Days	Number of days
Channel_Inventory_AtLeast	Quantity
Channel_Inventory_AtMost	Quantity
Channel_Policy_Min_Setting	Policy types
Channel_Policy_Min_Coverage_AtLeast	Quantity
Channel_Policy_Min_Coverage_AtMost	Quantity
Channel_Policy_Min_Past_Days_AtMost	Number of days
Channel_Policy_Min_Past_Days_AtLeast	Quantity
Channel_Policy_Min_Next_Days_AtMost	Number of days
Channel_Policy_Min_Next_Days_AtLeast	Quantity
Channel_Policy_Max_Setting	Policy types
Channel_Policy_Max_Coverage_AtLeast	Quantity
Channel_Policy_Max_Coverage_AtMost	Quantity
Channel_Policy_Max_Past_Days_AtMost	Number of days
Channel_Policy_Max_Past_Days_AtLeast	Quantity
Channel_Policy_Max_Next_Days_AtMost	Number of days
Channel_Policy_Max_Next_Days_AtLeast	Quantity
Channel_Any_Supplier_Production	N/A
Channel_Without_Supplier_Production	N/A
Filters_AND	Filter ids
Filters_OR	Filter ids
Filter_NOT	Another Filter

Table 1: Typical filter functions of the platform.

When creating the filter functions themselves, the filter name usually

consists of the property followed by the operation to be performed on that property extracted from a channel's data model components.

FilterParameters The `FilterParameters` table is a multi-value table where each entry is an unique parameter value of a function of a filter since in the minimal data model of the channels, duplicate functions and parameter values have no meaning.

2.4.4 Evaluating past customer and supplier performance

Evaluating the performance of the participants, and transitively, the platform itself is an important feedback-loop mechanism for improving the overall efficiency of the logistics operation. Since the platform is committed to transparency, both the customer's and supplier's performance of each individual channels is tracked by the platform and is visible to both sides.

On the customer's side, the platform measures the precision of the component forecast which is the deviation between the planned (forecasted) and actual component consumption by the customer. Precision can be measured per time unit (e.g., day, week), also in an aggregated form for several time units. The idea behind this latter measure is that moving demand from one time bucket to a neighboring one does not really matter. The real source of instability is when there is a considerable difference between the forecasted and actual total demand on a longer period. On the other hand, the deviation of between different plans which is the cause of the well-known system nervousness is the measure of the planning stability. The data can be visualized also in a tabular format (Figure 10). Data in the table provide the basis for measuring planning precision and stability.

Evaluating suppliers' performance is done by comparing the scheduled consumption of the material component of the channel with the accessible inventory of each day of the past short-term planning cycles (i.e., the daily demand in the `ChannelDemand` table compared agains the daily inventory in the `ChannelInventory` table. The comparison may consider the in-transit inventory as well as any consignment inventory (inventory at the customer site that is managed by the supplier). Since the calculation considers the inventory level at each day, the results cannot be manipulated by the supplier by issuing component production

for the current day as any such extra quantity would appear only in the next day's accessible inventory (due to production and deliver lead times being non-zero).

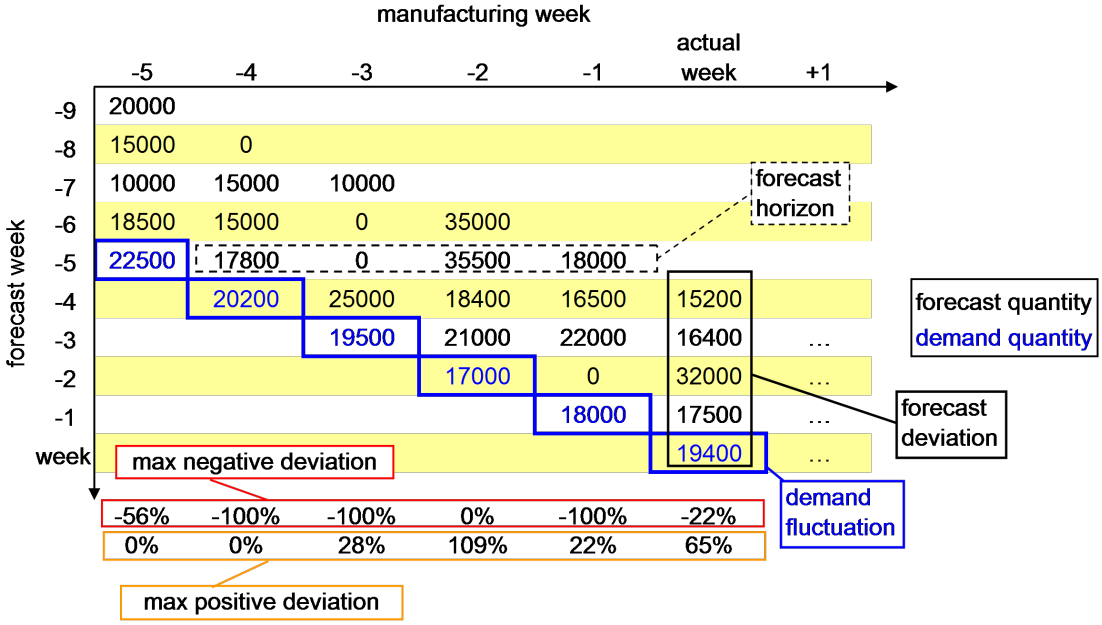


Figure 10: Example of forecast deviation and demand fluctuation in customer provided data of a channel.

The platform can present the performance evaluation results in diagram format as depicted in Figure 11.

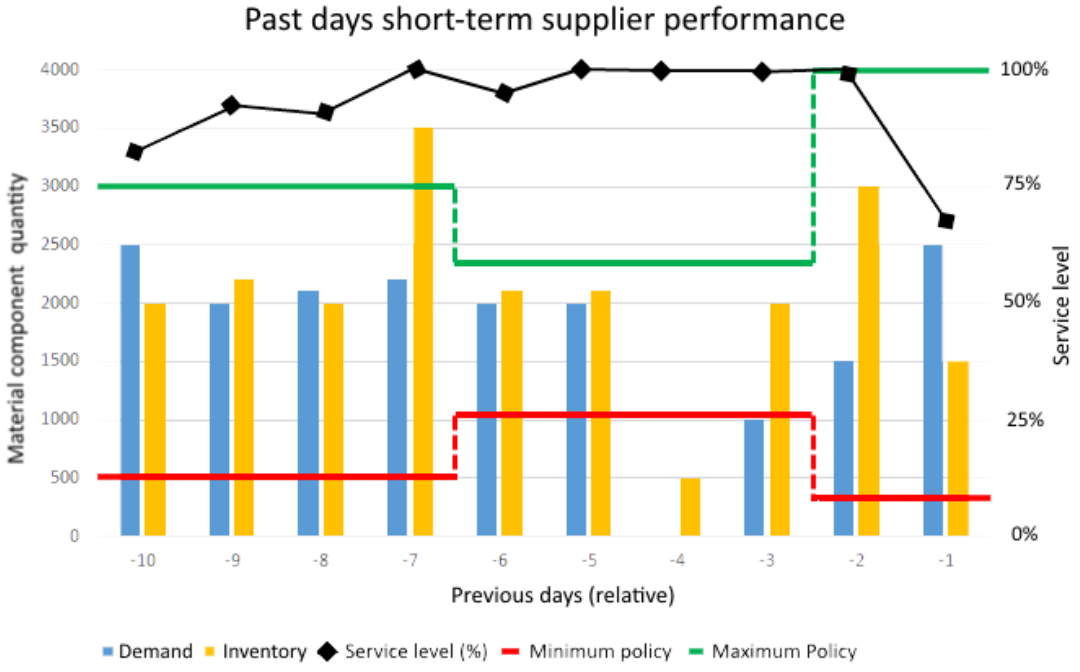


Figure 11: Example short-term performance of the supplier by matching customer demand and policies

The diagram demonstrates the cases where the supplier overfilled the inventory (day -7) and underfilled the inventory (day -3) according to thos days' bounds specified by the consumer. The service level is ratio between the available supply of the material component and the demand for it on a particular day in this example (capped at 100%). Such diagrams can be generated and aggregated for multiple channels (based on filters).

2.5 Application of the results

The version developed and deployed, called **Logistics Platform (LP)**, follows the focal structure of the supply network where it was applied. The LP is a standard Java Enterprise Edition (EE) web application, and as such, it can run on a traditional server or in the Cloud. The application can be accessed from the customer's intranet as well as from external suppliers through the Virtual Private Network (VPN) of the customer.

Each user has an associated list of channels which he/she can see and modify. This allows the privacy to be retained between different suppliers. On a channel, every assigned user can read the same data, but users at the sides of the customer and supplier have different modification rights. For example, the supplier's user can modify the delivery schedule for the component while the customer's user cannot, and the inventory checking rules can be modified by the customer user only (see Figure 12 and 13).

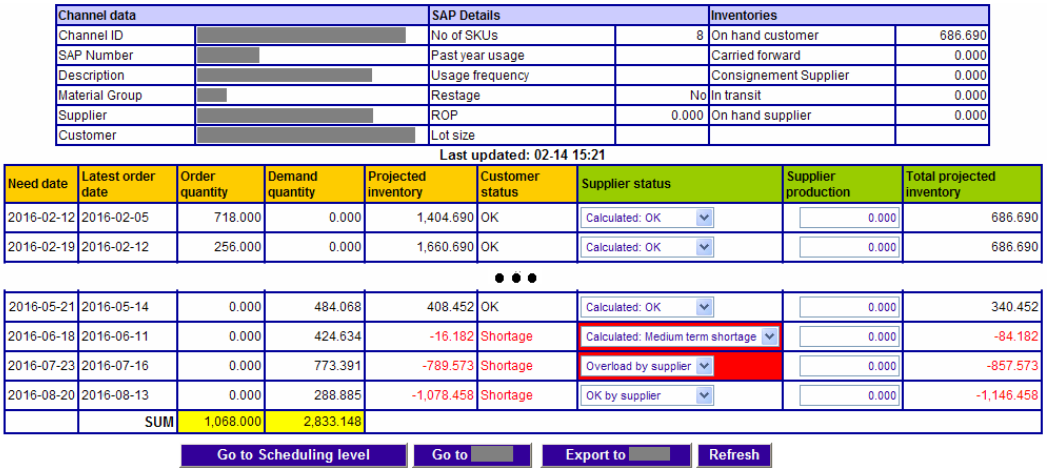
Channel data		Inventory policy		Inventories (2016-02-29 05:50:28)	
Channel id		Min. inventory policy	Fixed quantity	On hand customer	3 823,9
number		Min. quantity	1 260,0	Carried forward	-9,1
Description		Min. past days		Consignement Supplier	2 143,0
Material group		Min. next days		In transit	0,0
Supplier		Max. inventory policy	Fixed quantity	On hand supplier	2 220,0
Customer		Max. quantity	2 800,0		
Supplier material code		Max. past days		Unit	%S
Delivery calculation	Automatic Supplier; Deliver to order	Max. next days		Lot size	45,0

Last updated: 2016-02-29 07:08:01

Customer			Supplier	Inventory						Supplier		
Date	Scheduled demand	Open orders	Delivery schedule	Projected opening	Projected closing	Daily shortage	Delivery-demand balance	Coverage status	Order-demand balance	Delivery-order balance	Planned production	Projected closing inventory
2016-02-29	713,9	0	0,0	3 814,8	3 100,9	0	3 100,9	Overstock	3 100,9	0,0	0,0	4 363,0
2016-03-02	0,0	0	675,0	3 100,9	3 100,9	0	3 100,9	Overstock	3 100,9	0,0	0,0	3 688,0
...												
2016-03-05	670,8	0	0,0	2 603,8	1 933,0	0	1 933,0	OK	1 933,0	0,0	0,0	3 688,0
2016-03-06	712,4	0	0,0	1 933,0	1 220,6	0	1 220,6	Undestock	1 220,6	0,0	0,0	3 688,0
...												
2016-03-12	135,6	0	0,0	0,0	-135,6	135,6	-477,4	Undestock	-477,4	0,0	0,0	3 688,0
	4 967,2	675,0	675,0			477,4						

Go To Planning Level | Insert delivery item | Save and recalculate | Schedule and save | Export To | Refresh

Figure 12: Detailed scheduling level data of a channel.



Channel data		SAP Details		Inventories	
Channel ID	[redacted]	No of SKUs	8	On hand customer	686.690
SAP Number	[redacted]	Past year usage		Carried forward	0.000
Description	[redacted]	Usage frequency		Consignement Supplier	0.000
Material Group	[redacted]	Restage	No	In transit	0.000
Supplier	[redacted]	ROP	0.000	On hand supplier	0.000
Customer	[redacted]	Lot size			

Last updated: 02-14 15:21

Need date	Latest order date	Order quantity	Demand quantity	Projected inventory	Customer status	Supplier status	Supplier production	Total projected inventory
2016-02-12	2016-02-05	718.000	0.000	1,404.690	OK	Calculated: OK	0.000	686.690
2016-02-19	2016-02-12	256.000	0.000	1,660.690	OK	Calculated: OK	0.000	686.690
• • •								
2016-05-21	2016-05-14	0.000	484.068	408.452	OK	Calculated: OK	0.000	340.452
2016-06-18	2016-06-11	0.000	424.634	-16.182	Shortage	Calculated: Medium term shortage	0.000	-84.182
2016-07-23	2016-07-16	0.000	773.391	-789.573	Shortage	Overload by supplier	0.000	-857.573
2016-08-20	2016-08-13	0.000	288.885	-1,078.458	Shortage	OK by supplier	0.000	-1,146.458
	SUM	1,068.000	2,833.148					

Go to Scheduling level | Go to [redacted] | Export to [redacted] | Refresh

Figure 13: Detailed planning level data of a channel.

The web application collects data from legacy systems either via direct database access to the customer's scheduling and planning systems, or Extensible Markup Language-based (XML) data exchange with the suppliers' and customer's ERP systems. The XML-based data exchange with the suppliers can be automatic by using secure Simple Object Access Protocol (SOAP) services built into the web application or direct XML file upload, in which case the logged-in user's account is used for the data validation context.

Due to the minimal nature of the underlying data model, driving the application with real-time production data coming from a short-term scheduling system [5] or from a (what-if) simulation [2] of the manufacturing process is equally possible as the model does not assume the source of the customer's data. In practice, if the same application is used for evaluating real and simulated data, the user interface distinguishes the two with different coloring and/or using warning labels.

The data and information acquisition process works in three different ways: **periodically scheduled**, **event-based** and **ad hoc**. A job is running every morning right after the customer's scheduling system has generated its new production schedule (and, subsequently, its scheduled component demand). This job collects also the actual inventory data as well as the material planning and forecast data (from the customer's planning system). During the day, the customer's operators can change the initial schedule by hand and this change is propagated automatically to the LP. In the general case, the web application's administrator can trigger any time a complete re-synchronization of the LP with the related systems.

The web application's report and input screens are designed for maximum data- and access-security by utilizing

1. user roles, page- and object-level access checks,
2. client- (JavaScript) and server-side form validation and data integrity checks and
3. anti-SQL injection and anti-Cross Site Scripting (XSS) techniques.

Beyond guaranteeing security, another primary design goal was to make the access of the vast amount of data behind the LP fast and filterable. The speed requirement was achieved by using in-memory object caching technology for critical data such as actual inventory levels. *Filtering* is a key feature in the application, because each user can have hundreds of assigned channels, but space and time restrictions allow them to operate only on a small subset at a time. Therefore, each user can define his/her own set of filters which he/she can use later on in any situation (Figure 14). The filters which are logical constructs of property value set pairs belong to the personal profile of the users. Filters are used also for collecting basic and generating aggregate values for a set of channels, such as for evaluating the overall performance of a supplier who is responsible for a number of channels.

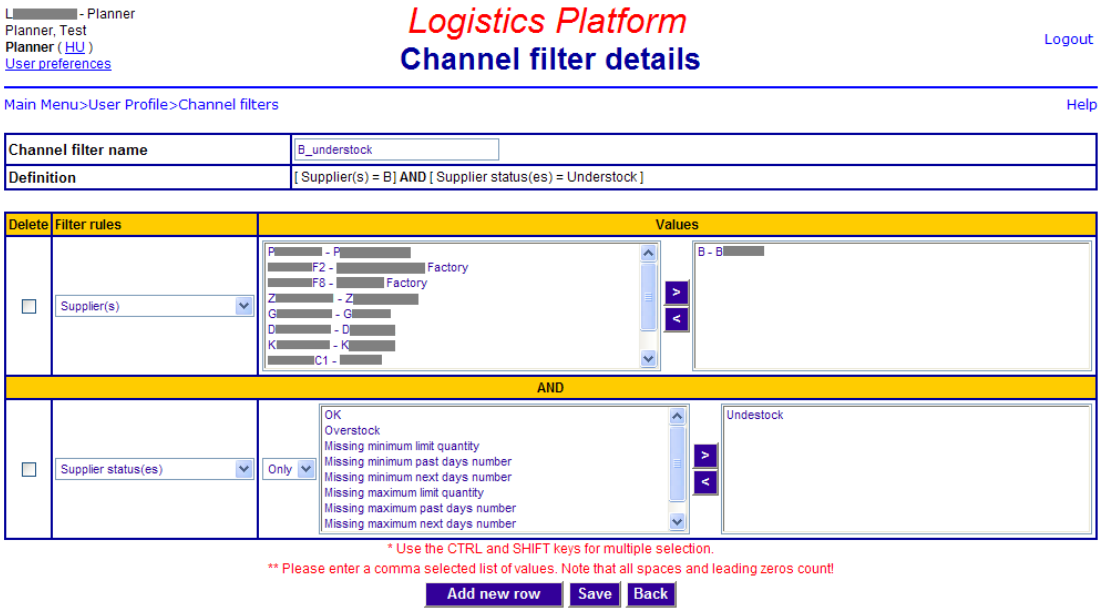


Figure 14: Assembling channel filters in LP

2.5.1 Planning on the customer side

Static, master data about the channels are fed into the LP from the transactional ERP system of the consumer (focal manufacturer). As for the dynamic data, the customer periodically runs a master planner that determines the output of end products for a longer horizon. Dependent component demand forecasts are generated from this master plan by standard material requirements planning (MRP) methods [49]. Demand for the same components are summed up and feed into the LP as medium-term demand forecasts. Purchase order channels are also filled in with orders generated by the supply planners of the customer.

In parallel to developing the LP, an automatic production scheduler was developed and deployed at the production facilities of the customer [37, 74]. This scheduler assigns in time each task to appropriate production resources so that it guarantees the satisfaction of hard technological, temporal and resource capacity constraints, while approaches optimisation objectives such as maximal resource utilisation and minimal backlog. The scheduler concerns also the availability of material: in a predetermined time window it takes material availability as hard constraint and postpones tasks that have no guaranteed supply. Component supply is sufficient if the total of inventories together with the scheduled delivery matches the actual demand on each day of this time window. Short-term scheduled demand for components is derived from this detailed production schedule. For example, consider the state of a channel in Figure 12: in the next few days no daily shortage is projected, hence the production schedule is feasible, even thought with a longer, 10 days look-ahead material shortage can already be expected given the actual inventories, scheduled delivery and demand.

2.5.2 Planning on the supplier side

For supporting the cost-oriented decisions of the supplier on the planning level, a portfolio of novel coordinated supply planning methods have been developed that take into account all the logistics costs and calculate also with the uncertainty of demand that may stop whenever market demand for the end-product(s) built of the component ceases for any reason. In this situation called run-out the component inventory becomes obsolete. The methods decide upon the time periods and quantities of component production. The total cost to be minimized includes the setup, in-

ventory holding and expected obsolete inventory costs. Hence, decisions that coordinate a specific channel can be made on the basis of information coming partly from the customer (demand and its uncertainty) and partly from the supplier (setup, production and inventory holding costs). Various methods have been developed for different situations: typically, when plans should be made for the whole horizon [42], or when it is enough to plan for the close future [95].

The coordinated supply planning methods have been extensively tested on industrial datasets and surpassed the performance of the prior methods [40], although, it is the individual supplier's decision whether he/she applies them or stick to their existing practice. This is the case also with generating delivery schedules: while some suppliers rely on the built-in methods of the system that produce delivery schedule as a response to scheduled demand automatically, some apply more advanced distribution and transportation planning methods and optimize not only for minimal material shortage but for minimal transportation costs, too.

2.5.3 Experiences and extensions

The logistics platform has been deployed at the focal manufacturer of a production network producing customized mass products and was used on a daily basis both by the planners of this manufacturer (in the customer's role) and of its strategic suppliers. Channels for almost 10,000 components (including packaging materials) have been set up. Since the business processes at the time were based on orders, only Purchase Order type channels could be defined.

The first main experience was that thanks to this application, the overall supply planning process became much more transparent. The misfit of the aggregate planned demand and detailed schedules, which is rather the rule than the exception in practical applications [26], became evident at first sight. In contrast to the actual practice, the LP made explicit a number of conflict situations. For instance, purchase orders for components do not cover in every case the actual scheduled demand. However, it did not take much to the planners to see this positively: while having the LP do the routine tasks (including the execution of various checks) they could focus on the really challenging tasks whose resolution needs human intervention, negotiation and creativity.

Just before the deployment of the LP, the automatic production sched-

uler [37, 74] was set into daily operation at the focal manufacturer. Given cc. 100 production lines, alternative bills of materials (BOMs) and alternative routings and some thousands tasks to schedule at a time, the scheduler strived to minimize the delays and maximize resource utilization. However, after making material availability checks using the on-hand inventory at the customer, the production schedule had to be modified extensively because of short-term material shortage. Now the scheduler and the LP are synchronized and the material check considers also the delivery schedules of suppliers. Provided suppliers are really able to keep their promises, this integration leads to far less re-scheduling and a more stable operation of the overall network.

The routine use of the system made also clear that the network as a whole needs novel business models and processes. An improvement easy to accomplish could have been the application of Coordinated channels where orders will be generated automatically whenever demand and supply matches on the short term. However, in reality the assumption of truthfulness that underlies the Logistics Platform (and most systems in production informatics) is untenable. Planning functions, even within the walls of the same enterprise, have distinct, occasionally conflicting objectives. To remain on the safe side, they have incentives to distort the information they share with others. Note that this does not imply deliberate change of some data; it is just enough to tweak some control parameters of a complex planning procedure.

All in all, in order to align potentially conflicting incentives, a further step has to be taken towards cooperation. Hence, based on recent experiences, the design of new mechanisms has been started that drive the partners towards disclosing and using unbiased information when trying to coordinate a channel. Accordingly, the supplier provides a service to the customer by committing itself to meet all short-term demand. In return, the customer pays for (1) the components delivered, (2) the flexibility of the supplier, and (3) the forecast deviation [96]. The performance measurements and the so-called coordinated channels will have a central role in these developments when the LP, that controls now only the flow of information and components, will be augmented with the flow of financial assets. The basis of financial calculations will be a fair share of the costs and profits of operating on a risky market.

Another option for cooperation is to tackle issues of product design, production and logistics on the network level and respond to dynamic

market demand with a product mix that can be built up from modular, easy-to-configure elements. According to the initial experiences, proper product design would significantly decrease the complexity of planning processes and the total logistics costs.

The main benefit of using the application is that the supply and demand planners can see, compare and analyze dynamic information coming from different and heterogeneous sources. This way, they can anticipate future conflict situations. The final goal was to make efficient local planning and reliable real-time plan execution each partner's primary interest. This is the key for inspiring cooperation in a production network whose overall performance depends on the decisions of its autonomous members.

The introduction of the Logistics Platform has resulted in an increased service level[14] over a broad range of material components between the focal manufacturing plant and its suppliers. Figure 15 shows a typical component's service level before and after the introduction of the platform:

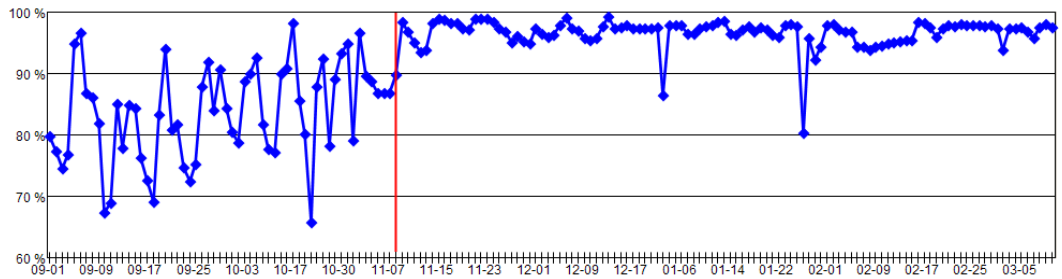


Figure 15: Service level of a component before and after the Logistics Platform went live (indicated by the red time marker).

2.6 Summary

Thesis 1: Coordinating the material component demand and supply in a single-(focal)-consumer and multiple-supplier logistics network requires the consumer to reveal its detailed production schedule and material component forecast to the respective supplier in exchange for the detailed production capacity and delivery schedule of said supplier. In this context, the aim is to ensure the consumer has little to no (unforeseen) shortage of its input material components and

[14]Service level here is defined as the ratio between the successful consumption of the component on a particular day in respect to the total planned demand for said component.

when and how many of the material components the respective suppliers are going to deliver while keeping the information private from the other suppliers. The information exchange should be kept up-to-date by the parties involved via a shared platform that provides a novel and minimal interaction model by defining:

1. **the interaction unit as a channel (containing information about the material component, the supplier, lead times, shortage policies);**
2. **an association between users of the platform and the individual channels while ensuring users from different suppliers cannot see or modify each other's data;**
3. **access to the dynamic detailed production(s), forecast, inventory levels and delivery schedule; and**
4. **the means for filtering for problematic channels that need manual intervention.**

The shared platform should also ensure that, given two levels of planning horizon (medium, short) and two levels of automation (semi-automatic, manual), the participating users follow a novel protocol for validating and confirming suggested plans for each channel or resolving conflicts by adjusting their plans and, consequently, the data provided to the platform for re-evaluation. The platform should provide the same information to both ends of the channel's participating users about the channel's up-to-date status but different modification rights (such as a demand-fulfilling policy of the consumer versus detailed delivery schedule of the supplier).

Related primary publications: [4], [1], [2],

Related additional publications: [9], [11], [12], [7], [14], [5], [13], [6], [17], [18], [20]

3 Data transparency in supply logistics

3.1 Modeling data types to enable structural reasoning and matching heterogeneous data sources

3.1.1 Introduction & problem statement

Networks of enterprises often experience limits in unfolding their potential due to their organizational heterogeneity and the resulting lack of process transparency. Multiple-participant logistics networks are no exception in this regard, with the following issues being perceived as the most notable obstacles to improvement:

- Although network-wide standards are likely to be implemented to support interoperability, differences may exist in their local interpretation as well as de-facto compliance with the requirements. Lack of discipline or misinterpretation are not always to blame for these variations – in many cases, company-internal data handling and reporting practices (including varying connectivity, e.g., on the shop-floor level) cannot be perfectly mapped onto the common specifications.

- Concerns about potential risks of data sharing can considerably limit what is being shared within the network. In some cases, business models for collaboration are already set up with transparency limits to make participation more attractive for newcomers. Another frequent pattern of such limits is their apparent simplicity – often, they are kept simple enough to maintain a favorable first impression but rarely address more complex risks as massive collection (i.e., "mining") of fragmented information, or inference attacks [32].

- Even if data are present at some point in the network, their place and time of availability, their granularity or level of abstraction may render them inadequate for establishing transparency. In logistics networks, for example, this may result in shipping information lagging behind an actual material stream, impairing operational decisions.

Some of these obstacles can be addressed very well without changing companies' attitudes or business models, merely by exploiting opportunities that already exist in the network, especially in the form of readily accessible but poorly structured data.

In networks, where the number of participants is numerous, the number of interoperation services are large, and partners may come and go frequently, such as in tracking-and-tracing scenarios [1], coordinating the design of the required data interoperation at the IT and software levels becomes difficult. Often, integration services reuse existing internal data sources which do not match the required data structure exactly and is expected from the rest of the network to adapt their consumers of data accordingly.

If all participants want to operate along this principle, the number of custom datatypes grow exponentially. Even if the transport method and data representation is the usual industry-standard (and decades old) Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP) message based exchanges, these require exact matching of components on the consuming end of a data exchange. In the aforementioned scenario, where more data could be provided, the structural mismatch of the data in the protocol results in a non-operational service.

In practice, usually one side of the information interchange to be established is already given: an existing service providing data or an existing consumer taking the data, needing the developer to implement the other side. Since the most widely used representation is still XML, its structural descriptor – XML Schema Definition (XSD) – is available or can be generated. If both the provider and consumer have such XSD for a particular service, matching the two by hand is tedious and error-prone.

Tools helping in this regard are productivity boosts, and due to service evolution, is a reoccurring need for developers, especially in fields such as mobile application development where time-to-market is a matter of a life-or-death situation on a daily basis.

3.1.2 Literature review

Describing data via XML, introduced by the World Wide Web Consortium (W3C) [102], is a widely used approach that helps to encode documents in a format readable by both humans and machines. Transmitting data on an XML basis between components of a system, between systems and between companies is a common operation and requirement in the modern computer and business-to-business era.

Processing XML-based data is very important in these environments, therefore, several functional languages and frameworks have been de-

veloped over the years to handle the task. Most known tools used with XML are the query languages XQuery [104] and XPath [99], as well as Extensible Stylesheet Transformation (XSLT) [100] describing transformations of XML documents. Unfortunately, XML by itself and the mentioned associated technologies do not convey or handle the semantics of the object the XML-based data represents and describes.

Beyond the well-formedness requirement of XML documents, several means of structural, semantic and content-wise definitions can be attached to the documents. Such a definition may come embedded in an XML document in the form of Document Type Definition (DTD) [98]. The drawback of DTD is that it does not specify the individual element or attribute types; every content is considered a string. A more expressive definition was created by the W3C in the form of XML Schemas (XSD) [103]. Apart from a few extreme corner cases, XSDs let the developer define the fine structure and data types required to be present or absent in an XML document. RELAX NG [77] is a similar schema language for the purpose of defining and verifying XML documents.

A common property of these processing languages and frameworks is that they do not make use of any of the available structural type information associated with the document (DTD or XSD) being processed, and they can fail or provide no results at runtime if a non-conform document structure is processed. To overcome this limitation, several extended query languages and libraries have been created. One notable example is the XDuce framework [51] which provides a statically typed programming language for XML processing supporting many functional constructs such as pattern matching and dynamic type checking. The theoretical foundation of the framework is built upon regular tree automata, and the formal definition and the proof of type safety are presented.

Another example is the XMλ framework [69] which uses a statically typed language with type inference. The language is polymorphic, higher-order, and supports pattern matching as well. One shortcoming of languages such as XDuce and XMλ is that they seem to work on DTD typed XML documents only and ignore the attributes.

XML type systems are, unfortunately, lacking the type polymorphism commonly found in many modern programming languages. The need, for example, to have the well-known Simple Object Access Protocol (SOAP, [101]) parameterized over its content document by another schema has led to several academic extension attempts to the XML

schema ecosystem. One example, based on schema annotations, can be found in [50].

XML processing languages may benefit from schema reduction and canonization. Creating an XML Normal Form similar to relational database normal forms is one possibility [23]. Alternatively, one can define a common structure for representing schema definition elements [38].

Transmitting XML data between companies poses an important challenge. Similarly typed documents might represent the same data on both sides of an XML exchange. To match parts of the schemas, similarity measures have been proposed, such as [54] which uses supervised learning. Another example is the Cupid generic schema matching tool [64], which employs linguistics-based matching, element- and structure-based matching and key and reference constraints.

If XML is chosen as a basis for a type system with subtyping, methods are required which can tell the **relation between two schemas**. Unfortunately, common tools and methods such as Cupid or the XML reduction algorithm mentioned above do not provide this information.

3.1.3 Solution approach

Definitions **Relation** defines four states between two objects:

- NONE: no correlation,
- EQUAL: they represent exactly the same,
- EXTENDS: given two objects A and B, if A can be used in every place where B is expected since A offers all capabilities that B holds, then the relation is covariant, and
- SUPER: the opposite to EXTENDS, i. e., where A is expected and B may be used since B offers all capabilities that A holds, the relation is contravariant.

Cardinalities represent how many times an element or an attribute may appear:

- ZERO: the element or attribute should not appear at all,
- ZERO_OR_ONE: the element or attribute may optionally appear once,

- ZERO_OR_MANY: the element may appear many times or not at all,
- ONE: the element or attribute should appear exactly once, and
- ONE_OR_MANY: the element should appear at least once.

The relational ordering is the same as the listing above. The covariance expectation is that algorithms expecting ZERO_OR_ONE can deal with an element instance which has ONE_OR_MANY. While elements can have all 5 cardinality categories, attributes may use only ZERO, ZERO_OR_ONE or ONE, respectively.

Name is a composite object representing the optionally fully qualified name of an element or attribute. Since programmatic access of elements and attributes in DOM (Document Object Model) differ, the name object also contains the fact whether it represents an attribute or not (XSD allows the definition of child elements and attributes with the same name, which may lead to confusion). Comparing an attribute with an element always yields NONE. In addition, name aliases and an extra semantic dimension may be defined in the XSD by using the xsd:annotation/xsd:appInfo node (note that xsd is used shorthand for the XML Schema namespace URI http://www.w3.org/2001/XMLSchema). The semantic dimension may be application-specific and can represent an additional relation level beyond the name similarity.

Capability represents the unit of the type system. It is a composite object consisting of:

- A Name property,
- the Cardinality property,
- an optional simple value type, and
- an optional complex type referring to a Type (see below).

Type represents a collection of unique Capabilities where no duplication is allowed in terms of the capability Name properties, e. g., two distinct capability names should always compare to NONE in their name.

The Comparison algorithm The role of the comparison algorithm is to tell the relation between two canonized XMLSchema- based Types A and B (see also example in Figure 16). Figure 17 shows the entire comparison algorithm flowchart.

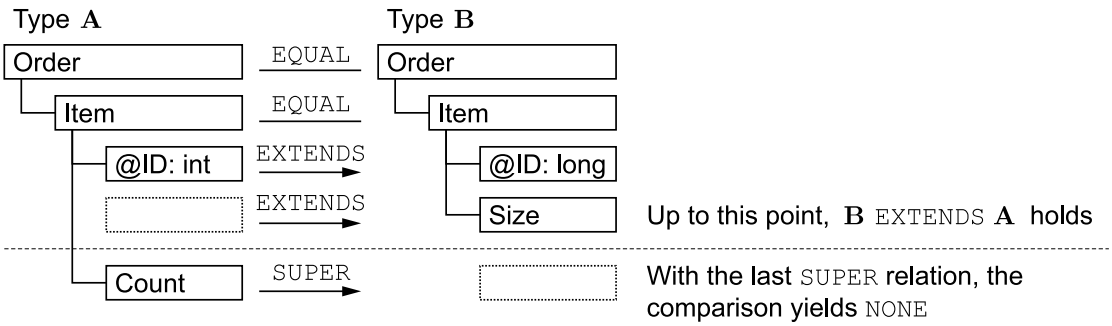


Figure 16: Example of comparing types A and B.

The *Type level comparison* algorithm consists of two nested loops for both types and pairwise compares the elements of both capability sets and counting how many times an EQUAL, EXTENDS or SUPER relation is encountered. The final verdict is met as follows:

1. If the number of common capabilities is less than either of the two capability counts, the relation is set as NONE. (This is the case when both seem to extend the same supertype.)
2. If all capabilities are EQUAL and no extra capabilities are present on either side, the relation is EQUAL.
3. If all capabilities are EQUAL and A has more capabilities than B, then the relation is EXTENDS. (In contrast, if B has more capabilities, it yields SUPER.)
4. If all capabilities are in an EQUAL or EXTENDS relation, then the relation is EXTENDS. (If they turn out to be EQUAL or SUPER, the result is SUPER.)
5. If all three relations appear, the relation is inconclusive, hence NONE.

Since the XSD allows the definition of recursive data types, the algorithm tracks the visited Type identities during the recursive pairwise comparisons. To determine the relation of such recursive types, the following heuristic approach is used. If a recursive type is identified on both capabilities, an index is constructed from the point where the types were seen in the call stacks of the recursive evaluations. If the indices match, the comparison yields EQUAL, otherwise, it will not contribute to the final evaluation (e. g., NONE relation).

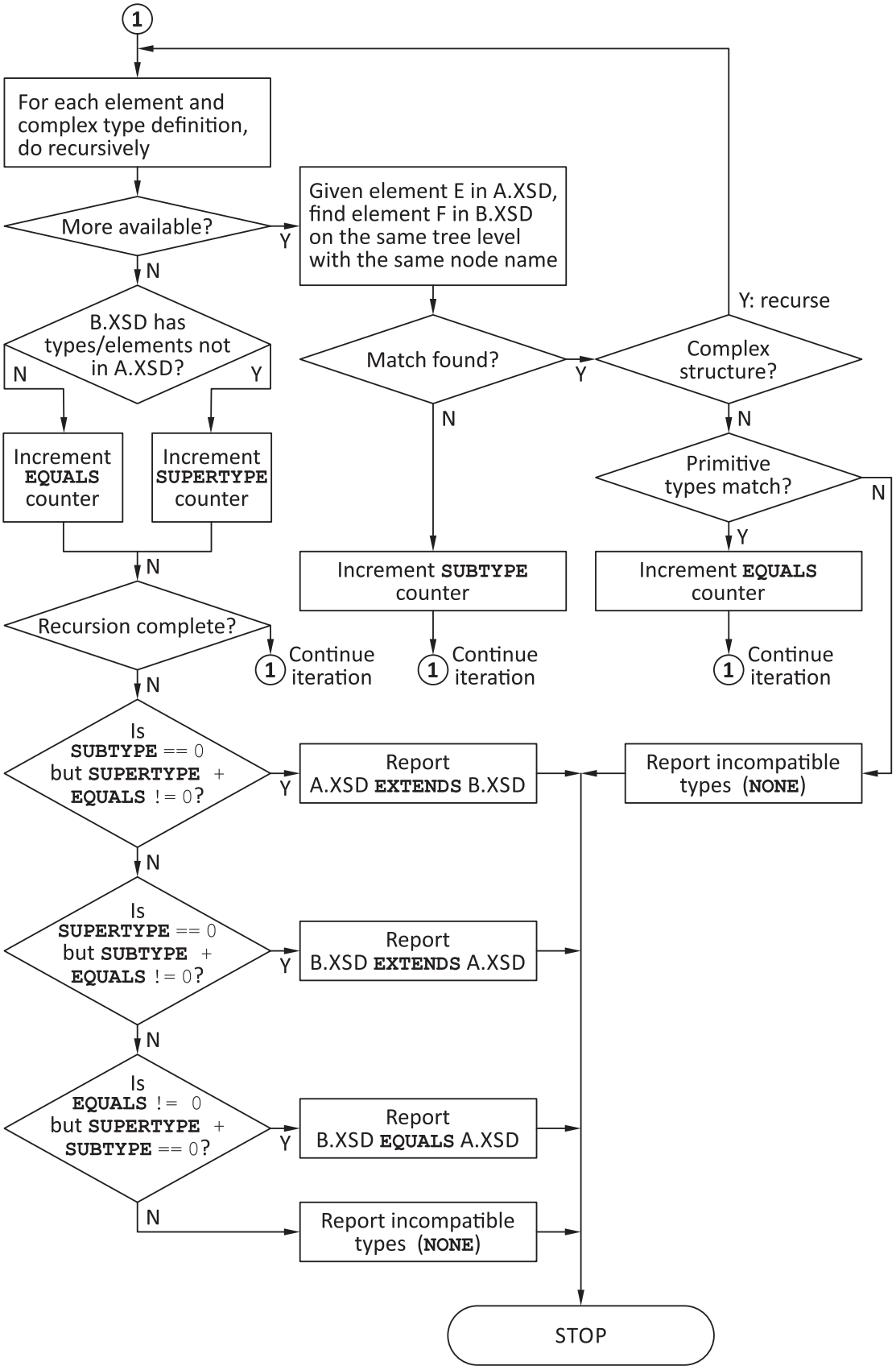


Figure 17: Flow chart of the type comparison algorithm

The *Capability level comparison* algorithm compares two concrete capabilities based on the following rules:

1. The Name property is compared. If the relation is NONE, the comparison terminates with NONE.
2. Both capabilities may refer to a simple or complex type. If this differs, the comparison yields NONE.
3. If both capabilities refer to two different simple value types, the comparison yields NONE.
4. If both capabilities refer to a complex type, the Type level comparison algorithm is called recursively.

The number of EQUAL, EXTENDS and SUPER properties are counted, and the final verdict is met similarly to the aforementioned type level comparison:

1. If all properties equal, the result is EQUAL.
2. If some properties EQUAL and the rest is EXTENDS, the result is EXTENDS. (Similarly, EQUAL + SUPER yields SUPER)
3. If the property comparison results in mixed relations, the end result will be NONE.

The Intersection algorithm The role of the intersection algorithm is to construct the common supertype of two XML types by using only the common capabilities of both in case their relation turns out to be NONE from the comparison algorithm (note that an EXTENDS or SUPER relation means that the intersection will be identical with one of the two types compared, as shown in Table 2.

Relation of A and B	**Intersection(A, B) identical with**
A EXTENDS B	B
A SUPER B	A
A EQUAL B	A (by convention)

Table 2: Cases where Intersection(A, B) delivers one of the original type arguments.

The algorithm performs the same pairwise comparison of capabilities and constructs a new type where only the smaller typed capabilities are kept (see also Figure 18). The common capability is constructed as follows:

1. If both original capability names compare to other than NONE, the smaller Name is used.
2. If one or both refer to a simple value type, the result will be an empty Type by convention.
3. If both are complex types but not recursive, the result will be the intersection of these types by recursively invoking the intersection algorithm itself.
4. The new cardinality will be the smaller one.

The resulting type structure C will result in a SUPER relation when compared to any of the original types: If C = Intersection(A, B), A EXTENDS C and B EXTENDS C are both true.

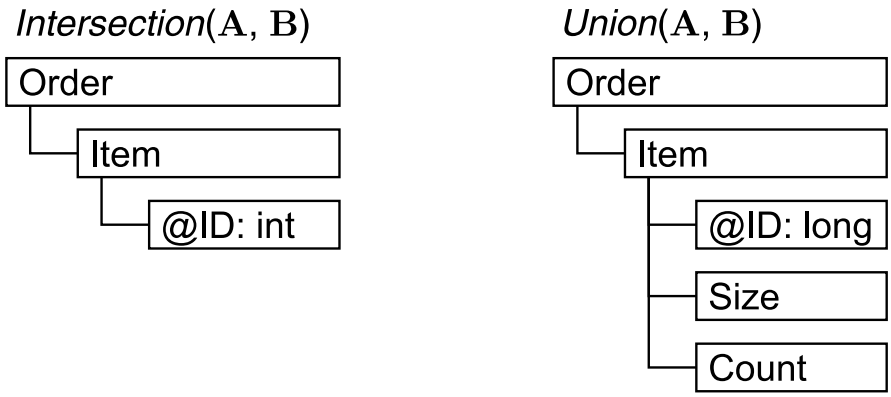


Figure 18: Intersection (left) and union (right) of types A and B from Figure 16.

The Union algorithm The role of the union algorithm is to construct a common subtype of two XML types where the best capabilities are joined together (see also Table 3 and Figure 18).

Relation of A and B	Union(A, B) identical with
A EXTENDS B	A
A SUPER B	B
A EQUAL B	A (by convention)

Table 3: Cases where Union(A, B) delivers one of the original type arguments.

The algorithm is similar to intersection, the comparison results are only considered the other way around:

1. If both original capability names compare to other than NONE, the better Name is used.
2. If both are complex types but not recursive, the result will be the union of these types by recursively invoking the union algorithm itself.
3. If one or both refer to an (unequal) simple value type, the union algorithm fails.

The new cardinality will be the broader one. The only exception in the algorithm exists when two unequal simple value types or one simple value type and a complex type meet. From a type inference perspective, the failure of constructing a union type may be the indication of a typing error or incompatible types.

Limitations Experience with real world XML schemas prompted the following simplifications and restrictions in the prototype implementation of the XML-Schema-based type system:

- Mixed content type is not supported.
- Only 5 simple value types are supported: boolean, integer, decimal, string and timestamp. Any derived or constrained primitive XML type is treated as these (e. g., NMTOKEN is string).
- Lists or enumerations of the 5 simple value types are treated as strings.
- Cardinality ranges other than those defined above are treated as ZERO_OR_MANY or ONE_OR_MANY, depending on whether or not the lower bound is zero.
- Nested xsd:choice and xsd:sequence are flattened and their cardinality values are merged into those of their children.

While type inference algorithm presented in subsection 3.2.3 works with parametric types, the current implementation of the XML type system does not use, enforce or handle any generic type information.

3.1.4 Application of the results

The main application of Theses 2, 3 and 4 was the **ADVANCE Flow Engine** where ADVANCE[15] is an acronym of the project in which it was piloted. In the course of the ADVANCE project, a pilot application was put into live use in a nationwide UK logistics network. The structure of the network consisted of a central hub facility and a set of selected local collection and delivery subcontractors, and has undergone incremental improvements based on findings of the introduction of new system components and functionalities. This so-called **hub-and-spoke** network organization exploits the efficiency improvement coming from the recognition that delivering parcels from various senders in the country to the same destination is more efficient when done by one contractor via dedicated transportation vehicles than each individual contractor trying to deliver to all the designated destinations by themselves. Figure 19 gives a brief overview of the flow of parcels and associated information in this type of logistics network.

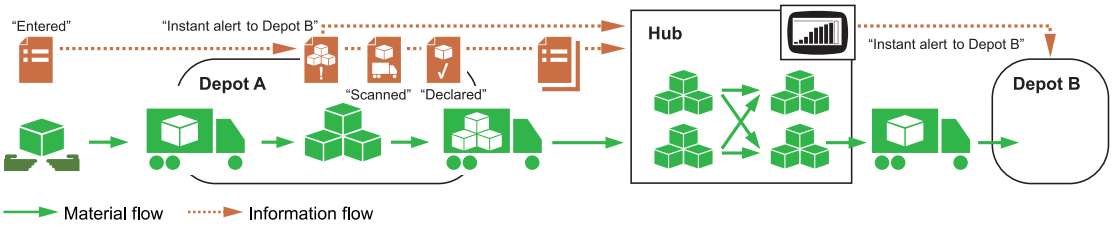


Figure 19: Typical material- and information-flow in a hub-and-spoke logistics network.

At each subcontractor, serving a postal region, parcels are collected and loaded onto lorries on a daily basis in a depot, then delivered to the central hub. There, the parcels are unloaded, then the lorry is loaded with different parcels destined to the postal region (depot) the lorry came from. The central hub's role is to unload, sort by destination and load lorries with the parcels. There were several issues related to this operation, especially on the information-sharing front:

- the set of subcontractors could change on a weekly basis, putting stress on information tracking,
- the IT systems of the subcontractors often did not speak the same vocabulary as the hub's IT system,

[15]EU FP7 under Grant No.257398, "ADVANCE – Advanced predictive-analysis-based decision-support engine for logistics".

- the digital status of the parcels was often not tracked real-time (due to the loose requirements of the legacy contract between the parties), was non-uniform among the various subcontractors and was often entered manually and
- the sheer amount of data generated by status changes, alerts and notifications on these parcels in the network was about to become untrackable by conventional means.

The project set out to provide solutions to these problems by introducing a new software system called ADVANCE Flow Engine that utilized the novel scientific results presented in the dissertation as well as providing a platform for machine learning, process optimization and real-time parcel tracking features developed in other work packages of the ADVANCE project.

As part of this ADVANCE Flow Engine, data types of inputs, outputs of the flows and specific processing blocks have to be defined. In practice, inside the flow editor of the Engine, this manifests itself in a short textual reference of `domain:typename` which is then backed by a `typename.xsd` file accessible to the Engine's type system manager. For example, the machine learning block `KMeansARXLearn1`[16] takes the input data as a tuple of (timestamp, group, value) and produces the learnt model in the form of a composite `arxmodel` consisting of the model and external coefficient numbers (Figure 20).

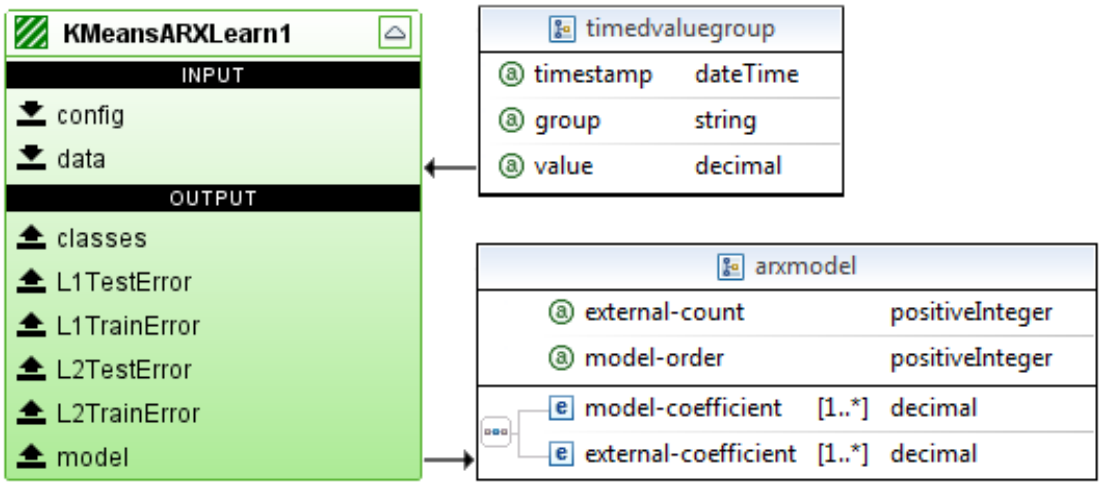


Figure 20: Example block with custom (and composite) input and output type structure.

[16] ARX: AutoRegressive with eXogenous inputs model where the model depends linearly on previous values, stochastic elements and external driving values.

While the strongly typed general purpose programming languages rely on the declared relations between data types when comparing them, the algorithms of Thesis 2 use structural information to determine the relation of two data types and/or generate an intersection (common parts) or union (combined parts without redundancy) as demonstrated on Figure 21:

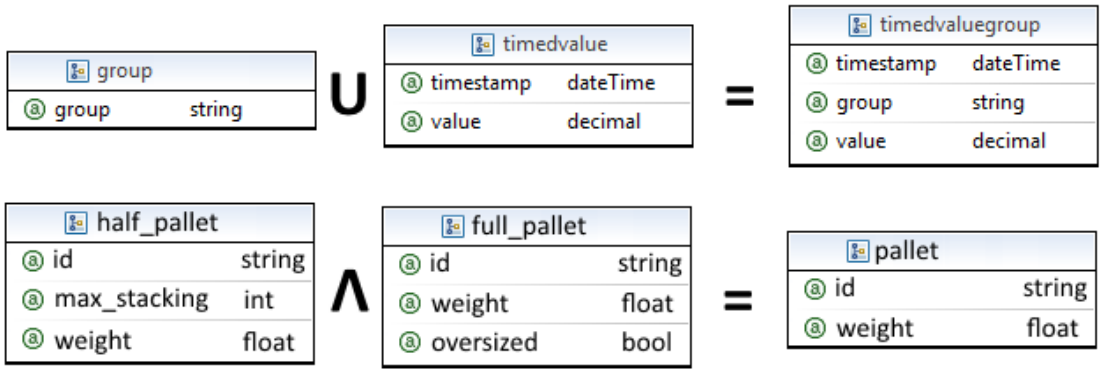


Figure 21: Generating union and intersection types from different data types.

If the individual data types to be combined are composite, the same union/intersection is performed on each level of the structure recursively.

Due to the scope of the ADVANCE project and complexity associated with ontologies in general, the concept matching of capabilities in the datatypes by the algorithms of Thesis 2 is limited to defining aliases (via annotations) in the XSDs of these datatypes and the option for the algorithms to call programmer-defined routines when performing the matching operation.

3.1.5 Summary

Thesis 2: Integrating raw or low-level data and events between the heterogeneous information sources and information consumers within a manufacturing company (focal consumer) and in connections with suppliers of material components should require the information systems involved to represent the data syntactically and semantically in a compatible, (semi-)automatic and computationally efficient fashion. This can be ensured, e.g., by using XML as the syntactic base as well as XML-Schema for defining the semantic baseline. Once the schemas of the sources and consumers are defined, they should be turned into a novel canonical representation that

1. **is organized as a directed, potentially cyclic graph where the nodes represent a leaf node with concrete data type or a named container of other nodes, both including the cardinalities of the contents; and**
2. **reduces the wide variety of data types of the underlying schemas to those typical in the manufacturing and logistics exchanges (number, text, timestamp).**

Given the canonical representation, three novel algorithms should be employed to

1. **determine the relation of two schemas (such as equivalence, incompatibility or if one encompasses the other);**
2. **generate the canonical representation of the common superset; and**
3. **generate the canonical representation of the common subset of the two input schemas**

by using coordinated, iterative descent on the graph, matching the nodes (by name or by semantical similarities through additional annotations), the nodes' parameters, and aggregating the primitive relations (equivalent, incompatible or one covers the other) into the overall relation result and/or building up the superset or subset canonical representation in the process. The output of the algorithms should be used for validating the interoperability level (or possibility) between source(s) and consumer(s) of specific data- or event-integration cases and should provide suggestions for better alignment by adjusting the appropriate data- or event-definitions.

Related primary publication: [3], [10], [15]

Related additional publications: [8], [16], [19]

3.2 Validating and inferring data types in information exchange networks

3.2.1 Introduction & problem statement

When working with data types coming from heterogeneous data providers such as the suppliers' own IT systems, requiring and receiving data in a certain minimal shape or structure is often not possible or

feasible at first. However, often existing data sources with broader information content are available to be collected, integrated and processed by a set of generic processing steps who only define a minimal structure based on their input needs. The relation between the minimal and practical structure is called covariance[17].

When using traditional web services such as SOAP, the data type definition of the particular data source is often misaligned with the data type definition of the input for the processing stage and either requires expensive/manual data transformation or rebuilding the logic behind the processing step to work with the data type that of provided by the source. However, if the data source represents some kind of event, the time nature of the data allows creating processing stages that do not really care about the data structure they have to work with but more like the way the data becomes available over time. These generic processing steps often use so-called parametric types in their inputs and outputs thus when a processing network is established, the concrete data types involved will be derived from other parts of the network.

In addition, the processing may involve generic container-like types (such as lists, maps) and even the SOAP message could be considered as a container type for some more specific message type, i.e., `SOAP<InventoryInfo>` where the SOAP message envelope is a container type and is parameterized by an inner type of `InventoryInfo`. This lets stages that only care about the outer envelope to disregard the inner data type and work with `SOAP<any>` message. This is called parametric polymorphism in programming languages.

If the dataflow network runs with non-variant and concrete data types, validating the connections between processing stages would be almost trivial via the classical Milner [70] algorithm that uses simple equivalence tests between the data types. However, the presence of covariance, parametrization and parametric polymorphism makes the equivalence test inadequate and often impossible. Therefore, validating a heterogeneous dataflow network requires a so-called **type inference** algorithm to work out the types in such networks and by doing so, the validity of the network is tied to the existence of valid typing of its components by either finding the concrete data types and/or not finding contradiction between the bounds of parametric types they have.

[17] In everyday terms, if I'm expecting fruit in general, I can work with both apples and oranges since apples and oranges are both fruits.

3.2.2 Literature review

Type inference is a well-studied field of programming languages, types and computer science in general. Starting from the well-known work of Milner [70], hundreds of type inference algorithms were proposed over the past decades, some extending the unification algorithm while others using graphs to perform the inference operations [80]. Type systems with many flavors are considered in the academic literature: non-structural subtyping [81, 84, 85]; structural subtyping [82, 89]; virtual types [52]; coercions (automatic type conversion) [93]; constrained types [78]; recursion [55, 57]; and polymorphic types [57] to name a few. Some approaches described by these sources are required by the type inference algorithm of the dataflow framework, but they do not apply as a whole:

- subtyping: process data with more fields than needed;
- polymorphic types: work with container-like types;
- intersection and union types: in the form of structurally combining types;
- recursion: used in definition of trees;
- no coercions: except for the ability to use integer where a real is needed;
- no first-class function types: simple Unified Modeling Language-style
  (UML) graphical programming with only blocks and wires;
- no first-class structural decomposition: same as before; and
- no let-polymorphism: the compiler takes care of unique (and unambiguous) labeling.

Unfortunately, many of the existing algorithms are tied to a concrete type system (or programming language) and feature some properties that are not required by the target problem domain of coordinating data between IT systems of a focal consumer and its suppliers. The second problem with the existing algorithms is that one cannot just pick-and-chose elements from them, leaving out or "mocking" the unneeded parts because they either stop working or are incompatible (or in logical conflict) with elements from another algorithm. Third, many type-inference and type-system combinations have difficulty supporting the so-called **generic coordination steps**: these are nodes in the type relation graph

describing a network that usually do not place requirements on their input and output data types flowing through them but have to match types correctly in a transitive inference scenario. In other cases, the processing step may apply a co- or contravariant **type bound** on the input or output parameters; in structural terms, a lower bound indicates a minimum set of properties present on the data type while an upper bound indicates a maximum set of properties. Therefore, the intended usage domain requires a novel type inference algorithm to be designed from first principles while taking inspiration from the existing ones.

3.2.3 Solution approach

Pluggable type system The type inference algorithm is loosely coupled with the type system [24] by providing basic constructs for types and externalizing functions required by the inference operations.

A Type is expected to provide at least one property, the kind of the type:

- CONCRETE_TYPE: represents a plain old type (example: string);
- VARIABLE_TYPE: indicates that the exact type is not yet known; and
- PARAMETRIC_TYPE: represents a type having types as parameters of their own, sometimes called generics (example: collection of string).

The inference algorithm requires the following type-system-related functions to be provided:

- Compare(A, B): function to determine the relation between two concrete types (or the base types of two parametric types) as NONE, EQUAL, EXTENDS or SUPER.
- Intersection(A, B): function to return or create the intersection of the two concrete types if possible.
- Union(A, B): function to return or create the union of the two concrete types if possible.
- Arguments(A): function to return the arguments of a parametric type.

- Fresh(): create a fresh type as a VARIABLE_TYPE.

Since the inference algorithm does not directly deal with the internal structure of the types, the type system may or may not contain a Top or Bottom type.

Inference The new type inference algorithm is loosely based on the framework described in [84]. The inference algorithm maintains several data structures – these help track the constraints of type variables as well as the early detection of typing errors.

- Upper bounds associated with VARIABLE_TYPEs. These bounds represent the supertype of the variable while going up on the type hierarchy.
- Lower bounds associated with VARIABLE_TYPEs. The bound represent the subtype of the variable while going down on the type hierarchy.
- A list of reflexive relations that helps propagate the bound constraints and helps in elimination of duplicate inequalities. It contains the same inequality record structures as the main queue (see below) with the exception that the left and right sides always reference VARIABLE_TYPEs.

The inference algorithm works on a queue of type inequality records. Each record consists of the left side and right side of the inequality, both referencing a Type. In addition, a reference is stored to the place where the record was produced (e. g., which wire or expression). This information is used by the wire-type determination algorithm after the inference. The algorithm picks one record from the queue and based on the kinds of the two referred types, the following 7 possible cases are handled:

Case 1. If both represent Concrete types, only the relation check is performed which must yield EQUAL or EXTENDS. Otherwise, a typing error is reported.

Case 2. If both represent Parametric types, the following steps are taken:

1. The argument counts are compared, and if they differ, a typing error is reported.

2. The base types are compared similarly to Case 1.

3. The parameters are extracted pairwise and added to the inequality queue.

Case 3. If the left side is a VARIABLE_TYPE (T) and the right side is a concrete type, the following steps are taken:

1. For each relation in the reflexive list, find those relations where the right side is T, and try to create a union of the upper bound of the left side and of T.

2. The upper bound of T is unioned with the right side.

3. The lower bound of T and the right side are taken and added to the inequality queue.

Case 4. If the left side is a VARIABLE_TYPE (T) and the right side is a parametric type (C¡V$_i$¿), the following steps are taken:

1. If no parametric upper bound exists for T, steps 2–3 from Case 3 are executed.

2. Fresh type variables (U$_i$) are constructed for each argument (V$_i$) of the right side parametric type, and the pairs (V$_i$, U$_i$) are added to the inequality queue.

Case 5. If the right side is a VARIABLE_TYPE (T) and the left side is a concrete type, the following steps are taken:

1. For each relation in the reflexive list, find those relations where the left side is T and try to create an intersection of the lower bound of the right side and of T.

2. The lower bound of T is intersected with the left side.

3. The left side and the upper bound of T are taken and added to the inequality queue.

Case 6. If the right side is a VARIABLE_TYPE (T) and the left side is a parametric type (C¡V$_i$¿), the following steps are taken:

1. If no parametric lower bound exists for T, steps 2–3 from Case 5 are executed.

2. Fresh type variables (U_i) are constructed for each argument (V_i) of the right side parametric type, and the pairs (U_i, V_i) are added to the inequality queue.

Case 7. If both sides of the inequality are of VARIABLE_TYPE (T, U), the upper and lower bounds of both variables are transitively combined with each other:

1. For each relation in the reflexive list, find those relations where the right side is T, and try to create a union of the upper bound of the left side and of T.

2. For each relation in the reflexive list, find those relations where the left side is U, and try to create an intersection of the lower bound of the right side and of U.

3. For each of the lower bound of T and upper bound of U, add the pair (T_{lb}, U_{ub}) to the inequality queue.

The algorithm terminates if no inequality relation remains or any of the union/intersection functions fail.

The flowchart of the whole inference algorithm can be seen in Figure 22.

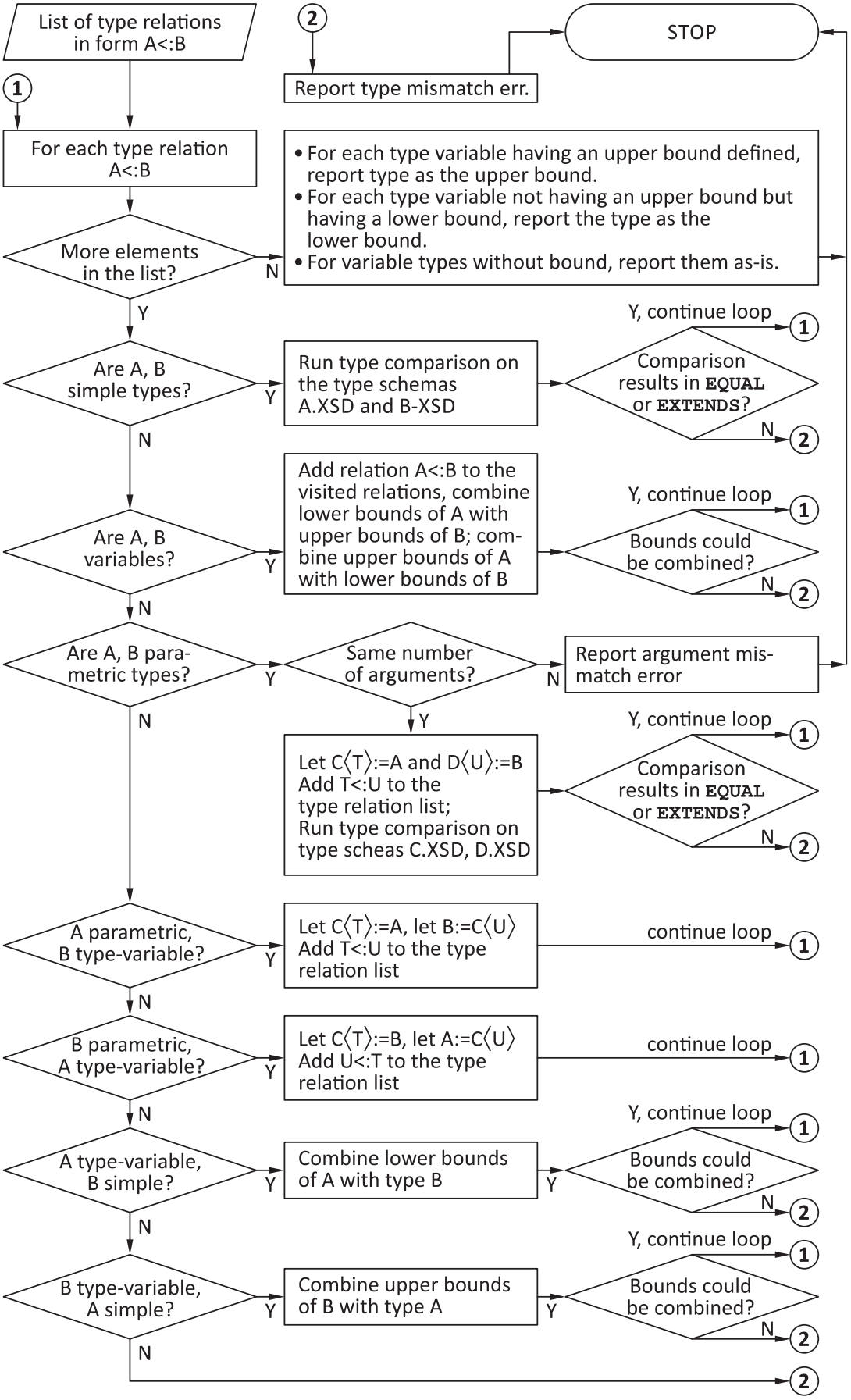


Figure 22: Flowchart of the type-inference algorithm.

Wire-type determination Once the inference algorithm is completed, the upper and lower bound structures contain the information about the inferred type of each expression or wire of the original input inequalities. In order to assign the most specific type to each of these, the bound structures must be searched for possible concrete or VARIABLE_TYPE results. The following algorithm searches for a most specific type in the structures for type T:

1. Check the upper bounds of T and look for a non VARIABLE_TYPE bound. Go to step 3 if found.
2. Check the lower bounds of T and look for a non VARIABLE_TYPE bound. Go to step 3 if found.
3. If neither of steps 1 and 2 produced results, the output type is T.
4. If the type found is non-parametric, it will become the output type.
5. If the type found is parametric, the output will be a copy of this type and each type argument is evaluated recursively by the algorithm again.

This algorithm is evaluated for both sides of the type inequality. The resulting types U and V are compared to each other to determine the final wire type:

- If any of these types is a VARIABLE_TYPE, the other type is used as the final type.
- If both are concrete types, the final type is determined by the type system comparison function.
- If both are parametric types, the type with the most specific type arguments is chosen recursively, determined by this algorithm.

Limitations The type inference algorithm is simpler than most existing algorithms, since it does not use function types (arrow types), automatically inserted type conversions (coercions) or the ”dot into” operator (structural decomposition).

A shortcoming of the algorithm in the prototype is that it requires parametric types compared against each other to have the same amount

of type arguments. This property comes from the inability of the XML-Schema-based type system (such as 3.1.3) to represent generic type information by default, since the structural inheritance might not be encoded in the XML Schema at all. To illustrate the problem, let us consider the following definition:

```
Map<Key, Value> extends
Collection<Pair<Key, Value>>
```

If we take `T := Map<Key, Value>` and `U := Collection<Pair<Key, Value>>` and pass the inequality (T, U) to the inference algorithm, it will report a typing error. The proper resolution of this issue is still subject to research.

The underlying XML-schema-based type system does not use composite union or intersection types [28] as individual types but rather creates a structural intersection and union of existing types. The drawback is that these structures usually do not have any human-understandable name associated with them (e. g., they are named `intersection_512` or `union_32`).

Another shortcoming is that the discovery of typing error and reporting depends on the initial order of inequalities in the queue, and might point to any element of a chain of bindings in the original dataflow network. Many inference algorithms exhibit this kind of property and several fixes have been proposed over the years [46].

3.2.4 Application of the results

As part of the ADVANCE Flow Engine mentioned in section 3.1.4, dataflow networks designed (and wired together) have to be verified before the engine can execute the flow. In generic, block-and-wire based editors, it would be easy to connect incompatible inputs and outputs by accident or deliberately unless the editor gives some form of feedback to the user. For example, the editor of the Flow Engine changes the colors of the (directed) wires (also called bindings) to red, and when these wires are selected, it presents a short error message about what the underlying data types were (Figure 23).

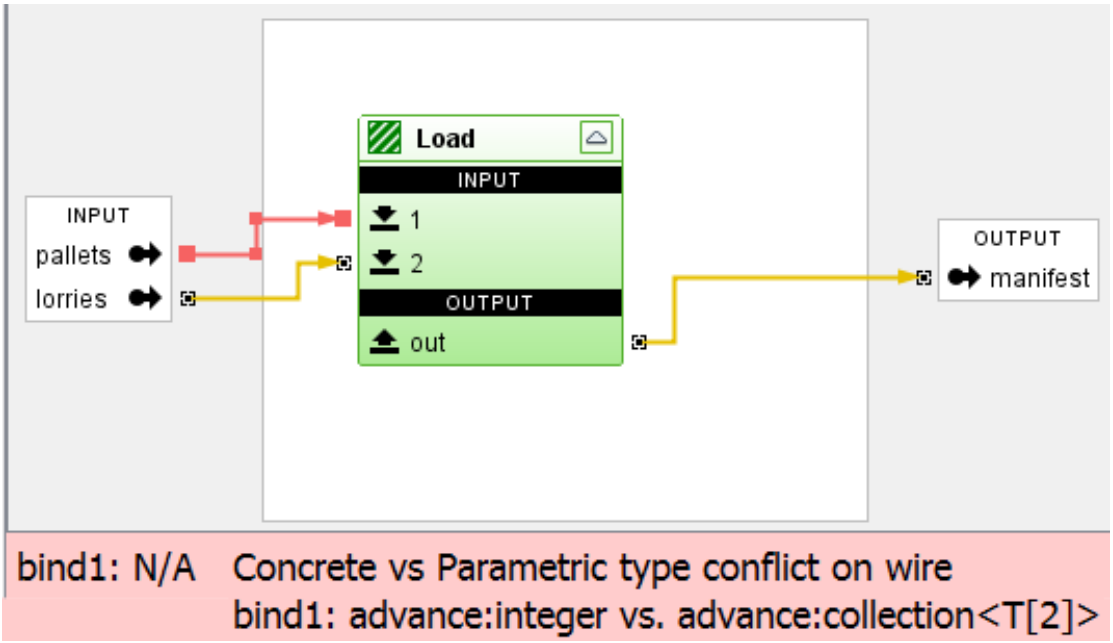


Figure 23: Type incompatibility when combining different sources.

In this setup, the `Load` block takes two inputs: the list of pallets and the list of lorries available, and should generate manifests specifying which lorry will hold which pallets. Unfortunately, the `pallets` input to the network is not a list of some pallet object but an integer (likely of the number of available pallets) instead of each individual pallet object expected. Figure 23 also demonstrates another error case when a block itself does not have constraints on the accepted data types. The `Load` block expects a collection of some generic datatype `T` (indexed with 2 since the type name T may appear multiple times, yet, it may designate different concrete types after the inference) that is to be concretized by the correct type-flow of the entire network eventually.

Figure 24 shows a case when all source and output data types are correctly matched and wired together:

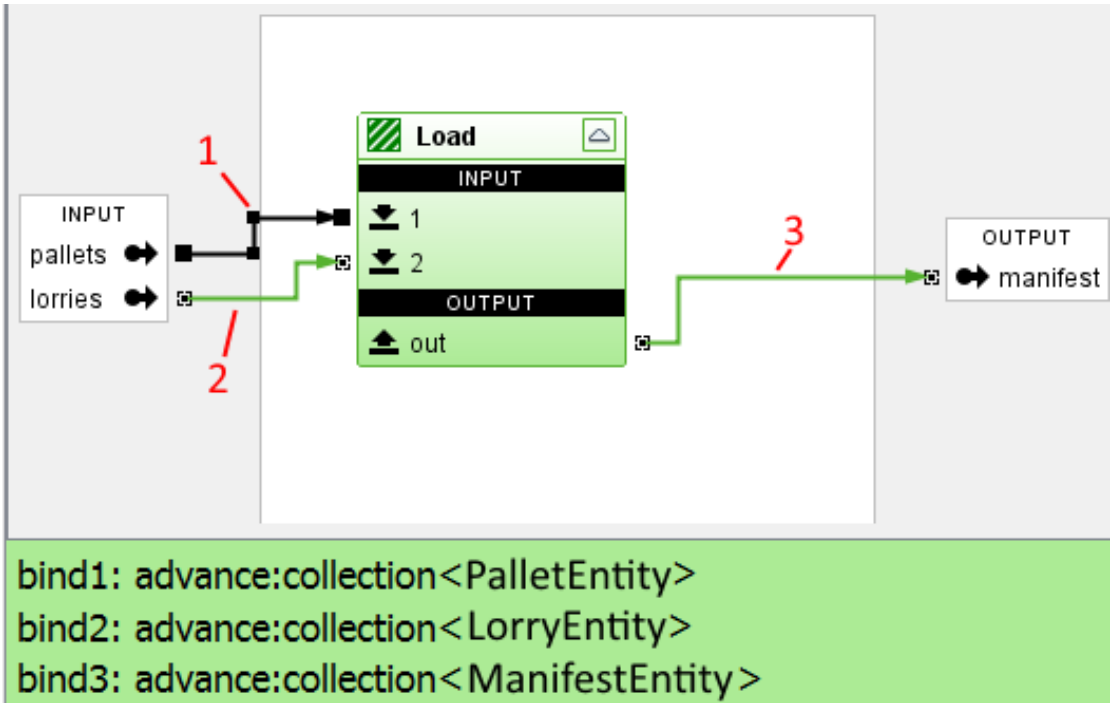


Figure 24: Data sources correctly wired and the inferred data types.

3.2.5 Summary

Thesis 3: Establishing a chain of data- and event-processing within the (focal) manufacturing company or in connections with suppliers of material components requires the transformation of the inputs into useful information supporting the day-to-day operations and decisions where the composition of the processing logic itself is also subject to frequent changes. This can be accomplished by a form of validation on how the various input sources are connected together in an usually complicated data processing graph. The dataflow between processing steps should be matched against each other by considering the covariance nature of the link itself in addition to supporting so-called generic coordination steps. Given the canonical representation of the data types involved in the information exchange and the service for evaluating the relation between these data types, a novel data type inference algorithm should be employed to match and verify the data connections between the processing steps by supporting

1. **steps requiring exact data types;**
2. **steps with constrained or unconstrained so-called generic data types; and**

3. **nested, container-like datatypes themselves composed out of (1) and (2).**

The algorithm takes a list of pairs of datatype definitions derived from the input(s) and output(s) of participating processing steps and iteratively

a. **verifies that exact datatypes form a covariant relation;**
b. **verifies that composite datatypes match on their base datatype, then formulates new relations out of the inner datatypes and adds them to the list; and**
c. **collects the so-called bounds of generic datatypes verifying that the covariant relation holds transitively between the upper- and lower-bounds.**

The output of the type-inference algorithm, on one hand, is the success or failure of verifying the processing graph in its entirety, and, on the other hand, it is either the inferred (or exact) datatype of each connection, or an explanation of where and why connections are incompatible (by showing the mismatch via the canonical datatype representation).

Related primary publication: [3], [15]
Related additional publications: [8], [19]

3.3 Driving information exchange and processing via reactive dataflows

3.3.1 Introduction & problem statement

In modern industrial environments, large amounts of data have to be channeled and processed – much of them have to succeed without significant time lag or risk of congestion or data loss. This calls for a solution which is resource-lean, capable of high throughput and allows an immediate response. The **reactive paradigm**, where dataflows are processed (usually) in a push-based manner [43], can very well suit these requirements.

In classical, so-called **pull-based** data processing, the consumer of the data requests the next item from the producer in a synchronous manner, that is, the consumer has to wait until the data is ready to be returned. This has the drawback that the consumer cannot perform other

tasks while waiting for the data which often leads to wasting system resources. In contrast, the **push-based** operation means that consumers react in the presence of incoming data only, while outputs are simply sent without the emitting producer checking if the given data was successfully received by the recipient (i.e., a fire-and-forget type of output). Special mechanisms, such as buffering, can be implemented as needed to prevent data loss, relieving most of the network of unnecessary computational demands. Both push- and pull-based data processing can be expressed in a declarative fashion which allows the underlying library or framework to perform certain optimizations that **adaptively switch between the two approaches**[18].

The reactive design principle mainly focuses on the lower levels of software development and requires mostly a constant definition of dataflows that do not and cannot change structurally once implemented. However, when the structure of such dataflows is subject to change, updating the lower level software solution may become infeasible. However, the declarative nature of such reactive flows enables building them in a visual fashion and consequently, such visual editors can become the part of services offered by a Cloud provider on top of the usual virtualized hardware and software environment the reactive approach can utilize more effectively. The ability to dynamically modify a dataflow of a processing network is, again, a productivity tool that doesn't need personnel trained in general purpose programming but mostly in the viewpoint (and mantra) of **"everything is a stream"**.

3.3.2 Literature review

Reactive or push-based programming [43] is not new as a paradigm, but only recent advances in the mainstream of imperative programming languages and libraries made it applicable in practice. A well-known C# [39] library is Reactive Extensions [63] with its ability to express and compose push-based operations with the same Language Integrated Query (LINQ) [68] that is generally used for pull-based data processing. In contrast, one of the most widely used programming languages, Java [79], does not provide much support for push-based operations per se – a possible workaround would be, e.g., the re-use of FlumeJava [31] or Frappé [35], originally meant to support parallel processing. This, how-

[18]For example, see (akarnokd.blogspot.hu/2016/03/operator-fusion-part-1.html) by the author.

ever, is clearly a laborious (and not very efficient) undertaking, suggesting the development of a reactive framework for Java from the ground up.

Often, simpler or smaller algorithms and data processing logic available on the programming language level are organized into larger entities called (processing) **blocks** in higher-level programming environments (i.e., graphical dataflow designers). These blocks define their data inputs and outputs via **ports** similar to how input and output parameters are defined in a mainstream programming language such as Java. The higher-level abstraction provided by these blocks allows the assembling of dataflow networks in a more economic fashion, among other things, with respect to development time. Such block-based and dataflow-oriented languages and libraries exist with similar aims: the LabView G [76] programming environment, and a recent development, the Task Parallel Library Dataflow Framework (TPL Dataflow, [92]). TPL dataflow employs a different value distribution strategy than most push-based solutions: only a single block can own a data value at the same time, requiring a – sometimes sophisticated – negotiation protocol between the blocks. While these block-based environments may prove effective in their intended range of use, it may be difficult to meet some of the key requirements inherent to the logistics-centered application in scope of the dissertation.

3.3.3 Solution approach

In order to build a dynamic dataflow network, operations on the data streams need to be encapsulated into computational units, so-called blocks. From the network designer's perspective, blocks are like black-boxes with well-defined input-, output- and execution semantics, and with the help of a backing type system and type inference, the construction and maintenance of such networks become simple tasks by using graphical or UML-style editing tools.

Implementing a block-based dataflow framework on top of a reactive programming framework (such as [68]) grants several advantages:

- Data processing occurs only when the data is actually available. No blocking or idle resource consumption is taxing the system.
- A large set of reactive can be provided to implement the block internals.

- Such frameworks have established operational semantics and reliable properties.

However, disadvantages may arise:

- Most reactive operators work with a limited (typically small) number of inputs, and provide a single output channel only. Should certain outputs be distributed to multiple blocks, appropriate replication blocks have to be used.
- The customized calculus required to execute most reactive operators needs to be written in the programming language the framework is implemented in. These must be ”hard coded” when the blocks are implemented.
- Regular chains of reactive operations are strictly typed and they are built up by the developers. Using the framework in dynamic environments requires loose typing and on-demand attack and detach semantics.

Given the limits of building composite processing structures with elementary blocks, developers will sooner or later face the need to implement their own specialised blocks. Aside from implementing the actual processing routines, the following tasks are necessarily involved:

- Defining the input and output parameters of the block, including their type.
- Defining the points of execution when the block is triggered (these are referred to as Schedulers).
- Defining which input needs to be present in order to execute the block-internal routines.

Certain blocks may need to work independently of their inputs, while others may only run when all inputs are available. Implementing the latter may become quite challenging, since data in the reactive framework may arrive at any time, and no guarantees are given for simultaneous reception of data once several different sources are involved. Most operations, nevertheless, require multiple inputs to be available before commencing, requiring buffering and collection – in response to this common need, most reactive frameworks offer such blocks out-of-box.

Blocks define their inputs and outputs with the same Observable – Observer pattern as the underlying reactive framework, making it simple to interface and use existing operators from these frameworks. This allows the so-called diagnostic ports of the block to be implemented with ease: each input and output parameter has a shadow parameter which places a projection of the incoming or outgoing data on these parameters with the same Observable – Observer pattern. When the run-time debugging of a block is needed, the debuggers may subscribe to these ports and receive a timestamped value for every data packet going in or out of the block.

Reactive frameworks are usually strongly typed (e. g., `Observable<Integer>`) but since block input and output parameters need to be bound dynamically (once the network description is compiled), these types need to be reduced to the most generic type (i. e., practically, everything becomes `Observable<Object>`) in order to allow the construction of the network on the fly in the host programming language. However, type safety needs to be retained, hence the need of a type inference algorithm of 3.2.3 running before the parameters of blocks are bound.

An important property of the blocks are the locations where their internal routines are executed. These are called Schedulers in most reactive frameworks. The blocks may use the following four kinds of schedulers:

- NOW scheduler: whenever data arrives at an input, the associated computations are carried out on the same scheduler that sent the data. This kind may be used for fast computations and to avoid unnecessary task-switching in the underlying operation environment.
- SEQUENTIAL scheduler: certain blocks may need to be executed in a globally sequential order. These blocks may work with resources outside the dataflow framework or use backchannels to communicate with each other and require a certain temporal order of execution to work properly.
- CPU (Central Processing Unit) scheduler: computationally intensive blocks may want to use all available CPU power. This is usually implemented with a fixed-size threadpool.
- IO (Input – Output) scheduler: some blocks may need to interface with synchronous data sources (such as databases), and usually stop the current thread until data becomes available.

Such behavior, if run on any other scheduler, may block the entire dataflow network processing. Therefore, a separate, larger threadpool is used where these blocks may be executed without disadvantageous consequences. Usually, reactive frameworks operate only when data is available, requiring special care when constants are bound to inputs of blocks that would also have non-constant (reactive) inputs. In this case, the constants need to be re-emitted ”manually” so that their reception coincides with that of non-constant input data. This can be effected in several ways:

- One of the reactive sources may be bound into a special constant block which emits its value once the reactive data arrives (i. e., the re-emission of the constant is triggered externally).
- The compiler of the dataflow network generates special inputs, and the input parameter access is generalized in a way that the block-internal implementation does not need to know whether the given data is constant or reactive.

The dataflow framework presented uses both techniques. Input parameters bound by literal constants are handled specially by the compiler. Effectively constant inputs (values calculated once by other network blocks without further effects) require the network designer to place so-called repeat-latest-value blocks.

An important simplification of the dataflow network is that block inputs are always covariant (convert from wider to narrower types) whereas block outputs are contravariant (convert from narrower to wider). This simplifies the definition of the blocks and has the consequence that bindings between parameters can be represented by the same source – destination type inequality for the type inference algorithm 3.2.3. Furthermore, the data propagating through the network should be treated as immutable: blocks need to create full copies and modify those copies instead of the original incoming values.

The **ADVANCE Flow Engine** is the prototype implementation of an XML-based, asynchronous, reactive, strongly typed framework combining and employing the new methods and algorithms described in the previous sections. It gives the foundation for sophisticated, domainspecific functions and operations of the ADVANCE project[19], and thus focuses on

[19] http://www.advance-logistics.eu, last accesses July, 2015.

typical data flows in logistics networks, specifically, the multi-participant hub-and-spoke case targeted by the project.

Dataflow description The ADVANCE Flow Engine is programmed via XML based flow description, i. e., the source code for the dataflow network to be executed. To describe the network, the flow description offers 4 basic building elements:

- constants,
- regular blocks,
- composite blocks, which may contain sub-networks, and
- bindings (or wires) to connect blocks, constants and composite blocks.

Composite blocks, similar to regular blocks, can define input and output parameters, allowing the network designer to create subroutine-like networks or simply group parts of a larger network into more easily maintainable logical units. An XML-coded example of a network is shown in Listing 1, and graphically in Figure 26.

Network desingners are not required to specify the dataflow networks in XML and text format. The ADVANCE Project contains a tailored, UML-style graphical editor for the purpose (Figure 25).

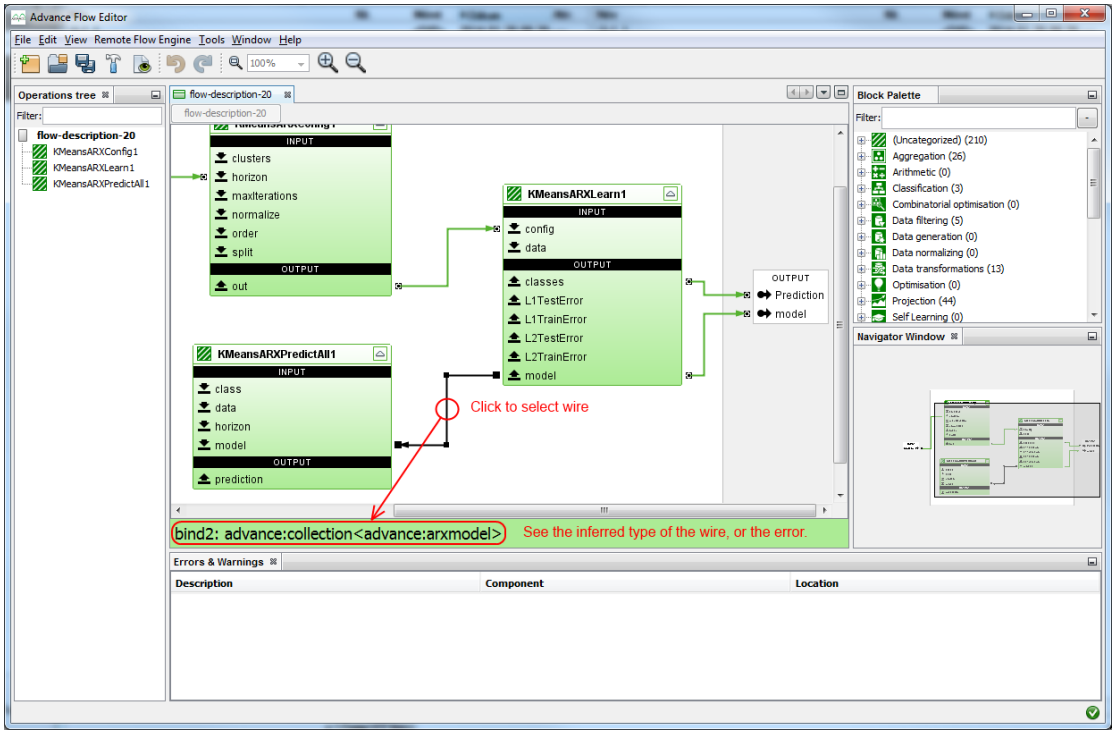


Figure 25: The graphical dataflow editor of ADVANCE.

Flow compilation The compiler verifies the dataflow description, instantiates the blocks from the reactive framework, then binds the appropriate inputs and outputs together. The verification step ensures that:

- regular blocks referenced in the flow description are available in the engine context,
- bindings refer to existing constants, blocks, composites, input and output parameters,
- binding connections are correct, e. g., two inputs or two outputs are not bound together, and
- the type flows on the bindings are correct (upon running the type inference algorithm).

```
<flow-description>
  <composite-block id='1'>
    <block id='2' type='Merge'/>
    <constant id='4' type='advance:integer'>
      <integer>1</integer></constant>
    <constant id='5' type='advance:string'>
```

```
      <string>Hello world!</string></constant>
    <composite-block id='6'>
      <input id='in' type='advance:object'/>
      <block id='3' type='WriteLine'/>
      <bind id='7' source-parameter='in'
        destination-block='3' destination-parameter='in'/>
    </composite-block>
    <bind id='8' source-block='4'
      destination-block='2' destination-parameter='in1'/>
    <bind id='9' source-block='5'
      destination-block='2' destination-parameter='in2'/>
    <bind id='10' source-block='2'
      source-parameter='out' destination-block='6'
      destination-parameter='in'/>
  </composite-block>
</flow-description>
```

Listing 1: Example network flow description XML

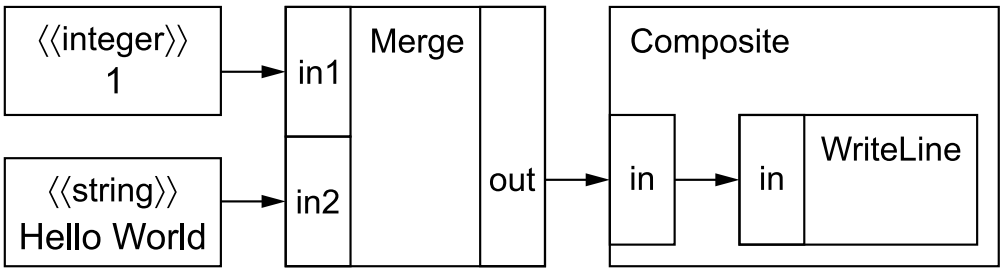


Figure 26: Example network with constants, blocks, composite blocks and bindings.

The type inference algorithm requires a set of inequalities to work with. The compiler takes all bindings of the flow description and extracts the declared types of both ends. Listing 1 will yield the following set of inequalities:

`advance:integer >= T1` (binding #8)
`advance:string >= T1` (binding #9)
`T1 >= T2` (binding #10)
`T2 >= advance:object` (binding #7)

Upon successful verification, the regular blocks are instantiated and parametrized by the compiler. Constants are inlined into these blocks. At this stage, the structures defined by composite blocks are ignored, and blocks are directly bound together through their input and output

parameters by walking the graph of the bindings in the flow description. For example, in Figure 26, the compiled network contains two blocks: Merge and WriteLine. Merge is compiled with the two constants as its parameters, and the in parameter of WriteLine is directly connected to the out parameter of the Merge block. The compiled network is then available for execution by the runtime environment.

Runtime environment In order to support multiple, parallel networks, the compiled flow descriptions are grouped into so-called realms. Each realm has its own security boundary, access rights to resources and established user roles regarding who can see, modify and run the particular network in the realm. Since the reactive framework only operates when data is available, compiled blocks with constant-only inputs need to be bootstrapped. If a realm is started, these blocks are triggered once.

The runtime environment offers the following services to a compiled dataflow network:

- management of startup and shutdown of the realm, including saving and restoring the internal states of the compiled blocks,
- access to various Scheduler environments to allow the blocks to perform parallel operations within themselves,
- a common interface to access a datastore where the blocks can store their intermediate results or access external resources indirectly (such as database connections, local files, etc.), and
- information about the block itself: the origin of the block in the flow description, the declared and runtime type information of its parameters, etc.

The runtime environment treats the inputs and outputs of the outermost composite block as the global input and output parameters of each network. The engine provides means to send and receive data to and from these parameters in a programmatical fashion. Debug access is provided to all input and output parameters of all compiled blocks.

Limitations The dataflow framework and the blocks were never intended to be a fully blown, Turing-complete programming language but rather a

domain-specific programming framework to support dynamic construction of dataflow networks. Due to the several added layers in the data access and the fact that block internals are usually implemented in a pull-based manner (as opposed to the push-based manner of the surrounding framework), performance may deteriorate in certain situations.

Another issue may arise from the fact that arithmetic operations on basic data types such as integer and string may be cumbersome to design: a simple `(a + b) * c` expression requires 3 inputs providing a constant or a data source, 2 arithmetic operators and 3 bindings.

To help with this issue, certain blocks may offer scripting support by using a dynamically typed language such as JavaScript to provide a more concise way of defining these operations. Two drawbacks can be associated with this type of approach: i) script execution is even slower than any compiled host-language routine; and ii) joining the type checking of the script-external type system and the script-internal type system is a very complicated task.

The prototype implementation does not contain functions or dot-into (i. e., structural decomposition) operations as first class citizens. From a type inference perspective, they are relatively easy to implement, but from the network designer's perspective, they may appear quite different from the blocks-and-wires type of visual representation. Due to the lack of these language constructs, many blocks require so-called type tokens to be explicitly defined. For example, given a record structure with a complex type member, a block may be implemented that accesses that member and returns its value (e. g., a projection operator). If the dot-into operator was available, the return type of such a block could easily be determined by the type inference algorithm. Without the operator, however, the designer has to explicitly provide a type token for the member as another parameter, and the block internals would have to ensure the proper type is returned at execution time.

3.3.4 Application of the results

As part of the ADVANCE Flow Engine mentioned in section 3.1.4, the engine is programmed via XML-based flow description, i.e., the source code for the dataflow network to be compiled and executed. To describe the network, the flow description offers four basic building elements:

- constants,

- regular blocks,
- composite blocks, which may contain sub-networks, and
- bindings (or wires) to connect blocks, constants and composite blocks.

Composite blocks, similar to regular blocks, can define input and output parameters, allowing the network designer to create subroutine-like networks or simply group parts of a larger network into more easily maintainable logical units.

Network designers are not required to specify the dataflow networks in XML and text format. The ADVANCE Flow Engine contains a tailored, UML-style graphical editor for the purpose (Figure 25).

A Flow Engine instance hosts a set of so-called realms where an individual flow network runs and processes data (Figure 27).

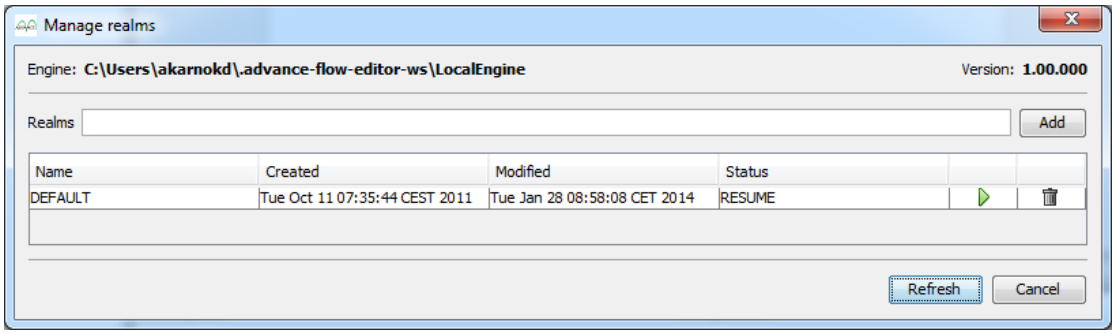


Figure 27: Execution realms of a Flow Engine instance.

Once the network has been designed, it can be uploaded into one of the realms and begin its execution (Figure 28):

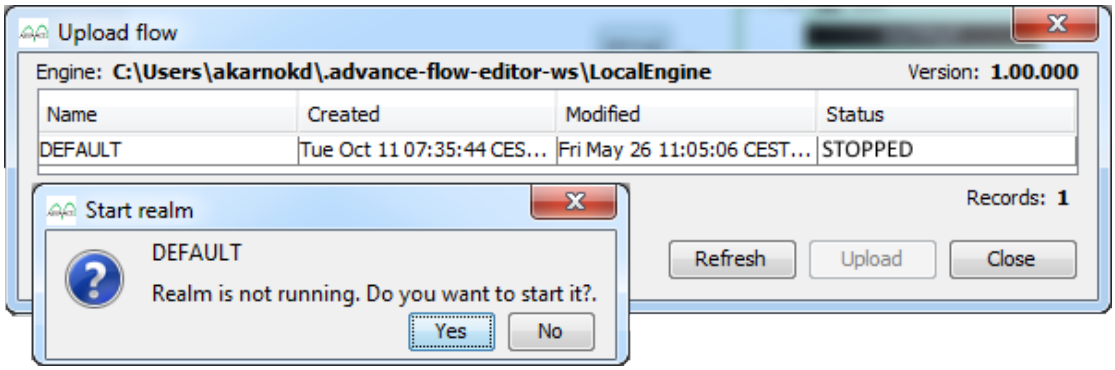


Figure 28: Upload a flow and start execution in a Flow Engine realm selected.

Inspecting and debugging flows are supported by the Flow Engine as well (Figure 29). The example shows a simple network where a button

is wired to a log output. Every time the button is clicked, the log lists the timestamp and content of the event generated by the button click. At the same time, the debug window of the engine lists which realm, component ant port produced an event. In practice, the Advance Block Visualization is not displayed because most processing blocks provided by the engine do not have visual outputs, yet, they can still be inspected via the debug window mentioned.

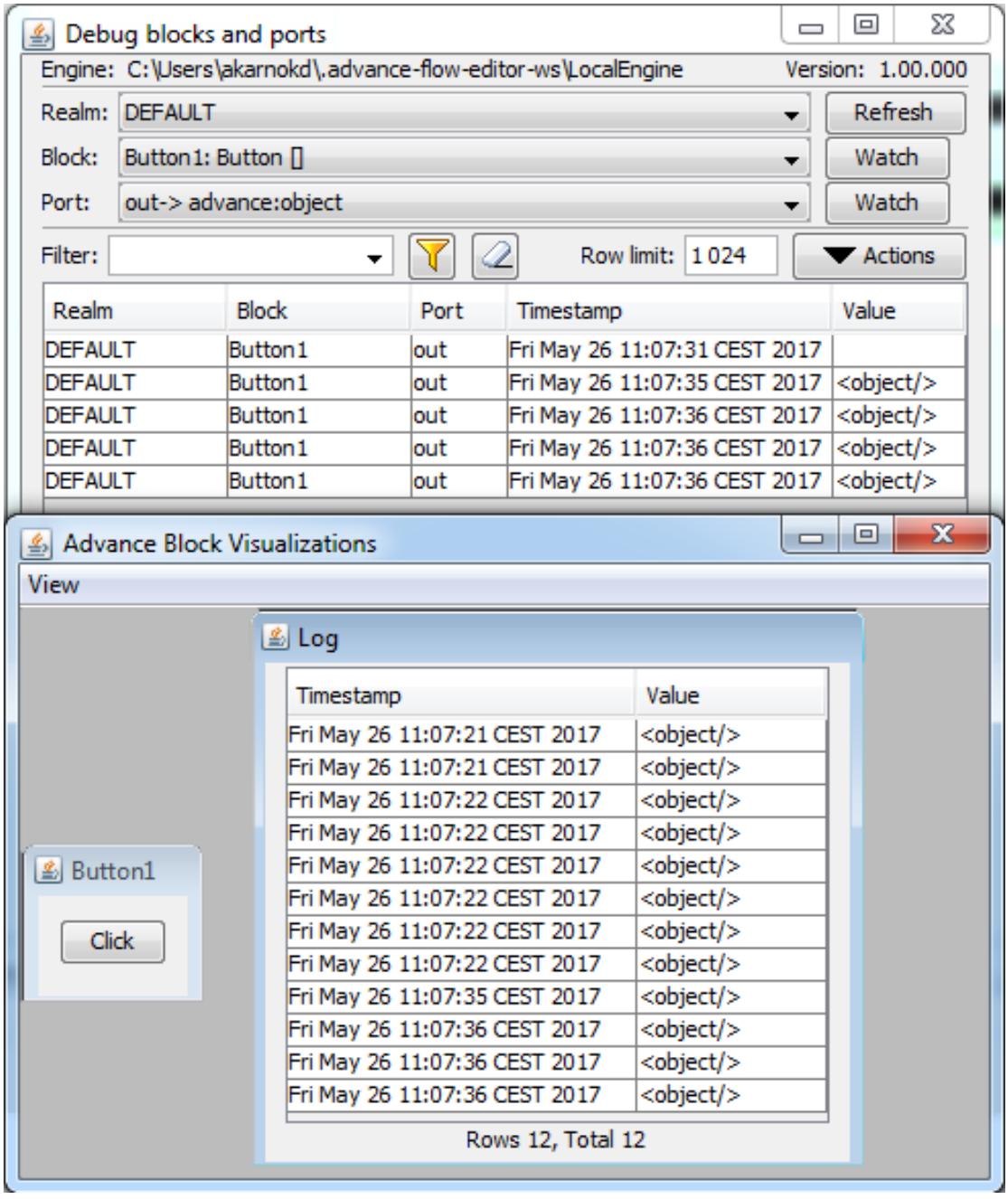


Figure 29: Displaying (debugging) the event flow in a Flow Engine Realm of a simple button-log dataflow network.

3.3.5 Summary

Thesis 4: Creating and executing the data- and event-processing computational network – which may be subject to frequent structural changes – within the (focal) manufacturing company and in connections with suppliers of material components should occur via

1. **a novel generic, declarative XML-based dataflow-network description, and**
2. **a novel transformation algorithm (compiler) that should translate the processing network definition into a machine-executable format,**

given the canonical datatype definitions and the results of the validation as well as the exact datatypes of the dataflow graph of the network. Due to the nature of the underlying availability of the data and events over time (and location), the executable format should be established on a novel composable, reactive and adaptively push- or pull-based computation and execution model where the processing stages themselves are represented as abstract computation and coordination blocks with well-defined (and canonically typed) input and output ports. The runtime environment supporting the executable format should be responsible for binding the input and output ports of these blocks according to the dataflow graph, and drive the data- and event delivery between them when the data- and event-processing is in operation. The reactive nature and transparent data- and event-interoperability should extend beyond an organizational border by defining service entry points that allow both retrieving the entry point's canonical datatype definition as well as enabling the data- and event-exchange itself. The service host is also required to apply proper authentication and authorization checks when these service entry points are accessed.

Related primary publication: [3], [8], [16]

Related additional publications: [1], [10], [15],

4 Future directions

Modern Information Technology can provide great value to the fourth industrial revolution that now embraces the Cloud and the ubiquitous availability of smart computing components in the manufacturing and logistics environments.

The main challenge, namely turning offline and paper-based material requisitions into a semi-automated digital process as described by **Thesis 1** is not the end, but a stepping stone into a human-assisted automated logistics solution that integrates data from all levels of manufacturing and logistics from within the participating companies. This level of integration will help with typical day-to-day management of the underlying material flow while providing detailed information and a toolset for resolving issues by manual intervention if necessary (i.e., cockpit task management). Although this idea has already been expressed by the Enterprise Integration movement, the difference is that the problem of improving the key performance indicators is addressed bottom-up by designing and implementing custom tailored models and systems (such as the **Logistics Platform**) instead of placing an ERP over everybody and trying to work and decompose the (complex) problem via top-down (and often buzzword-loaded) approach.

In the (digital) service industry, this bottom-up thinking led to the so-called **microservices** design principle. The idea is to compose the business logic or processes from services that are designed (and limited) to perform small, simple or dedicated tasks. Unlike previous approaches, such as the SOAP-based web service orchestrations and "classical" Enterprise Service Busses, the requirement of being streaming, (near) real-time makes previously existing technologies inadequate to solve the problems economically.

One major contribution towards microservice architectures is the reactive programming paradigm **Thesis 4** was built upon. In fact, the research and development work on Reactive4Java actually led to a small programming revolution on the mobile (Android) platform through a follow-up collaborative open-source library called RxJava[20]. With the help from companion libraries aimed at (streaming) networked dataflows, the application of the paradigm helps the software/service developers concentrate on declaratively specifying the data processing logic while enabling the

[20] https//github.com/ReactiveX/RxJava

underlying libraries and frameworks to optimize the resulting dataflows to achieve better (Cloud) resource utilization as well. The success of RxJava – of which the present author contributed about 80% of the code, documentation and the support time combined[21] – mostly in the form of disseminating and proving the usefulness of the reactive paradig, resulted in an industry standard specification for in-memory dataflows called Reactive-Streams now expected to be implemented by major library and framework providers and to be the starting point for new application designs. Along this angle, the novel approach (minimal data model and protocol) of **Thesis 1** fits nicely into this world and the solution can be reworked relatively easily.

The microservices design also prompts for solutions regarding the data types involved in a distributed and reactive dataflow. Being distributed, the concrete data types are no longer available at one location, and such extra information has to be available and/or sent along with regular data in some form or another. In addition, the rigid typing of classical SOAP messages limit the free consumption of services because it does not support covariance (i.e., consume services which provide way more data per message than actually necessary). Since most industrial data are specified as records or structs, structural typing and, consequently, the structural comparison between types enabled by the results of **Thesis 2** is becoming essential when designing new and modern solutions. (On a sidenote, several recent programming languages, such as the Pony[22] language, were designed with structural types in mind, reinforcing the utility of solutions presented in the thesis.)

Having a structural type system by itself is of not much use, and requires an application environment to function. Even though building reactive systems has become easier, the need for dynamically establishing dataflows without the need for software developers all the time is still present. Similar to the **ADVANCE Flow Engine** based on **Theses 3** and **4**, Cloud-embracing companies are providing similar visual editing tools to design and deploy fully reactive dataflows, for example, Spring Cloud Dataflow Server (see Section 24.2 Figure 11). Such editors have a natural need for proper type inference since the lower level language (such as Java) compiler cannot help with the required validation of the established

[21] Based on `git blame`, this number consists of about 65% contributed to versions prior to 2.x of the library and 95% of 2.x itself including its design.

[22] https://www.ponylang.org/

on-screen dataflow network. (Although the author contributed significant amount of code to their underlying reactive library, Reactor-Core[23], there is no actual evidence the editor or the underlying type-system/type-inference logic was inspired by or is in relation to the ADVANCE Flow Engine from this thesis.)

On general terms, it turned out that the first-class nature (requirement) of the Industry 4.0 prompted for a novel and better set of programming paradigms on the computing side as well, bringing the reactive programming paradigm into focus, and offers the option of establishing a strong foundation to build modern services, solutions and industry-specific business logic on top. However, the research on reactive streams is far from over as industrial application experience prompts for new problems to be solved. For example, the speed at which data is propagated may overwhelm consumers of data (or processing stages) leading to excess (computational) resource usage and instability. The approach **Thesis 4** took was to sample/page incoming data which requires conscious dropping of data or peculiar communication protocol design. The problem, called **backpressure**, was eventually solved through the Reactive-Streams Application Programming Interface (API) by including specific elements and rules to tackle the problem[24]. Consequently, the author identifies the following problems/domains requiring further research:

- Operator-fusion: executing reactive flows even more efficiently by exploiting sequential stages[25].
- Parallel-flows: better utilization of multi-core hardware[25].
- Reactive Remote/Interprocess communication: better isolation, more fluent interoperation between distributed components.
- Bi-directional reactive flows.
- Data/Computational resource lifecycle awareness.
- Computational context-aware flows.
- Vector/Signal dataflow: more data or no data per event.
- Reliable data delivery in multi-stage reactive flows.
- Latency-sensitive flows and guaranteed timely data delivery and timely reactions.

[23] https://github.com/reactor/reactor-core

[24] See Reactive-Streams goals.

[25] The development and evaluation of related solutions are already in progress in the RxJava 2.x and Reactor-Core 3.x libraries.

- Reactive-Library-as-a-service: pick-and-chose-style generation of reactive solutions from the properties above.

The novel scientific results presented in the dissertation provide a sound and foundational basis for tackling these challenges in the future.

Publications

Web-of-Science lectorated foreign language articles

[1] Monostori, L., Ilie-Zudor, E., Kemény, Z., Szathmári, M., and **Karnok**, **D**. Increased transparency within and beyond organizational borders by novel identifier-based services for enterprises of different size. *CIRP Annals – Manufacturing Technology*, 2009, 58(1):417–420. (DOI: 10.1016/j.cirp.2009.03.086), IF: **1.603**, Cites: **5**.

[2] Monostori, L., Kádár, B., Pfeiffer, A., and **Karnok**, **D**. Solution approaches to real-time control of customized mass production. *CIRP Annals – Manufacturing Technology*, 2007, 56(1):431–434. (DOI: 10.1016/j.cirp.2007.05.103), IF: **0.779**, Cites: **20**.

[3] **Karnok**, **D**., Kemény, Z., Ilie-Zudor, E., and Monostori, L. Data type definition and handling for supporting interoperability across organizational borders. *Journal of Intelligent Manufacturing*, 2016, 2:167–185. (DOI: 10.1007/s10845-014-0884-9), IF: **1.142**, Cites: **2**.

[4] Váncza, J., Egri, P., and **Karnok**, **D**. Planning in concert: A logistics platform for production networks. *International Journal of Computer Integerated Manufacturing*, 2010, 23(4):297–307. (DOI: 10.1080/09511921003630092), IF: **0.553**, Cites: **8**.

Lectorated foreign language articles

[5] Monostori, L., Váncza, J., Kis, T., Kádár, B., **Karnok**, **D**., Drótos, M., and Egri, P. Real-time, cooperative enterprises: Results of an industry – academia project. *Journal of Machine Manufacturing*, 2009, 49(E1):8–12.

[6] **Karnok**, **D**. and Monostori, L. Novel approaches for better transparency in production and supply chains. *Asian International Journal of Science and Technology in Production and Manufacturing Engineering*, 2009, 2(3):57–68.

Publications in Hungarian

[7] Monostori, L., Váncza, J., Kis, T., Kádár, B., **Karnok**, **D**., Drótos, M., and Egri, P. Valós időben együttműködő vállalatok: egy ipari–akadémiai projekt eredményei. *Gépgyártás*, 2008, 48(4):11–15.

[8] **Karnok**, **D**. Átláthatóság a gyártásban és a logisztikában: egy reaktív megközelítés. *Gépgyártás*, 2015, 55(2):73–77.

International conference or other articles

[9] Egri, P., **Karnok**, **D**., and Váncza, J. Information sharing in cooperative production networks. In: Monostori, L. (editor), *Preprints of the IFAC Workshop on Modelling, Management and Control*. Location: MTA SZTAKI, Budapest, Hungary, 14–16 November, 2007, pages 115–120. (ISBN: 978-963-311-366-0). Cites: **4**.

[10] Ilie-Zudor, E., Ekárt, A., Kemény, Z., **Karnok**, **D**., Buckingham, C., and Jardim-Goncalves, R. Information modelling and decision support in logistics networks. In *5th international conference on experiments process system modeling simulation optimization*. Location: Athens, Greece, 3–6 July, 2013, pages 279–286. (ISBN: 978-618-80527-1-0).

[11] Ilie-Zudor, E., Szathmári, M., Kemény, Z., **Karnok**, **D**., and Monostori, L. Identity-based, item-centric tracking platform for logistics applications. In *Proceedings of the International Workshop on Harbour, Maritime & Multimodal Logistics Modelling and Simulation (HMS2008)*. Location: Campora San Giovanni, Italy, 17–19 September, 2008, pages 10–19.

[12] Monostori, L., Ilie-Zudor, E., Kádár, B., **Karnok**, **D**., Pfeiffer, A., Kemény, Z., and Szathmári, M. Novel it solutions for increasing transparency in production and in supply chains. In: Spath, D., Ilg, R., and Krause, T. (editors), *ICPR 21, Proceedings of the 21st International Conference on Production Research*. Location: Fraunhofer IRB Verlag, Stuttgart, Germany, 7 July – 4 August, 2011, pages 1–6. (ISBN: 978-3-8396-0293-5).

[13] Monostori, L., Váncza, J., Kis, T., Erdős, G., **Karnok**, **D**., and Egri, P. Real-time, cooperative enterprises for customized mass production; challenges and solution approaches. In: Chung, M. J. and Misra, P. (editors), *Proceedings of the 17th IFAC World Congress*. Location: Seoul, South Korea, 6–11 July, 2008, pages 13851–13856. (ISBN: 978-1-605-60758-0).

[14] Monostori, L., Váncza, J., Kis, T., Kádár, B., **Karnok**, **D**., Drótos, M., and Egri, P. Novel it approaches and solutions towards real-time, cooperative enterprises; plenary paper. In *13th IFAC Symposium INCOM 2009, Control Problems in Manufacturing*. Location: Moscow, Russia, 3–5 June, 2009, pages 724–731.

[15] **Karnok**, **D**. and Kemény, Z. Definition and handling of data types in a dataflow-oriented modelling and processing environment. In: Ilie-Zudor, E., Kemény, Z., and Monostori, L. (editors), *MITIP 2012. 14th International Conference on Modern information Technology in the Innovation Processes of the Industrial Enterprises*. Location: MTA SZTAKI, Budapest, Hungary, 24–26 October, 2012, pages 561–574. (ISBN: 978-963-311-373-8).

[16] **Karnok**, **D**. and Kemény, Z. Framework for building and coordinating information flows in logistics networks. In: Ilie-Zudor, E., Kemény, Z., and Monostori, L. (editors), *MITIP 2012. 14th International Conference on Modern information Technology in the Innovation Processes of the Industrial Enterprises*. Location: MTA SZTAKI, Budapest, Hungary, 24–26 October, 2012, pages 551–560. (ISBN: 978-963-311-373-8).

[17] **Karnok**, **D**. and Monostori, L. Logistics platform: developing a distributed, cooperative production and logistics system. In: Váradi, K. and Vörös, G. (editors), *Proceedings of the 6th Conference on Mechanical Engineering*. Location: Budapest University of Technology and Economics, Budapest, Hungary, 29–30 May, 2008, Paper G-2008-C-4. (ISBN: 978-963-420-947-8).

[18] **Karnok**, **D**. and Monostori, L. Novel approaches for better transparency in production and supply chains. In *42nd CIRP Conference on Manufacturing Systems, Sustainable Development of Manufacturing Systems*. Location: Grenoble, France, 3–6 June, 2009.

[19] van Blommestein, F., **Karnok**, **D**., and Kemény, Z. In: Lee, I. (editor), *RFID Technology Integration for Business Performance Improvement*, chapter Meta-data alignment in open Tracking & Tracing systems. IGI Global, July 2014, pages 121–139. (DOI: 10.4018/978-1-4666-6308-4.ch006), (ISBN: 978-1-4666-6308-4).

[20] Váncza, J., Egri, P., and **Karnok**, **D**. Planning in concert: a logistics platform for production networks. In: Maropoulos, P. and Newman, S. (editors), *Proceedings of the 4th International Conference on Digital Enterprise Technology (DET2007)*. Location: University of Bath, Bath, England, 19–21 September, 2007, pages 462–470. (ISBN: 978-0-86197-141-1).

Bibliography

[21] Akkermans, H. A., Bogerd, P., Yücesan, E., and van Wassenhove, L. N. The impact of ERP on supply chain management: Exploratory findings from a european delphi study. *European Journal of Operational Research*, 2003, 146(2):284 – 301. (DOI: 10.1016/S0377-2217(02)00550-7).

[22] Albrecht, M. Supply chain coordination mechanisms: New approaches for collaborative planning, volume 628. Springer Science & Business Media, 2009.

[23] Arenas, M. and Libkin, L. A normal form for XML documents. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '02. Location: Madison, Wisconsin, 2002, pages 85–96, New York, NY, USA. ACM. (DOI: 10.1145/543613.543625), (ISBN: 1-58113-507-6).

[24] Bracha, G. Pluggable type systems. In *In OOPSLA'04 Workshop on Revival of Dynamic Languages*. 2004. (pdf)

[25] Brettel, M., Friederichsen, N., Keller, M., and Rosenberg, M. How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective. *International Journal of Science, Engineering and Technology 8 (1), 37*, 2014, 44. (pdf)

[26] Buxey, G. Strategy not tactics drives aggregate planning. *International Journal of Production Economics*, 2003, 85(3):331 – 346. (DOI: 10.1016/S0925-5273(03)00120-8).

[27] Cachon, G. Supply chain coordination with contracts. In: de Kok, A. and Graves, S. (editors), *Supply chain management: Design, coordination, cooperation. Handbooks in OR and MS*, volume 11. Location: New York, USA, 2003, pages 229–339.

[28] Carlier, S. and Wells, J. B. Expansion: the crucial mechanism for type inference with intersection types: A survey and explanation. *Electron. Notes Theor. Comput. Sci.*, July 2005, 136:173–202. (DOI: 10.1016/j.entcs.2005.03.026).

[29] CERN. Computing in CERN. (link)

[30] Chae, B. K. Developing key performance indicators for supply chain: an industry perspective. *Supply Chain Management: An International Journal*, 2009, 14(6):422–428. (DOI: 10.1108/13598540910995192).

[31] Chambers, C., Raniwala, A., Perry, F., Adams, S., Henry, R. R., Bradshaw, R., and Weizenbaum, N. Flumejava: easy, efficient data-parallel pipelines. In *Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '10. Location: Toronto, Ontario, Canada, 2010, pages 363–375, New York, NY, USA. ACM. (DOI: 10.1145/1806596.1806638), (ISBN: 978-1-4503-0019-3).

[32] Cho, S.-w. and Pak, M.-s. An integrative view on cyber threat to global supply chain management systems. *Journal of Korea Trade*, 2011, 15(3):55–87.

[33] Christensen, C. The innovator's dilemma: when new technologies cause great firms to fail. Harvard Business Review Press, 1997.

[34] Colombo, A. W., Bangemann, T., Karnouskos, S., Delsing, J., Stluka, P., Harrison, R., Jammes, F., and Lastra, J. Industrial cloud-based cyber-physical systems. *The IMC-AESOP Approach*, 2014.

[35] Courtney, A. Frappé: Functional reactive programming in java. In *In Proceedings of Symposium on Practical Aspects of Declarative Languages. ACM*. pages 29–44, 2001. Springer-Verlag.

[36] Dong, Y. and Xu, K. A supply chain model of vendor managed inventory. *Transportation Research Part E: Logistics and Transportation Review*, 2002, 38(2):75 – 95. (DOI: 10.1016/S1366-5545(01)00014-X)

[37] Drótos, M., Erdős, G., and Kis, T. Computing lower and upper bounds for a large-scale industrial job shop scheduling problem. *European Journal of Operational Research*, 2009, 197(1):296–306.

[38] Duta, A. C., Barker, K., and Alhajj, R. Ra: An XML schema reduction algorithm, 2006. (pdf)

[39] ECMA International. C# language specification. Standard ECMA-334, 4th Edition, 2006. (pdf)

[40] Egri, P. Coordination in production networks. PhD thesis, Eötvös Loránd University, Budapest, 2008. (pdf)

[41] Egri, P. and Váncza, J. Incentives for cooperative planning in focal supply networks. In *IWES 2006. 6th international workshop on emergent synthesis. Tokyo, 2006.* Location: Tokyo, Japan, 2006, pages 17–24.

[42] Egri, P. and Váncza, J. A logistics framework for coordinating supply chains on unstable markets. In: Cunha, P. and Maropoulos, P. (editors), *Digital Enterprise Technology*. Springer, 2007, pages 59–66.

[43] Elliott, C. M. Push-pull functional reactive programming. In *Proceedings of the 2nd ACM SIGPLAN symposium on Haskell*, Haskell '09. Location: Edinburgh, Scotland, 2009, pages 25–36, New York, NY, USA. ACM. (DOI: 10.1145/1596638.1596643), (ISBN: 978-1-60558-508-6).

[44] Fleischmann, B. and Meyr, H. Planning hierarchy, modelling and advanced planning systems. In: de Kok, A. and Graves, S. (editors), *Supply chain management: Design, coordination, cooperation. Handbooks in OR and MS*, volume 11. Location: New York, USA, 2003, pages 457–523.

[45] Gunasekaran, A. and Kobu, B. Performance measures and metrics in logistics and supply chain management: a review of recent literature (19952004) for research and applications. *International Journal of Production Research*, 2007, 45(12):2819–2840. (DOI: 10.1080/00207540600806513).

[46] Heeren, B. J. Top quality type error messages. PhD thesis, Universiteit Utrecht, The Netherlands, 2005. (pdf)

[47] Helo, P., Suorsa, M., Hao, Y., and Anussornnitisarn, P. Toward a cloud-based manufacturing execution system for distributed manufacturing. *Computers in Industry*, 2014, 65(4):646 – 656. (DOI: 10.1016/j.compind.2014.01.015).

[48] Hon, K. Performance and evaluation of manufacturing systems. *CIRP Annals – Manufacturing Technology*, 2005, 54(2):139–154.

[49] Hopp, W. and Spearman, M. Factory physics foundations of manufacturing management. McGraw Hill, 1996. (ISBN: 9780256154641).

[50] Hosoya, H., Frisch, A., and Castagna, G. Parametric polymorphism for XML. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '05. Location: Long Beach, California, USA, 2005, pages 50–62, New York, NY, USA. ACM. (DOI: 10.1145/1040305.1040310), (ISBN: 1-58113-830-X).

[51] Hosoya, H. and Pierce, B. C. XDuce: A statically typed XML processing language. *ACM Trans. Internet Technol.*, May 2003, 3(2):117–148. (DOI: 10.1145/767193.767195).

[52] Igarashi, A. and Pierce, B. C. Foundations for virtual types. *Information and Computation*, 2002, 175(1):34 – 49. (DOI: 10.1006/inco.2001.2942). (link)

[53] Jacobs, F. R. and Jr., F. T. W. Enterprise resource planning (ERP) – brief history. *Journal of Operations Management*, 2007, 25(2):357 – 363. (DOI: 10.1016/j.jom.2006.11.005). Special Issue Evolution of the Field of Operations Management SI/ Special Issue Organisation Theory and Supply Chain Management

[54] Jeong, B., Lee, D., Cho, H., and Lee, J. A novel method for measuring semantic similarity for XML schema matching. *Expert Syst. Appl.*, April 2008, 34(3):1651–1658. (DOI: 10.1016/j.eswa.2007.01.025).

[55] Kaes, S. Type inference in the presence of overloading, subtyping and recursive types. In *Proceedings of the 1992 ACM conference on LISP and functional programming*, LFP '92. Location: San Francisco, California, United States, 1992, pages 193–204, New York, NY, USA. ACM. (DOI: 10.1145/141471.141540), (ISBN: 0-89791-481-3).

[56] Kagermann, H. and Wahlster, W. Recommendations for implementing the strategic initiative Industrie 4.0. acatech, National Academy of Science and Technology, Germany, 2013. (pdf)

[57] Kfoury, A. J., Tiuryn, J., and Urzyczyn, P. Type reconstruction in the presence of polymorphic recursion. *ACM Trans. Program. Lang. Syst.*, April 1993, 15(2):290–311. (DOI: 10.1145/169701.169687).

[58] Kim, K.-D. and Kumar, P. R. Cyber–physical systems: A perspective at the centennial. *Proceedings of the IEEE*, 2012, 100(Special Centennial Issue):1287–1308. (DOI: 10.1109/JPROC.2012.2189792).

[59] Lee, J., Bagheri, B., and Kao, H.-A. Recent advances and trends of cyber-physical systems and big data analytics in industrial informatics. ResearchGate, 2014. Keynote given at the 12th IEEE International Conference on Industrial Informatics (INDIN 2014), Porto Alegre, Brazil

[60] Lee, J., Bagheri, B., and Kao, H.-A. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, 2015, 3(0):18 – 23. (DOI: 10.1016/j.mfglet.2014.12.001).

[61] Lee, J., Lapira, E., Bagheri, B., and an Kao, H. Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters*, 2013, 1(1):38 – 41. (DOI: 10.1016/j.mfglet.2013.09.005).

[62] Li, X. and Wang, Q. Coordination mechanisms of supply chain systems. *European Journal of Operational Research*, 2007, 179(1):1–16. (DOI: 10.1016/j.ejor.2006.06.023).

[63] Liberty, J. and Betts, P. Programming reactive extensions and linq. Apress, 1st edition, 2011. (ISBN: 978-1-4302-3747-1).

[64] Madhavan, J., Bernstein, P. A., and Rahm, E. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01. pages 49–58, 2001, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. (ISBN: 1-55860-804-4).

[65] Maropoulos, P. G., Kotsialos, A., and Bramall, D. G. A theoretical framework for the integration of resource aware planning with

logistics for the dynamic validation of aggregate plans within a production network. *CIRP Annals – Manufacturing Technology*, 2006, 55(1):483–488. ID number: ISI:000240073900112

[66] McGowan, M. K. *Electronic Data Interchange (EDI)*, pages 860–868. John Wiley & Sons, Inc., 2007, (DOI: 10.1002/9781118256107.ch55), (ISBN: 9781118256107).

[67] McKay, K. N. and Wiers, V. C. Integrated decision support for planning, scheduling, and dispatching tasks in a focused factory. *Computers in Industry*, 2003, 50(1):5–14. (DOI: 10.1016/S0166-3615(02)00146-X).

[68] Meijer, E. The world according to LINQ. *Queue*, August 2011, 9(8):60:60–60:72. (DOI: 10.1145/2016036.2024658).

[69] Meijer, E. and Shields, M. XMlambda - a functional language for constructing and manipulating XML documents. 2000. (pdf)

[70] Milner, R. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 1978, 17:348–375.

[71] Monostori, L., Váncza, J., and Kumara, S. Agent-based systems for manufacturing. *CIRP Annals - Manufacturing Technology*, 2006, 55(2):697–720. (DOI: 10.1016/j.cirp.2006.10.004).

[72] Monostori, L. Cyber-physical production systems: roots from manufacturing science and technology. *AT-AUTOMATISIERUNGSTECHNIK*, 2015, 63(10):766–776. (DOI: 10.1515/auto-2015-0066).

[73] Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W., and Ueda, K. Cyber-physical systems in manufacturing. *CIRP Annals – Manufacturing Technology*, 2016, 65(2):621–641. (DOI: 10.1016/j.cirp.2016.06.005).

[74] Monostori, L., Kis, T., Váncza, J., Kádár, B., and Erdős, G. Real-time, cooperative enterprises for customised mass production. *International Journal of Computer Integrated Manufacturing*, 2009, 22(1):55–68.

[75] Mula, J., Poler, R., García-Sabater, J., and Lario, F. Models for production planning under uncertainty: A review. *International Journal of Production Economics*, 2006, 103(1):271–285. (DOI: 10.1016/j.ijpe.2005.09.001).

[76] National Instruments. Labview system design software. Official website, 2012. (link)

[77] OASIS / RELAX NG Technical Committeee. RELAX NG. Official reference page, 2012. (link)

[78] Odersky, M., Sulzmann, M., and Wehr, M. Type inference with constrained types. *Theory and practice of object systems*, 1999, 5(LAMP-ARTICLE-1999-001):35.

[79] Oracle Technology Network. Oracle Technology Network, Java developers' index page. Official website, 2012. (link)

[80] Palsberg, J. Efficient inference of object types. *Inf. Comput.*, December 1995, 123(2):198–209. (DOI: 10.1006/inco.1995.1168).

[81] Palsberg, J., Wand, M., and O'Keefe, P. Type inference with non-structural subtyping. *Formal Aspects of Computing*, 1997, 9:9–49.

[82] Palsberg, J. and Zhao, T. Type inference for record concatenation and subtyping, 2003. (pdf)

[83] Pochet, Y. and Wolsey, L. A. Production planning by mixed integer programming. Springer Science & Business Media, 2006.

[84] Pottier, F. A framework for type inference with subtyping. In *Proceedings of the third ACM SIGPLAN international conference on Functional programming*, ICFP '98. Location: Baltimore, Maryland, United States, 1998, pages 228–238, New York, NY, USA. ACM. (DOI: 10.1145/289423.289448), (ISBN: 1-58113-024-4).

[85] Pottier, F. Type Inference in the Presence of Subtyping: from Theory to Practice. INRIA, 1998. (pdf)

[86] Schuh, G., Kampker, A., Narr, C., Potente, T., and Attig, P. myOpenFactory. *International Journal of Computer Integrated Manufacturing*, 2008, 21(2):215–221. (DOI: 10.1080/09511920701607766).

[87] Schuh, G., Gottschalk, S., and Höhne, T. High resolution production management. *CIRP Annals – Manufacturing Technology*, 2007, 56(1):439–442. (DOI: 10.1016/j.cirp.2007.05.105).

[88] Shin, D.-H., He, S., and Zhang, J. Robust, secure, and cost-effective design for cyber-physical systems. *IEEE Intelligent Systems*, 2014, (1):66–69. (DOI: 10.1109/MIS.2014.9).

[89] Simonet, V. and Rocquencourt, I. Type inference with structural subtyping: A faithful formalization of an efficient constraint solver, 2003. (pdf)

[90] Stadtler, H. Supply chain management and advanced planning – basics, overview and challenges. *European Journal of Operational Research*, 2005, 163(3):575–588. (DOI: 10.1016/j.ejor.2004.03.001).

[91] Tao, F., Zhang, L., Venkatesh, V., Luo, Y., and Cheng, Y. Cloud manufacturing: a computing and service-oriented manufacturing model. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 2011, page 0954405411405575.

[92] Toub, S. Introduction to tpl dataflow. Microsoft online document, 2011. (link, last accessed: April 2011)

[93] Traytel, D., Berghofer, S., and Nipkow, T. Extending hindley-milner type inference with coercive structural subtyping. In: Yang, H. (editor), *APLAS*, volume 7078 of *Lecture Notes in Computer Science*. pages 89–104, 2011. Springer. (ISBN: 978-3-642-25317-1).

[94] Upton, D. M. and McAfee, A. P. A path-based approach to information technology in manufacturing. *International Journal of Manufacturing Technology and Management*, 2000, 1(1):59–78. (DOI: 10.1504/IJMTM.2000.001334).

[95] Váncza, J. and Egri, P. Coordinating supply networks in customized mass production: A contract-based approach. *CIRP Annals - Manufacturing Technology*, 2006, 55(1):489 – 492. (DOI: 10.1016/S0007-8506(07)60465-X).

[96] Váncza, J., Egri, P., and Monostori, L. A coordination mechanism for rolling horizon planning in supply networks. *CIRP Annals - Manufacturing Technology*, 2008, 57(1):455 – 458. (DOI: 10.1016/j.cirp.2008.03.105).

[97] Wang, X. V. and Xu, X. W. An interoperable solution for cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 2013, 29(4):232 – 247. (DOI: 10.1016/j.rcim.2013.01.005).

[98] World Wide Web Consortium. Document type definition (dtd). W3C recommendation reference page, 1999. (link)

[99] World Wide Web Consortium. XML path language (XPath) version 1.0. W3C recommendation reference page, 1999. (link)

[100] World Wide Web Consortium. XSL transformations (XSLT) version 1.0. W3C recommendation reference page, 1999. (link)

[101] World Wide Web Consortium. SOAP version 1.2. W3C recommendation reference page, 2007. (link)

[102] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (fifth edition). W3C recommendation reference page, 2008. (link)

[103] World Wide Web Consortium. XML schema version 1.1. W3C recommendation reference page, 2010. (link)

[104] World Wide Web Consortium. XQuery 1.0: An XML query language (second edition). W3C recommendation reference page, 2010. (link)

[105] Wu, D., Rosen, D. W., and Schaefer, D. Cloud-based design and manufacturing: Status and promise. In: Schaefer, D. (editor), *Cloud-Based Design and Manufacturing (CBDM)*. Springer International Publishing, 2014, pages 1–24. (DOI: 10.1007/978-3-319-07398-9_1), (ISBN: 978-3-319-07397-2).

[106] Wu, D., Rosen, D. W., Wang, L., and Schaefer, D. Cloud-based manufacturing: Old wine in new bottles? *Procedia CIRP*, 2014, 17(0):94 – 99. (DOI: 10.1016/j.procir.2014.01.035).

[107] Wu, D., Rosen, D. W., Wang, L., and Schaefer, D. Cloud-based design and manufacturing: A new paradigm in digital manufacturing and design innovation. *Computer-Aided Design*, 2015, 59(0):1 – 14. (DOI: 10.1016/j.cad.2014.07.006).