High resolution and transparent production informatics

Key enabling information technologies for supplier collaboration management systems

PhD Thesis Booklet

Dávid Karnok

Supervisor: László Monostori, academician

Faculty of Mechanical Engineering,
Budapest University of Technology and Economics

Pattantyús Ábrahám Géza Doctoral School,
Material and Production Sciences Program

Institute for Computer Science and Control (SZTAKI),
Hungarian Academy of Sciences (MTA)

Budapest, Hungary, 2017

1 Introduction and motivation

Today, we are at the brink of a new industrial revolution process often referred to as **Industry 4.0** [37] and **Cyber-Physical Production Systems** (CPPS) [50]. 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 [49].

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** [63] 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 [37] 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[1] 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 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).

In the past 10 years, I have been working on various academia–industrial research and development projects and many different companies involved in these

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

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 [37, 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) [39] 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.

2 New scientific results

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

2.1.1 Introduction and state of the art

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.

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.

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 [59]. 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 [31, 65]. 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.

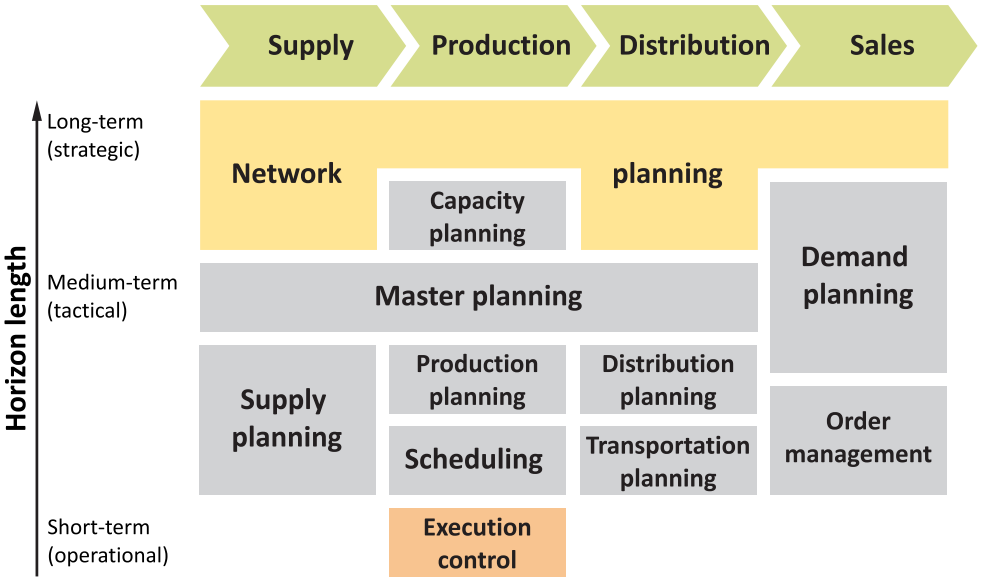

Figure 1: Matrix of strategic, tactical and operational planning functions [4] p. 298 after [31].

Within an enterprise, the coordination of segregated planning modules is a grave problem in itself [31, 45]. 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 [43]. 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.

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 [51, 68].

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 ap-

proach 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 [21]. 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 [23].

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 [29, 68].

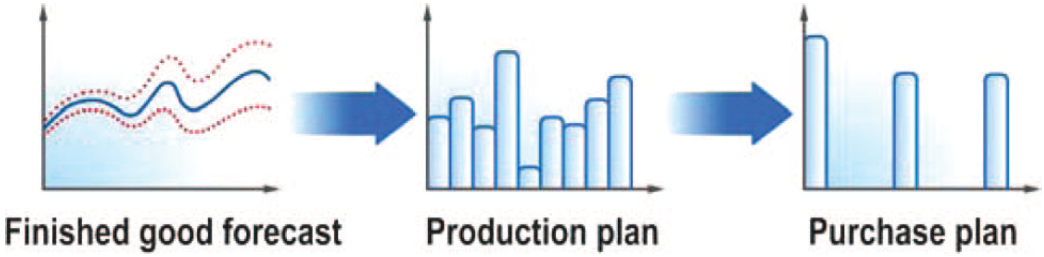

Figure 2: Transition of demand forecasts [4] p. 299.

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[2]. It uses Electronic Data Interchange (EDI) as a basic format but also supports WebEDI, which technically requires only Internet access [44]. A similar approach called myOpenFactory [62] proposes a centralized information sharing agency. 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 [40], the following general roadmap was suggested for such studies [51, 68, 69].

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

2.1.2 Coordination via a novel information sharing platform in logistics

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 when and how many of the material components the respective suppliers are going to deliver while keeping the information private from the other suppliers.

Thesis 1: The information exchange between the focal customer and its suppliers 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 uploading, 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.

In addition, 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 in accordance with user roles (such as a demand-fulfilling policy of the consumer versus detailed delivery schedule of the supplier).

Related primary publications: [1], [2], [4]
Related additional publications: [5], [6], [7], [9], [11], [12], [14], [13], [17], [18], [20]

2.1.3 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 3 and 4).

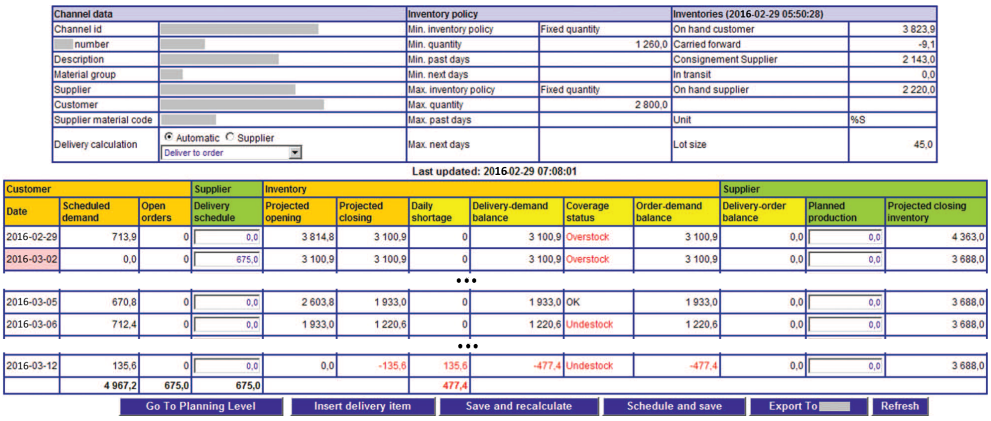

Figure 3: Detailed scheduling level data of a channel.

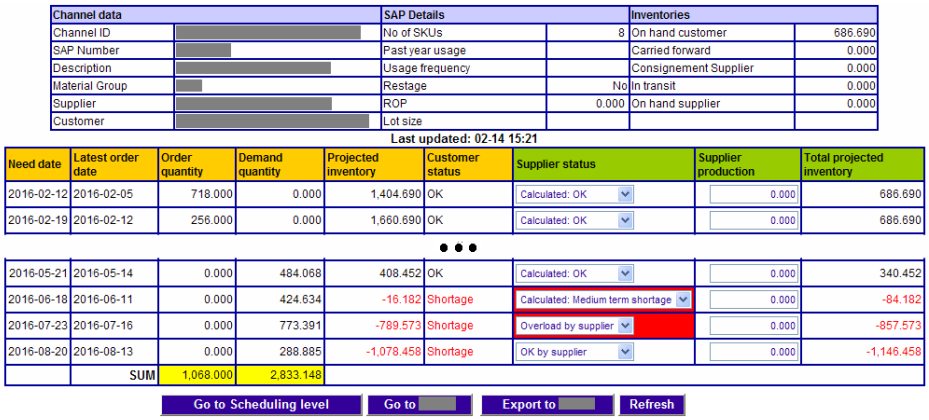

Figure 4: Detailed planning level data of a channel.

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 introduction of the Logistics Platform has resulted in an increased service level[3] over a broad range of material components between the focal manufacturing plant and its suppliers. Figure 5 shows a typical component's service level before and after the introduction of the platform:

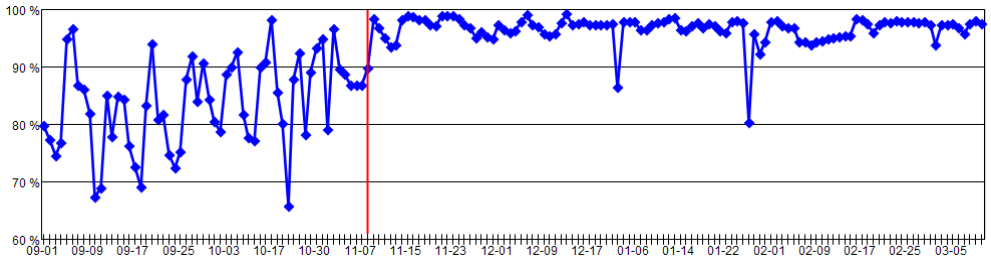

Figure 5: Service level of a component before and after the Logistics Platform went live (indicated by the red time marker).

[3]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.

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

2.2.1 Introduction and state of the art

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 [25].

- 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.

Describing data via XML, introduced by the World Wide Web Consortium (W3C) [74], 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 developed over the years to handle the task. Most known tools used with XML are the query languages XQuery [76] and XPath [71], as well as Extensible Stylesheet Transformation (XSLT) [72] 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) [70]. 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) [75]. 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 [53] 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 [33] 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 [47] 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, [73]) 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 [32].

XML processing languages may benefit from schema reduction and canonization. Creating an XML Normal Form similar to relational database normal forms is one possibility [22]. Alternatively, one can define a common structure

for representing schema definition elements [27].

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 [35] which uses supervised learning. Another example is the Cupid generic schema matching tool [42], 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.

2.2.2 Modeling and reasoning about data types

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.

Thesis 2: Given the XML schemas of the datasources and -consumers in a data interoperation scenario, 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 (comparison) such as equivalence, incompatibility or if one encompasses the other;**
2. **generate the canonical representation of the common superset (union); and**
3. **generate the canonical representation of the common subset (intersection) 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, subset or superset) 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]

2.2.3 Application of the results

The main application of Theses 2, 3 and 4 was the **ADVANCE Flow Engine** where ADVANCE[4] 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 6 gives a brief overview of the flow of parcels and associated information in this type of logistics network.

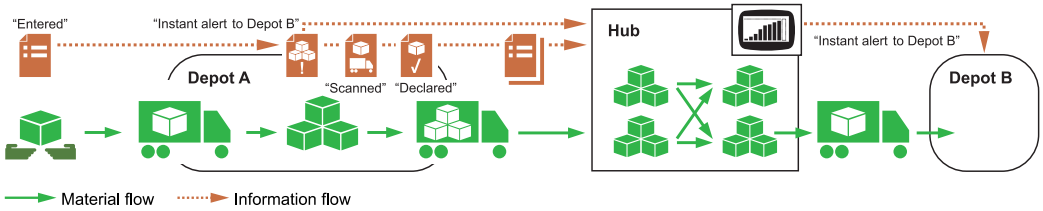

Figure 6: Typical material- and information-flow in a hub-and-spoke logistics network.

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

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,
- 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`[5] 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 7).

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

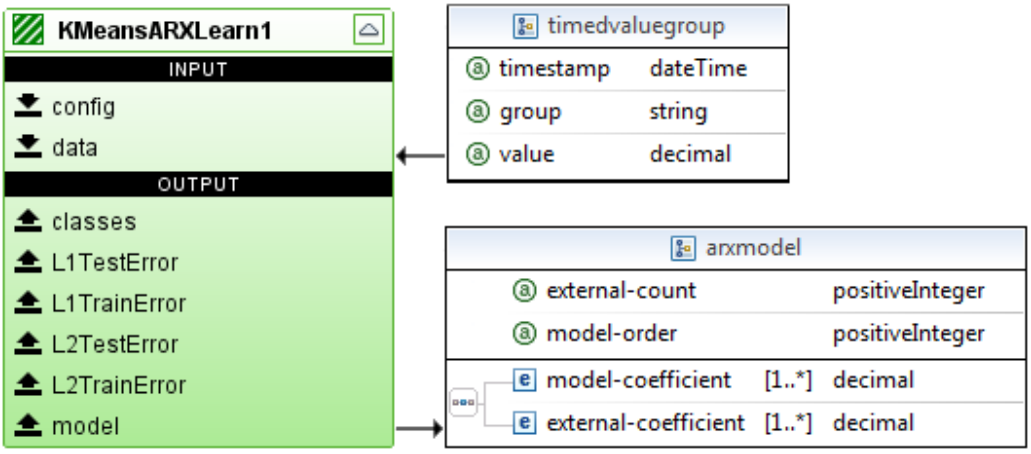

Figure 7: Example block with custom (and composite) input and output type structure.

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 8:

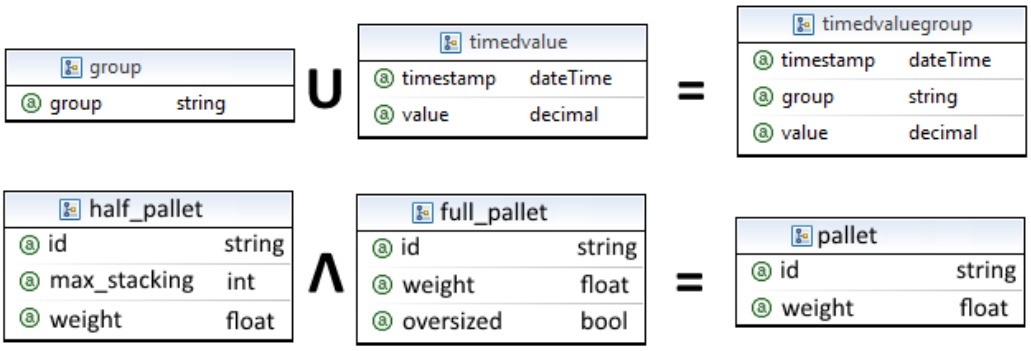

Figure 8: Generating union and intersection types from different data types.

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.

2.3 Validating and inferring data types in information exchange networks

2.3.1 Introduction and state of the art

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[6].

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 [48] 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

[6] 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.

bounds of parametric types they have.

Type inference is a well-studied field of programming languages, types and computer science in general. Starting from the well-known work of Milner [48], 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 [56]. Type systems with many flavors are considered in the academic literature: non-structural subtyping [57, 60, 61]; structural subtyping [58, 64]; virtual types [34]; coercions (automatic type conversion) [67]; constrained types [54]; recursion [36, 38]; and polymorphic types [38] 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.

2.3.2 Novel data type inference and validation over a pluggable type system

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.

Thesis 3: 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 if exact datatypes form a covariant relation;**
b. **verifies if 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]

2.3.3 Application of the results

As part of the ADVANCE Flow Engine mentioned in section 2.2.3, 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 9).

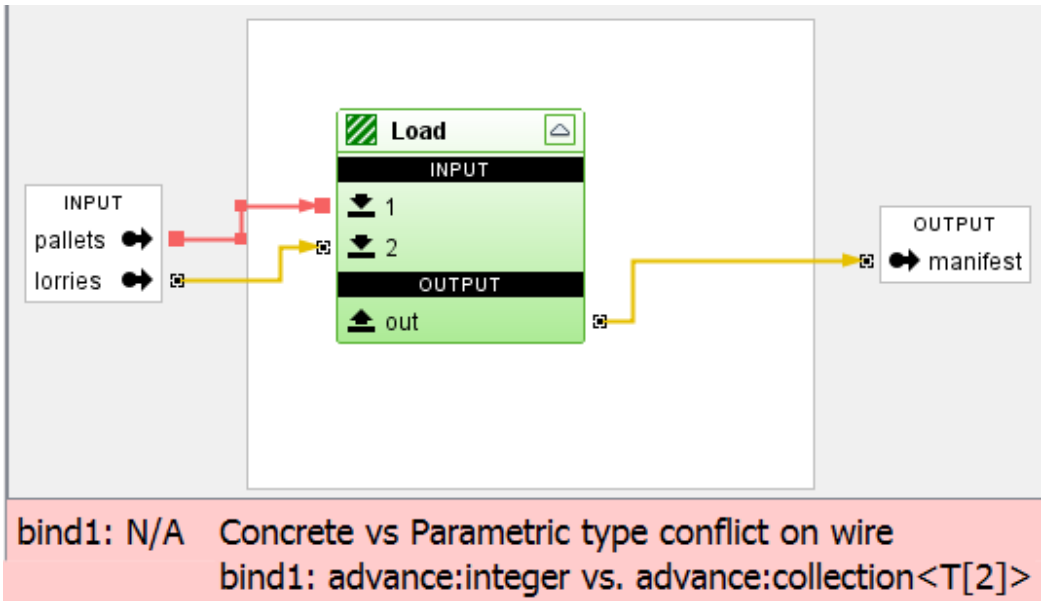

Figure 9: Type incompatibility when combining different sources.

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 9 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 10 shows a case when all source and output data types are correctly matched and wired together:

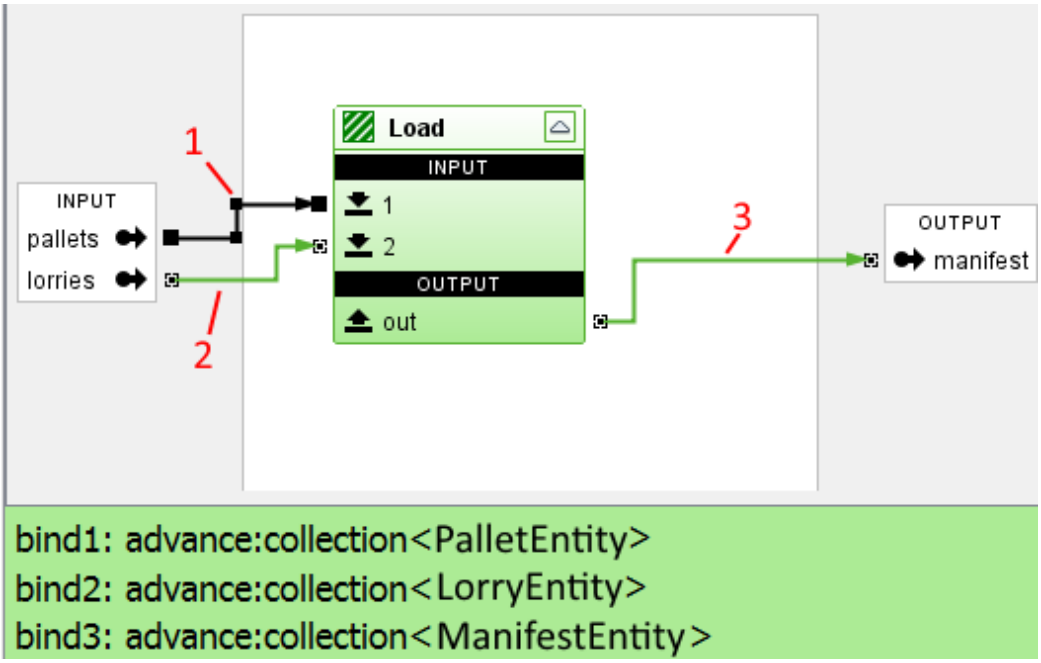

Figure 10: Data sources correctly wired and the inferred data types.

2.4 Driving information exchange and processing via reactive dataflows

2.4.1 Introduction and state of the art

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 [30], 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**[7].

Reactive or push-based programming [30] 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# [28] library is Reactive Extensions [41] with its ability to express and compose push-based operations with the same Language Integrated Query (LINQ) [46] that is generally used for pull-based data processing. In contrast, one of the most widely used programming languages, Java [55], does not provide much support for push-based operations per se – a possible workaround would be, e.g., the re-use of FlumeJava [24] or Frappé [26], originally meant to support parallel processing. This, however, 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 lan-

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

guage 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 [52] programming environment, and a recent development, the Task Parallel Library Dataflow Framework (TPL Dataflow, [66]). 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.

2.4.2 Novel framework for driving information flow and processing

Creating and executing a 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 simple, declarative dataflow-description and then transformed (compiled) into an efficient, machine-executable format.

Thesis 4: A dataflow network should be modeled in a novel, XML-based declarative format consisting of only four element types: constants, processing blocks, input and output ports, and wires connecting them. Given the canonical datatype definitions involved in the dataflow network, the results of the validation as well as the exact datatypes inferred from the dataflow graph of the network, a given dataflow-description XML should transformed via a novel algorithm into a runtime representation with novel composable, reactive and adaptively push- or pull-based computation and execution model. The runtime environment supporting this execution model should be responsible for binding the input and output ports of these blocks according to the dataflow graph, drive the data- and event delivery between them when the data- and event-processing is in operation and ensure the computational resources are utilized by the blocks in an efficient manner.

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],

2.4.3 Application of the results

As part of the ADVANCE Flow Engine mentioned in section 2.2.3, 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 11).

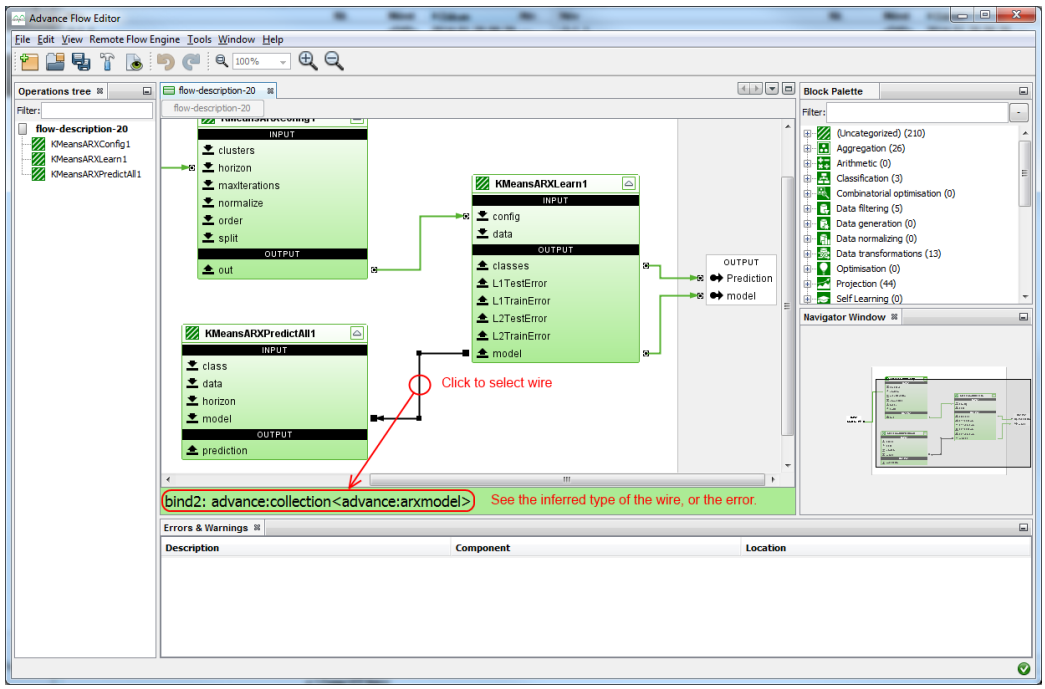

Figure 11: The graphical dataflow editor of the ADVANCE Flow Engine.

A Flow Engine instance hosts a set of so-called realms where an individual flow network runs and processes data (Figure 12).

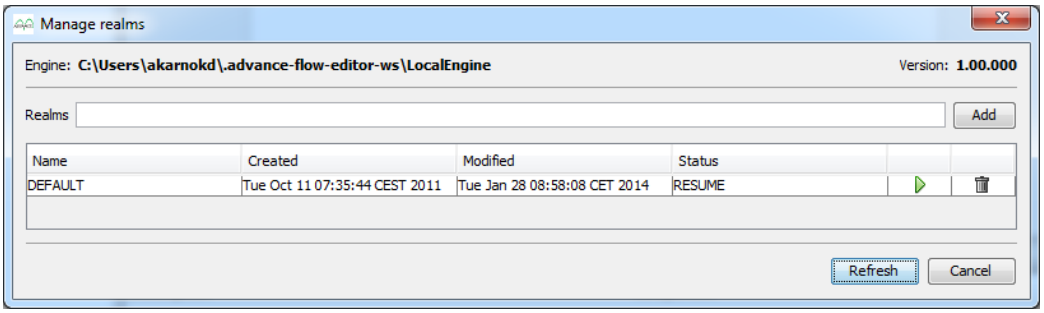

Figure 12: Execution realms of a Flow Engine instance.

Once the network has been designed, it can be uploaded into one of the realms and begin its execution (Figure 13):

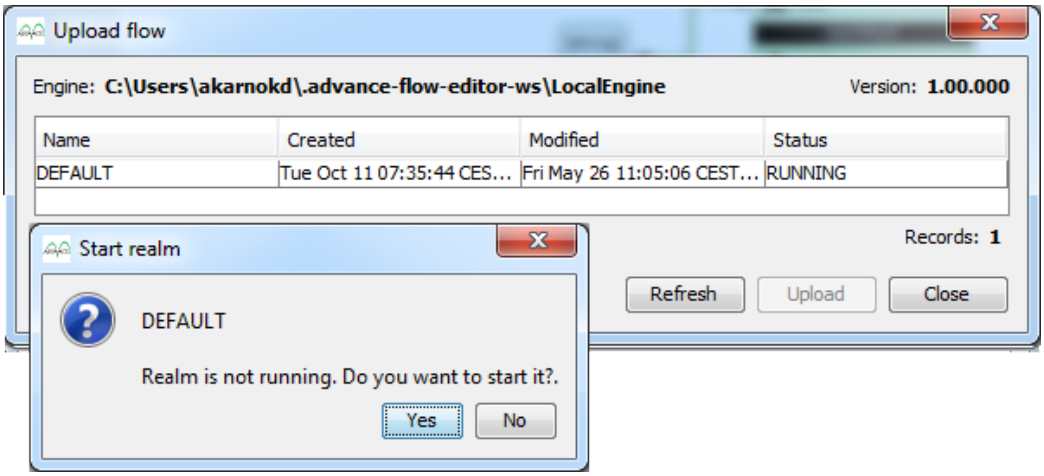

Figure 13: Upload a flow and start execution in a Flow Engine realm selected.

Inspecting and debugging flows are supported by the Flow Engine as well (Figure 14). 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.

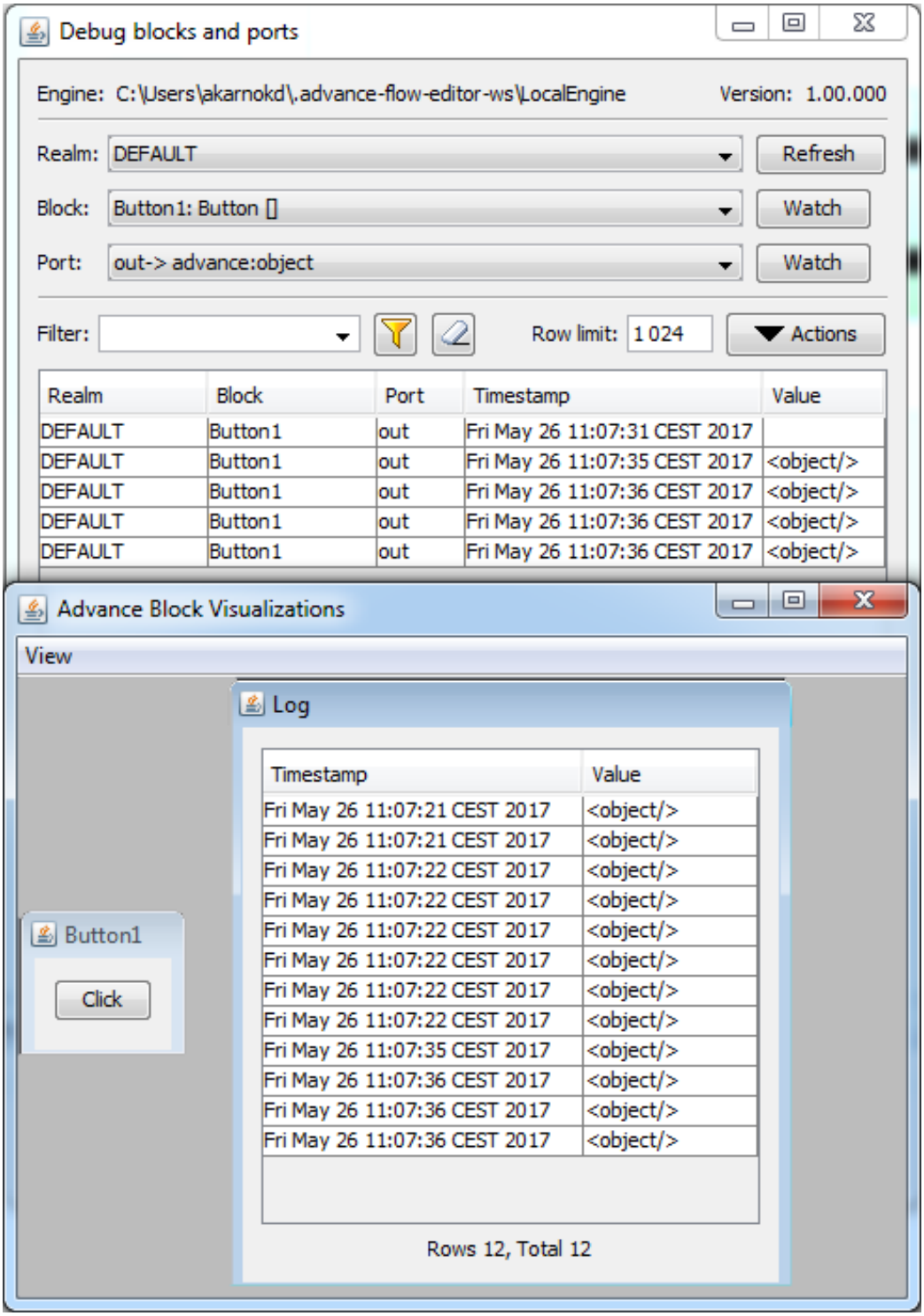

Figure 14: Displaying (debugging) the event flow in a Flow Engine Realm of a simple button-log dataflow network.

3 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[8], of which the present author contributed about 80% of the code, documentation and the support time combined[9]. 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 underlying libraries and frameworks to optimize the resulting dataflows to achieve better (Cloud) resource utilization as well. The success of RxJava (mostly in the form of disseminating and proving the usefulness of the reactive paradigm) resulted in an industry standard specification for in-memory dataflows called Reactive-Streams now expected to

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

[9] 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.

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[10] 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 on-screen dataflow network. (Although the author contributed significant amount of code to their underlying reactive library, Reactor-Core[11], 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 (com-

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

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

putational) 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[12]. Consequently, the author identifies the following problems/domains requiring further research:

- Operator-fusion: executing reactive flows even more efficiently by exploiting sequential stages[13].
- Parallel-flows: better utilization of multi-core hardware[13].
- 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.
- 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.

4 Acknowledgements

This research has been supported by the Hungarian Scientific Research Fund (OTKA), Grant No. 113038. and by the GINOP-2.3.2-15-2016-00002 grant on an "Industry 4.0 research and innovation center of excellence".

[12] See Reactive-Streams goals.

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

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

[22] 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).

[23] 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.

[24] 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).

[25] 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.

[26] 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.

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

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

[29] 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.

[30] 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).

[31] 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.

[32] 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).

[33] 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).

[34] 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)

[35] 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).

[36] 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).

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

[38] 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).

[39] 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).

[40] 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).

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

[42] 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).

[43] 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

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

[45] 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).

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

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

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

[49] 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).

[50] 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).

[51] 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.

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

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

[54] 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.

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

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

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

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

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

[60] 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).

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

[62] 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).

[63] 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).

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

[65] 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).

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

[67] 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).

[68] 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).

[69] 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).

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

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

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

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

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

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

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

Administrative information

Number of publications: **20**
Number of publications as primary author: **7**
Number of independent citations (as of November 2017): **39**
Total impact factor: **4.077**

Minimum requirement	**Actual**	**Minimum**	**percent**
Publications related to the Theses	20	4	500%
Foreign, peer-reviewed articles	241%	200%	121%
– Web of Science articles	125%	50%	250%
Publications in Hungarian	116%	100%	116%
International conference or any article	11	1	1100%