mathematics

Article

The Application of Directed Hyper-Graphs for Analysis of Models of Information Systems

Bálint Molnár *,† and András Benczúr *,†

Faculty of Informatics, Eötvös Loránd University, ELTE, Pázmány Péter 1/C, 1117 Budapest, Hungary
* Correspondence: molnarba@inf.elte.hu (B.M.); abenczur@inf.elte.hu (A.B.)
† These authors contributed equally to this work.

Abstract: Hyper-graphs offer the opportunity to formulate logical statements about their components, for example, using Horn clauses. Several models of Information Systems can be represented using hyper-graphs as the workflows, i.e., the business processes. During the modeling of Information Systems, many constraints should be maintained during the development process. The models of Information Systems are complex objects, for this reason, the analysis of algorithms and graph structures that can support the consistency and integrity of models is an essential issue. A set of interdependencies between models and components of architecture can be formulated by functional dependencies and can be investigated via algorithmic methods. Information Systems can be perceived as overarching documents that includes data collections; documents to be processed; and representations of business processes, activities, and services. Whe selecting and working out an appropriate method encoding of artifacts in Information Systems, the complex structure can be represented using hyper-graphs. This representation enables the application of various model-checking, verification, and validation tools that are based on formal approaches. This paper describes the proposed representations in different situations using hyper-graphs, moreover, the formal, algorithmic-based model-checking methods that are coupled with the representations. The model-checking methods are realized by algorithms that are grounded in graph-theoretical approaches and tailored to the specificity of hyper-graphs. Finally, the possible applications in a real-life enterprise environment are outlined.

Keywords: hyper-graph; information systems; enterprise architecture; horn clause; business process modeling; formal representation of processes

MSC: 68U35; 68M99; 05C65; 97M99; 68Q85

Citation: Molnár, B.; Benczúr, A. The Application of Directed Hyper-Graphs for Analysis of Models of Information Systems. *Mathematics* **2022**, *10*, 759. https://doi.org/10.3390/math10050759

Academic Editor: Frank Werner

Received: 31 January 2022
Accepted: 21 February 2022
Published: 27 February 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The notion of Information Systems looks back for decades; nonetheless, it is difficult to define, because of the complexity of such systems and their diverse application area. This is different from the General Systems Theory, which states

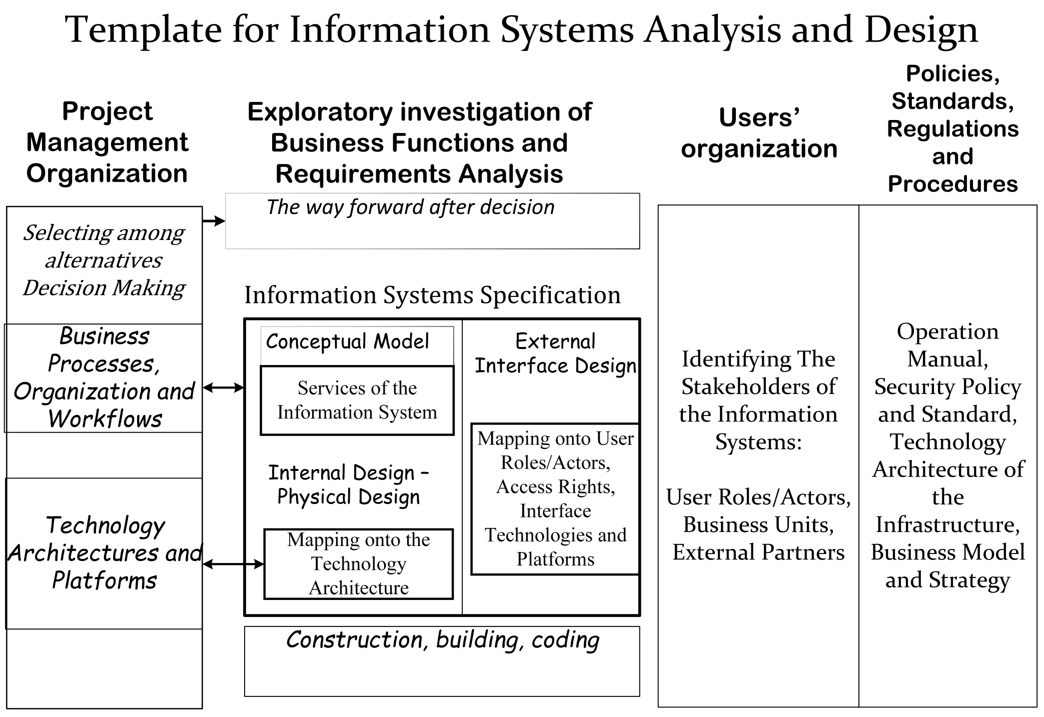
"A system can be defined as a complex of interacting elements." [1], p. 55.

A suitable definition is necessary for Information Systems in an enterprise environment. Based on the enormous relevant literature, we can conceptualize Information Systems as follows: Information Systems based on Information Technology operate in an organizational and human environment to achieve well-defined objectives through processing, storing, retrieving, disseminating, and transferring data to yield information for the end-users (cf. [2]).

An information system has various facets, aspects, perspectives, and views. We can investigate the services of Information Systems from the point of Service Science, i.e., what kind of services are provided by Information Systems [3,4]. Several types of Information Systems are typically embedded into the Service Systems of an Enterprise. In the Service Economy, Service Systems can be ameliorated through correct and accurate information management of Information Systems, which are the workhorses of Company Operation. Service Systems, and the underlying Information Systems, can be perceived as socio-technical systems because of their strong interactions with the human environment. Service innovation that is based on ICT (Information and Communication Technologies) requires a disciplined design approach between the carbon (human) and silicon (machine/computer) agents [5]. Designing a proper human–machine interaction is a challenge, as the major players can be cyber-physical systems, IoT (Internet of Things), sensors, actuators, edge computing, Cognitive Information Systems, or Decision Support Systems based on advanced Data Science [6]. To make the services that are yielded by Information Systems better, there is a need for an elaborated model for both the carbon and silicon agents. One of the issues that should be handled is the adaptive capability that the system should have to be prepared for prompts and unexpected stimuli from the human side. Service providing through Information Systems demands considering disciplines such as psychology, cognitive sciences and societal sciences [7,8]. An effective and efficient methodology for Information Systems Analysis and Design should take into account the issues of interactions between the carbon (human) and silicon (computational equipment). The discrepancies between the two sides can be partly handled by the notion of Cognitive Resonance [9,10]. The method for Information Systems Analysis and Design that wants to buttress Service Systems covers the broad spectrum that starts with cyber-physical systems and Edge Computing, through to Enterprise Resource Planning Systems, Decision Support, and Cognitive Information Systems incorporating the recent developments of Machine Learning, Computational Intelligence, and Data Science. Since various scientific disciplines play a role in socio-technical systems, the challenges can be surmounted by a common *mathematical language* that can be managed by Business Analysts, Systems Analysts and Designers, Data and Cognitive Scientists, and Systems Constructors and Implementers.

We can analyze the architecture of Information Systems—that provide services—by considering the architecture continuum of the generic solutions through to the sector-specific standards to their actual, particular implementation [11]. One dimension of the spectrum of Architecture Description deals with stakeholders, individuals who play important roles in the environment of Information Systems, e.g., Business Analysts, System Analysts/Designers, Software Developers, Implementers, and Operators that can have related well-defined views of the system, consisting of a set of models. The other dimension deals with the perspectives that embody the facets of an information system, namely data, process, placement, time/events, motivation/business rules, and people, i.e., the users of the system [12]. There is a schema which originated from the standards of database management systems that can describe the various facets of Information Systems (see Figure 1) [13–15].

This formal approach based on hyper-graphs can be applied to Enterprise Architectures and Information Systems (see Table 1). The model checking and analysis can be operationalized through appropriate executable languages at the XML level and using graph algorithms [16–19].



Template for Information Systems Analysis and Design
Project Management Organization
Exploratory investigation of Business Functions and Requirements Analysis
Users' organization
Policies, Standards, Regulations and Procedures
Selecting among alternatives Decision Making
The way forward after decision
Information Systems Specification
Business Processes, Organization and Workflows
Conceptual Model
Services of the Information System
External Interface Design
Mapping onto User Roles/Actors, Access Rights, Interface Technologies and Platforms
Internal Design – Physical Design
Technology Architectures and Platforms
Mapping onto the Technology Architecture
Construction, building, coding
Identifying The Stakeholders of the Information Systems:
User Roles/Actors, Business Units, External Partners
Operation Manual, Security Policy and Standard, Technology Architecture of the Infrastructure, Business Model and Strategy

Figure 1. A Conceptual Architecture. Template for Information Systems Analysis and Design.

Table 1. Representation of Enterprise Architecture by Hyper-graph.

Concept of Enterprise Architecture	Representation of Concept in the Domain of Hyper-Graph Theory
Information System (IS)	This consists of various models that depict the system from different aspects. The models and their constituents can be mapped onto a generalized hyper-graph to reflect the hierarchy of the models and their components.
A vertex in a hyper-graph	A vertex can represent a model element that can be characterized by constraints, pre- and post-conditions, and business rules formulated in logical statements.
Edge in a hyper-graph	An edge corresponds to a traditional edge in graphs, which connects two vertices in hyper-graphs. A simple edge designates the binary relationship between model components.
Hyperedge	A hyperedge represents a relationship among a specific group of vertices, e.g., models and their artifacts that are constituents of specific architecture layers.
System graph	This is a hyper-graph dedicated to describing the whole Information System; it contains hyperedges and their vertices, which represent models and their artifacts. It includes a disjointed node to denote the surrounding environment.
Sub-system	A specific module or well-defined part of an Information System. A set of hyperedges representing a sub-system composed of models and their artifacts belonging to this specific part of the system.

As we can see from the Zachman framework/ontology a universal language for modeling Information Systems does not exist. The models that are represented in the cells of the Table 2 are depicted in various field-specific languages [20]. However, the disparate modeling languages can be represented by the language of mathematics, especially graphs and related linguistic approaches. There are meta-modeling languages that support the classification, grouping, and analysis of the properties of modeling languages devoted to Information Systems [21]. In the representation through models of Information Systems, the Finite State Machines/Automaton occurs, especially in the time dimension taking events into account. However, the models of data collections and processes are strongly coupled to the actual state of the specific information systems (Figure 2, and the "why/motivation column" of Table 2). The "who column" of Table 2 primarily refers to organizations and their constituents; however, the contemporary Information Systems are more general, e.g., climate systems, autonomous systems/vehicles, or even chess or go games, etc. [22,23].

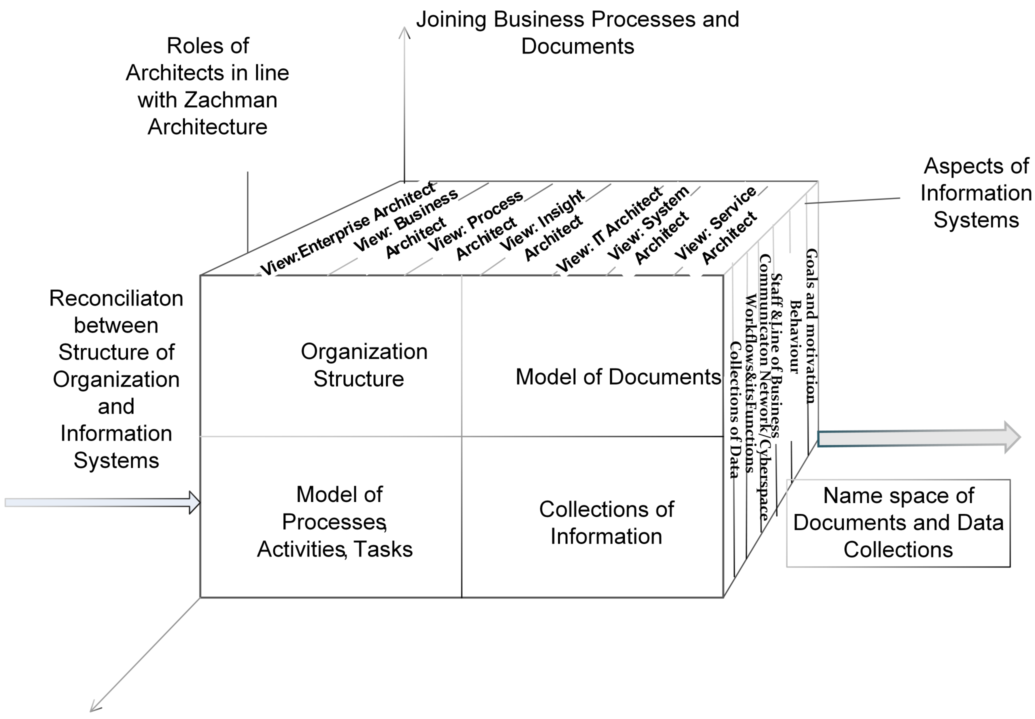

Joining Business Processes and Documents
Roles of Architects in line with Zachman Architecture
Aspects of Information Systems
View:Enterprise Architect
View: Business Architect
View: Process Architect
View: Insight Architect
View: IT Architect
View: System Architect
View: Service Architect
Goals and motivation
Behaviour
Staff &Line of Business
Communicaton Network/Cyberspace
Workflows&itsFunctions
Collections of Data
Reconciliaton between Structure of Organization and Information Systems
Organization Structure
Model of Documents
Model of Processes, Activities, Tasks
Collections of Information
Name space of Documents and Data Collections

Figure 2. Information System Architecture based on Zachman's and Blokdijk's approaches [12,24].

The columns of Table 2 try to answer the questions starting in English with "W" [25,26]. These six facets of Information Systems are important to create a consistent and integrated model, i.e.:

— *Who* are the stakeholders, end-users?
— What are the stimuli, events *when* the system is used?
— *What* are the entities, concepts, things that are specified in models within the system?
— *How* could the system be used by the members of the ecosystem?
— What is the goal of the system, motivation? *Why* the system is used?

Table 2. Zachman architecture and current Information Systems' model [12].

Aspects/Model Tier	What	How	Where	Who	When	Why	Viewpoints of Actors	Viewpoints of Roles
Extent of Business Domain	Facts, documents	Governance of Business Processes	Workflows for Business Processes	Organigram	Strategic Plan	Fundamental Objectives	Strategic Planner	Enterprise Architect
Notions of Business	Notional level data model of Data Collections (Directed Graph)	Process Model (Directed Graph)	Locations connected to Process Model (Graph)	Actor, Role coupled to Process Model (Graph)	Project Chart of Program (Directed Graph)	Association of Ideas and Objectives (Graph)	Business Analyst System Analyst	Business Architect and Process Architect
Logical Model of System	Logical Model for Data Collections (Directed Graph)	Logical model of Activities and Tasks (Directed Graph)	Logical Model of Components and Placement for Communication (Graph)	Actor, Role joined to Activity and Task Model (Graph)	Event and timing model (Directed Graph)	Business Rule (RDF/OWL Directed Graph)	System Designer	Insight Architect
Technology and Physical	Physical Data model (Graph)	Executable/ Interpretable Process/ Activity Model (Directed Graph)	Communication structure represented by deployed components (Graph)	Components for Access Rights and Roles (Directed Graph)	Choreography and Orchestration depicted by State Machines and Automaton (Directed Graph)	Executable/ Interpretable Rule Design (Directed Graph)	Program Designer and Developer	IT Architect
Assemblies of Constituents	Data in the physical implementation of DBMS	Code for Executable/ Interpretable Process/ Activity Model (Directed Graph)	Code for Communication structure represented by deployed components (Graph)	Code for Access Rights and Roles (Directed Graph)	Code for State Machines and Automaton (Directed Graph)	Code for Executable/ Interpretable Rule Design (Directed Graph)	System Builder and Implementer	System Architect
Functioning Enterprise	Data	Function	Network	Organization	Schedule	Strategy	Staff	Service Architect

The Enterprise Information Systems were considered to be a well-defined, stable structure regarding Business Processes. If the Business Process and their task could be organized in well-defined workflows, then their behaviors would be perceived as successful from the viewpoints of end-users—despite their high complexity, the behaviors of the systems could be kept in hand. Notwithstanding this, the Web Information Systems introduced a new factor, namely, that the interaction with users cannot be defined in detail in advance [27]. The Zachman architecture/ontology provides a description framework for grasping the static and dynamic facets of the system, along with the utilization of the system by the stakeholders. Apart from the traditional end-users and analysts, designers, new roles are currently emerging (see Figure 2) (cf. [28]). The various roles of architects, besides analyst and designer, convey an overarching view about the requirements of the organization that are formulated at the different architecture levels. The models within the architecture describe disparate aspects and serve distinct aims. The common goal is to yield a conceptual and technical view that effectively and systematically represents all models and architecture levels in a cohesive representation. The application of graphs as descriptive languages of models offers the opportunity to deal with the artifacts of models in a unified and uniform manner. A graphical representation allows us to depict the resources of the systems as interrelated components that were originally conceived in distinct formats [29] (see Figure 2).

The contribution of our paper is an apt encoding of the models, model elements, artifacts, and components of Information Systems. This makes it possible to represent an Information System in hyper-graphs in such a way that the graph-theoretic algorithms, in tandem with model-checking methods, can be applied to these representations to produce reasonable and useful analysis results. Furthermore, we showcase a transformation of the hyper-graph representing an Information System into simplicial complexes. This transformation enables the investigation of similarities and differences between existing and newly created business processes by exploiting matrix algebra and homology groups. Moreover, this transformation opens up the pathway to apply the tool-set of Data Science to these issues extensively and intensively. Our proposal differs from previous attempts, in that we take into account the heterogeneity of the components of an Information System. We can depict the heterogeneous component by exploiting the flexibility of the generalized hyper-graph by differentiating the properties of vertices and hyper-edges.

The structure is our paper is as follows: in Section 2 we provide an overview of the mathematical background, as well as definitions that are relevant to the representation of Information Systems. In Section 3, a qualitative literature review is provided about the related works. In Section 4, we present our models and representations of Information Systems in hyper-graphs and simplicial complexes; furthermore, the model-checking methods and algorithms that operationalize the methods in the digital universe are discussed. In Section 5, we discuss and compare our models and model-checking methods to other approaches and outline possible future research directions. In Section 7, we provide the accessible software code-base that was created in the projects that were related to the results presented in the paper. In Section 8, we close our paper by summarizing our results.

2. Hyper-Graph Representation of Information Systems

In this section, we overview the necessary theoretical, mathematical basis of hyper-graphs, then we showcase the proposed representation of Information Systems. The generalized hyper-graphs are apt tools to represent complex structures; nevertheless, the hyper-graphs are graphs, therefore the graph analysis tools that are grounded in mathematics, along with a set of algorithms out of Computer Science, are readily available for use in this area [30]. This approach can be considered to be a conceptual language for the description of Information Systems; however, the technical level of representation can be interpreted easily by the machine, and the description can be understood by humans too. As we have seen up to this point, both collections in Information Systems and their constituents (“things”, artifacts, entities, objects, relationships, etc.) have structure. What structure description and representation formal language can be used for analyzing Information Systems? We propose the hyper-graphs for descriptive purposes and the transformation of hyper-graphs into appropriate graphs for computationally effective and efficient handling. There have been various attempts to grasp phenomena, some examples of these are when the collection of "things" ("entities") and the "things" (entities) themselves have their internal structures, these examples include :

— **Assemblages** Sawyer et al. consider Information Systems as digital "assemblages" that are the interconnections among parties participating in the information exchange of business and information flows among institutions. The Information Technology underpins and yields models for these configurations [31–33]. Following this viewpoint, we may take into account the properties of both individuals and groups of some things.

— **Granules** In the first cut, we can think of granules as equivalence classes. However, other structures of granules are well-known, e.g., within Soft Computing [34]. A granule as an entity may have properties that induce the clustering of various individuals into specific clusters considered to be similar to granules. However, it is allowed that the same element may occur in disparate granules depending on the actual perspective. Following this approach, both granules and the elements of the granules may have their own specific properties.

— **Connections** There are several examples, e.g., most recently, social media where the avatars, the digital personas, and any defined communities possess a diverse set of properties. Similarly, the network of roads or rail tracks together with the gas and rail stations showcase similar structures. Thus, the links and essential entities of the system, the higher-level organization of elements, hold a distinct set of properties [35].

2.1. Graph Models of Systems

Generic graphs include the mathematical construction of networks that formulate various constraints and restrictions. The graphical representation means that the domain of discourse is represented by tuples and the relationships among them [36]. In the Information Systems field, we can typically encounter deterministic constraints that can be described by variables, logical statements depicting integrity constraints, consistency assumptions, and security conditions. Each single graph model has its own inquiries of the problem that are formulated as queries in a language apt to the specific context. The task is to find a solution in the form of assignments of variables to satisfy the queries and constraints. We can perceive a graph model as a set of functions whose arguments consist of a subset of variables. The variables transmit information about constraints, restrictions, and preferences in a deterministic manner.

We follow the convention of how to define a graph model: *A graph model* $\mathcal{G}_\mathcal{M} = [\boldsymbol{\Xi}, \boldsymbol{\Delta}, \boldsymbol{\Phi}, \bigoplus]$consists of a set of variables, domains, and functions.

$$\begin{aligned}
&\boldsymbol{\Xi} = \{\xi_1, \xi_2, ..., \xi_n\} \text{ variables,} \qquad (1)\\
&\boldsymbol{\Delta} = \{\delta_1, \delta_2, ..., \delta_n\} \text{ is the set of domains of data types that care for the values of variables,}\\
&\boldsymbol{\Phi} = \{\varphi_1, \varphi_2, ..., \varphi_k\} \text{ this designates the set of functions that may have a subset}\\
&\text{of variables as their arguments,}\\
&\textbf{Var\_Arg} = \{\boldsymbol{\sigma_1, \sigma_2, ..., \sigma_k}\}, \text{where } \{\xi_{i_1}, \xi_{i_2}, ..., \xi_{i_n}\} = \boldsymbol{\sigma_i} \subset \boldsymbol{\Xi} \text{ the subset of variables}\\
&\text{ that are input arguments to functions } \varphi_i,\\
&\text{the aggregation operator } \bigoplus \text{in case of logical statements and Horn clauses}\\
&\text{can be the Boolean functions } \bigoplus = \langle \wedge | \vee | \bigotimes \rangle,\\
&\text{in the case of functions with ranges in real or discrete domains } \bigoplus = \langle \prod | \sum | \bowtie | \ltimes | \rtimes \rangle.
\end{aligned}$$

The graph models contain various graphs as fundamental layers that represent the knowledge in the models. Typically, it depicts the dependencies and independencies among the disparate constituents of models, as well as the constraints and restrictions, with the help of variables and functions. An entire graphical model $\mathcal{G}\_\mathcal{M}$ can be perceived as an *overarching function* with the input arguments, variables, Ξ. The manifestation of the *overarching function* is the combination of all available functions of the graphical model $\mathcal{G}\_\mathcal{M}$ in the form of $\bigoplus_{i=1}^{n} \varphi_i$. The set of single functions determines the graphical model and implicates the behavior of the whole model. The overarching function yields the meaning of the entire graphical model. The calculation of the overarching function is computationally intensive and complex; nevertheless, it is tractable in the case of Information Systems [37,38]. All problems that can be raised can be formulated relative to the overarching function. For example, we look for a valuation of all variables in logical statements that a logical value is *true*. In other cases, we search for solutions that satisfy the constraints specified in the model.

Definition 1. *A primal graph (independence map) is an undirected graph that represents the variables as the vertices of the graph, and the edges designate that the connected vertices representing the variables belong to the same function [39–41].*

2.2. The Fundamentals of Hyper-Graphs

The various artifacts of models within the architecture are formulated in XML, JSON, or newer descriptive languages to represent documents [42,43]. These languages can be considered to be graphs that depict processes, tasks, workflows, events, organizations, and data collections. The generalized hyper-graph can be applied to represent these documents and artifacts.

The hyper-edges in a hyper-graph may contain any number of vertices.

Definition 2. *Let **(G (V, E)** be a hyper-graph, where **V** represents a finite set of vertices, and **E** stands for the set of hyperedges **e**.A hyperedge* $\mathbf{e} \in \mathbf{E}, \mathbf{e} \subset \mathbf{V}$ *is a subset of **V**.*

A directed hyper-edge or hyper-arc is an ordered pair, $E = (X, Y)$*, of (possibly empty) disjointed subsets of vertices; X is the tail of E, while Y is its head. The tail and the head of hyper-arc E can be denoted by* $T(E)$ *and* $H(E)$*, respectively, or alternatively a hyper-arc* $\overrightarrow{e}_i \in \overrightarrow{H} = (V; \overrightarrow{E} = \{\overrightarrow{e}_i | i \in I\})$ *can be perceived as an ordered pair* $\overrightarrow{e}_i = \left(\overrightarrow{e_i^+} = (e_i^+; i); \overrightarrow{e_i^-} = (e_i^-; i)\right)$*, where* $e_i^+ \subseteq V$ *is the set of vertices of* $\overrightarrow{e_i^+}$ *, and* $e_i^- \subseteq V$ *is the set of vertices* $\overrightarrow{e_i^-}$*. The elements of* $\overrightarrow{e_i^+}$ *(hyperedges and/or vertices) are called tail of* $\overrightarrow{e}_i$*, while the elements of* $\overrightarrow{e_i^-}$ *are called head* [44].

The incidence matrix for the directed hyper-graph $\overrightarrow{H}$ is a $n \times m$ matrix

$$a_{ij} = \begin{cases} -1, & \text{if } v_i \in T(e_j) = \overrightarrow{e_j^+}, \\ 1, & \text{if } v_i \in H(e_j) = \overrightarrow{e_j^-}. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Figure 3 depicts a directed hyper-graph that represents the document flow containing documents and the roles that are responsible for manipulating them. The direction of the flow and the relationship between the roles is described by the tails and heads of the hyper-arcs.

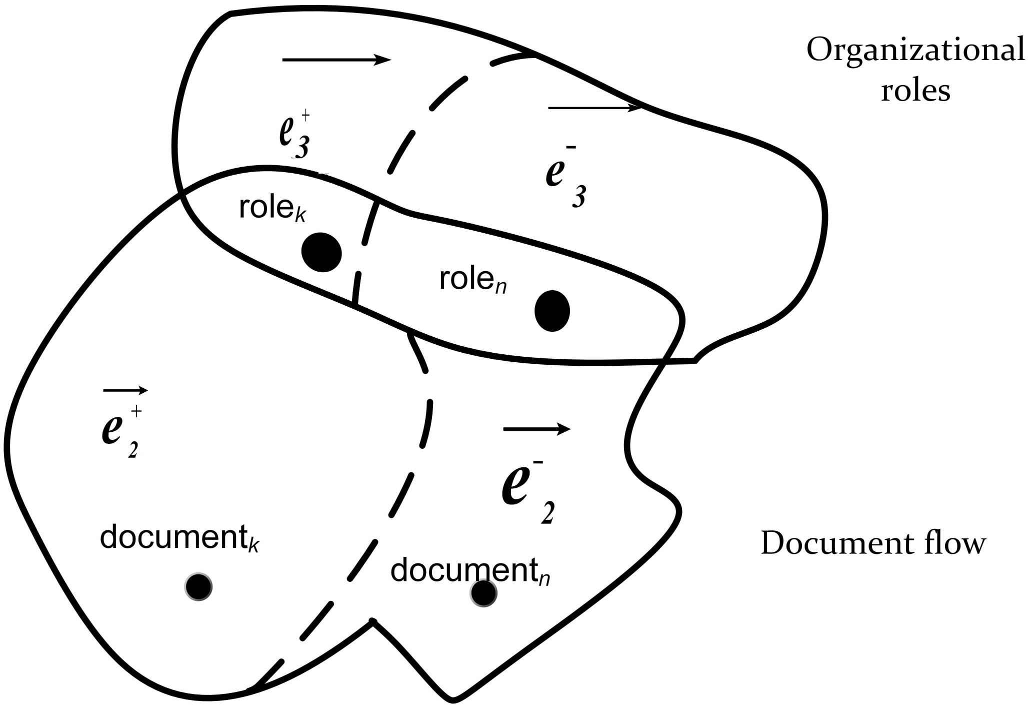


Figure 3. An example of s Directed Hyper-graph.

Definition 3. *A hyperedge,* **e** *, may also consist of both vertices and hyperedges in a* ***generalized*** *form.The hyperedge comprises other, distinct hyperedges, i.e., the hyperedge* **e** *that contains other hyperedges should be different from* **e** *[44].*

Figure 4 shows a simple, generalized hyper-graph as an illustration. Table 3 describes the incidence matrix that represents the hyper-graph in Figure 4.

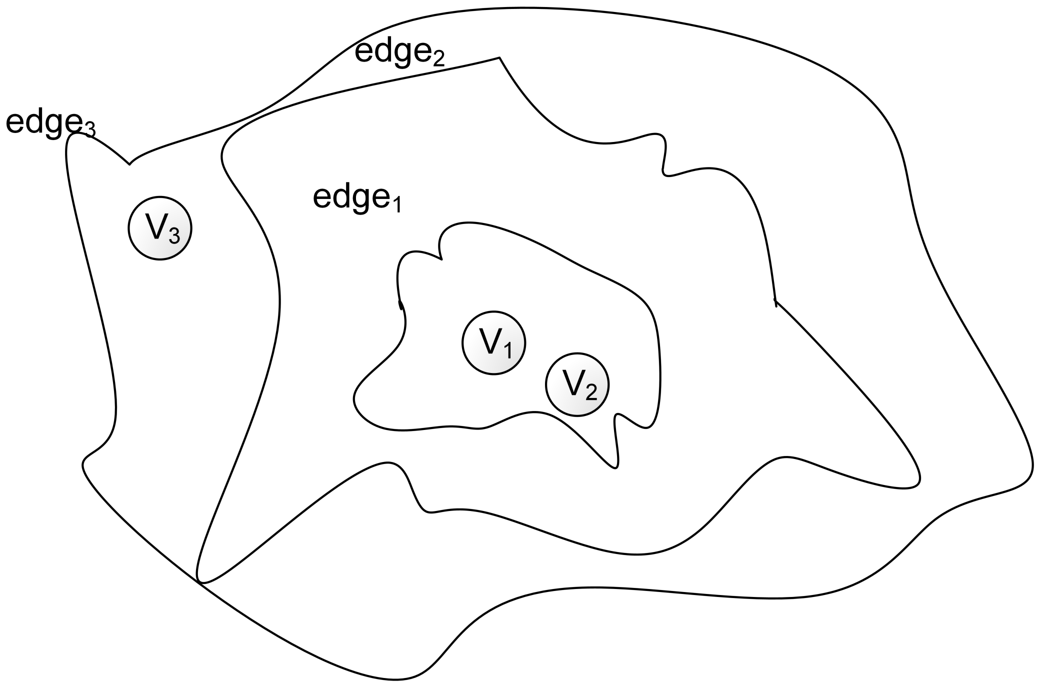


Figure 4. Representation of a Generalized Hyper-graph.

The incidence function of a hyper-graph is given by $H(V,E)\ \ inc : E \longrightarrow \wp(V)$. This definition allows for a number of edges to contain the same set of vertices, i.e., being incident, and any edge may contain an empty set of vertices. The incidence function can be represented by an incidence matrix.

Table 3. Incidence matrix for the generalized hyper-graph example.

	v_1	v_2	v_3	$edge_1$	$edge_2$	$edge_3$
$edge_1$	1	1	0	0	0	0
$edge_2$	0	1	1	1	0	0
$edge_3$	1	1	1	1	1	0

2.3. Operationalization and Implementation

There are a lot of approaches and formulations in mathematics about how graphs can be represented. There are many representational approaches, even for hyper-graphs. Model verification and validation and executable code generation impose certain constraints on an adequate faithful mapping. Therefore, if we want to achieve these goals, we need a mode of representation that allows us to represent hyper-graphs as conventional graphs. This approach to hyper-graph representation permits the hyper-graph to be stored in a common graph database without any loss of information (Figure 5). One of the best-known hyper-graph-to-graph mappings is when the hyper-graph is displayed as a bipartite graph.

Thus, $(G_h(V_h, E_h)$, a hyper-graph, can be represented by a bipartite graph, $G_{bip}(V \cup V', E_{G_{bip}}), \forall e_i \in E_h \longrightarrow v_{e_{i_{bip}}}$, i.e., for each hyper-edge, a corresponding vertex is ordered to in the bipartite hyper-graph, thereby a set of vertices is created: $V' = \{v_{e_{i_{bip}}} | e_i \in E_h\}$. This bipartite graph is denoted as being the incidence graph of the hyper-graph, $(G_h(V_h, E_h)$, and can be represented by an incidence matrix (see Table 4). If $x \in V \cup V'$ and $e \in E_h$, then they are adjacent in G_{bip} iff. $x \in E_h$ [44].

As such, the hyper-edges are represented as labeled vertices, and, when a hyper-edge contains a graph vertex or another hyper-edge, the vertices in the bipartite graph representing them are connected. Figure 6 shows how the same hyper-graph can be represented as a hyper-graph and bipartite graph [19,45–47].

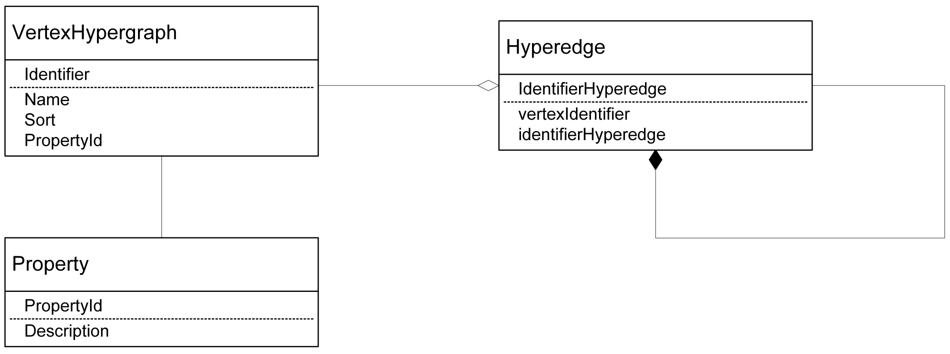


Figure 5. Representation at the implementation level of vertices in a hyper-graph.

Table 4. Incidence matrix for the bipartite graph example in Figure 6.

	v_1	v_2	v_3	v_4	v_5
$edge_1$	1	1	0	0	0
$edge_2$	1	1	1	0	0
$edge_3$	0	0	1	1	1

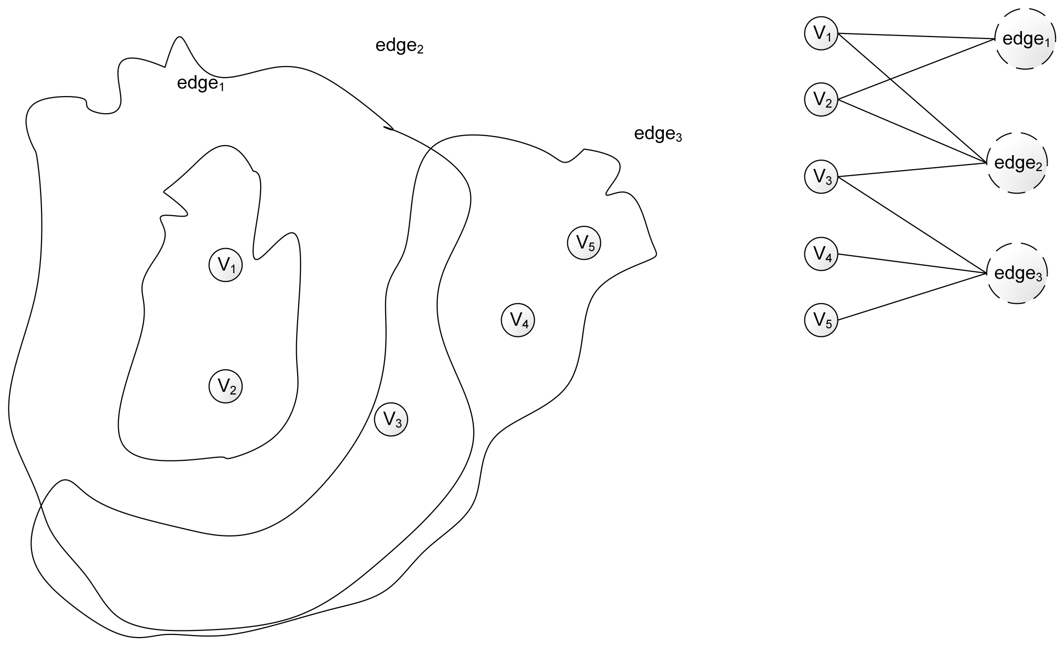


Figure 6. Bipartite graph.

2.4. Implementation Details in Graph Databases

Exploiting the hyper-graph mapping into a bipartite graph, the similarities between model elements can be analyzed. Once the hyper-graphs representing the model elements and design artifacts have been mapped to even graphs, it becomes possible to perform an analysis with the available algorithm set.

One possibility is to use Smith Normal Form to find similar models and design artifacts, and then to perform further analysis. The Smith Normal Form is a matrix representation that can be defined for any matrix over integers $\mathbb{Z}$. According to ring theory, the elements of the matrix that is to be transformed into Smith Normal Form can generally be in a principal ideal domain (PID), thus the statements about the Smith Normal Formal hold, but the ring of integers $\mathbb{Z}$ is sufficient in Computer Science applications [48,49], p. 479. The incidence matrix of a graph or a hyper-graph can be transformed into Smith Normal Form; thereby, the integer programming methods and algorithms can be exploited [50]. Thus, there were efforts to create a computationally feasible approach that could be utilized to discover hyper-graph features and their representations [51,52].

2.5. Hyper-Graph Representation by Simplicial Complexes

For further analysis, a hyper-graph can be mapped onto *a simplicial complex*. The basic principle for this transformation is that, if a hyper-edge is a part of a simplicial complex, then any subset of vertices belonging to the hyper-edge is a part of the simplicial complex. This property of simplicial complexes representing hyper-graphs provides the opportunity to study complex relationships among the elements of systems, i.e., systems that can contain a large number of elements with a high number of interconnections, but it also allows for low-order interrelationships. The notion of simplicial complexes allows for differentiating the underlying structures of hyper-graphs.

Definition 4. *An abstract simplicial complex* $ASC = (V, S)$ *is where* V is the set of vertices, $S = \{S_k \subseteq V | \overline{\quad S_k \neq \emptyset \quad \} \quad ,and \quad \forall\, i,j} \quad S_{k_i} \in S, \quad S_{k_j} \subset S_{k_i} \Rightarrow \quad S_{k_j} \in S.$ *Thus, a* σ k-simplex *is* $\sigma_k = \{v_0, v_1, \ldots, v_k\} \quad \forall i,j \quad v_i, v_j \in V, \quad v_i \neq v_j \quad \forall i,j \quad i \neq j$. *A face of* σ_k k-simplex is a *(k-1) simplex* $\sigma_{k-1} = \{v_0, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k\}$.

Thus, a S simplicial complex consists of non-empty subsets of V vertices, and it is closed under the *subset* set-algebraic operation. Simplicial complexes can be visualized in a geometric space with enough dimensions. For instance, a 0-dimension simplex is a vertex, a 1-dimension simplex is a line, a 2-dimension simplex is a triangle, a 3-dimension simplex is a tetrahedron, etc. The 1-dimensional simplicial complexes are networks or graphs; 2-dimensional simplicial complexes can describe the interrelationships among several vertices. Generally, a simplicial complex with dimension $d \geq 2$ can indicate interconnections among $d+1$ vertices. Thereby, simplicial complexes can depict complex networks of interactions represented by hyper-graphs. Thus, simplicial complexes can efficiently and effectively outline the interactions among any arbitrary number of "things". An abstract simplicial complex ASC comprises simplices in such a way that, if $\sigma \in ASC$, then all faces of σ belong to ASC.

2.6. Homology and Similarities

The other approach that has proved useful to describe hyper-graphs is the usage of simplicial complexes. Since we are interested in similarities and dissimilarities between the representations of models within Information Systems, the algebraic topology application seems apt to highlight invariants of the representation that could interest to pinpoint problems and phenomena.

Reduced homology groups are topological invariants, which means if two algebraic topological spaces are homeomorphic, i.e., homotopy equivalent, then their associated homology groups are isomorphic.

Definition 5. *A $C \subset G$ clique C of a graph $G = (V, E)$ exists if $\quad \forall v_i, v_j \in C, i, j = 1, \dots n$, $\{v_i, v_j\} \in E$. A clique C is a k-clique C if $|C| = k$.*

Definition 6. *A clique complex is a simplicial complex that is mapped from the cliques of a graph, i.e., $\forall \quad C \subset G, C \in \mathcal{C} = \{C \subset G | C \quad is k-simplex\ k = 1, \dots n\}$, onto $k-1$ simplices that are construed by the vertices of C. As a simple example, a three-clique transformed into a two-simplex (i.e., a full triangle).*

Definition 7. *Informal definition of Homology Groups: Homology Groups are defined mathematically and precisely in algebraic topology [53]. $H_k(ASC)$ is an algebraic group of an abstract $(k+1)$-dimension simplicial complex; it is a set of the equivalence classes of k-cycles. The k-cycles are linked to k-dimensional holes. For instance, the homology group $H_1(ASC)$ depicts 2-dimensional non-bounding cycles, i.e., a hole, it is not filled-in, that are bounded by one-dimensional cycles. $H_2(ASC)$ characterizes 3-dimensional holes that are bounded by 2-dimensional cycles, and so on [54–56].*

A homology group is a topological invariant that can describe a topological space by what kind of and how many dimensional holes it has, and, moreover, it determines the dimensions of holes, more exactly the boundary cycles that are non-bounding, i.e., within the boundary, there is a void. By a hole, we mean a part of a given abstract topological space that has a boundary but is not filled, i.e., the part within the boundary does not belong to the simplicial complex under consideration. The dimension of a hole is directly related to the dimension of its boundary. The boundary of a two-dimensional hole is a one-dimensional circle, a cycle; the 3-dimensional interior of a polyhedron is bounded by a 2-dimensional surface; etc. We start from a hyper-graph representation of information system models in hyper-graphs and map generalized hyper-graphs to simplicial complexes. In the case of models for Information Systems, the connections among the elements are the prime interesting feature. The defining metrics for models to apply algorithms of Data Science and Computational Intelligence do not seem an adequate approach, it is unnatural to define artificial distances, since the essential information is qualitative in such an abstract space that contains hyper-graphs and their simplicial complex representations. For that reason, the algebraic topology approach seems to fit the purpose of exploring the connections along with cycles and higher dimension constructions within the abstract space. Therefore, the application of algebraic topology methods, homologies, and related invariants seems a useful approach to investigate the models and their representations qualitatively. The properties of the relationships between vertices and edges in the case of hyper-graphs and the associated simplicial complexes are not dependent and not sensitive on the selected metrics. Thereby, the topological methods are appropriate when examining phenomena where the quantitative values of distance metrics are not relevant [57].

2.7. Modeling and Verification of Information Systems

The information system consists of three different aspects. The three facets are as follows: event (time), data/information, and functions. The integrity and consistency between the three aspects must be ensured by reconciling the dichotomies between the aspects by clarifying and reconciling the differences in the dichotomy and counterpoints between the aspects.

- **Information and Data:** The integrity and consistency in the Information and Data collection is vital for the operation of Information Systems. The Information and Data collection can be implemented by various structuring approaches, namely, relational and NoSQL databases, XML (Document Object Model), JSON, etc. [29,40,42,43,58–61]. The data collections can be manipulated by the Data Manipulation Language of relational calculus and SQL. The documents described in XML can be handled by XPATH and XQuery. The data stored in other NoSQL databases can be managed by the native and system-specific languages. The Data Definition Language in Relational Database Management Systems (RDBMS) and the same brethren in other Data Base Management Systems offer the possibility to define integrity constraints. The normalization that was originally defined for relational databases has been extended to XML and object-oriented data collections [62–65]. Thereby, the entity (entity $\leftrightarrow$ identity), referential, and business integrity can be interpretable for tuples of data collections and not only for relational databases.
- **Processes, Events and Entities**: Events are intimately coupled to Business Processes and Workflows. Activities in Business Processes and Workflow are initiated by either external or internal events. The chain of activities and the state changes exerted by the event-triggered activities can be semi-formally represented in Business Process Modeling Notation (BPMN), Event Process Chain (EPC), or in Petri nets [66–69]. The formal ground are the process algebra and finite-state machines [70–72], respectively. The descriptive, syntactic representation of Business Processes, Workflow, Petri Nets, and their activities is realized in XML nowadays [73]. The behavior of the Information System and its dynamic constituents is described by processes and activities, which are represented typically in XML; therefore, they can be grasped as documents. The events represented within diagrams of BPMN, EPC, Petri net, or UML Activity cause state-transition of entities, objects, or "things" in the data collections [74]. The entity-relationship diagrams, or object class diagrams, which describe the relationships among entities of data collections, are inherently hierarchic so that some entities are subsumed into other entities. There are two aspects of effects that are exerted by events on entities. One aspect is when the chain of events is tracked through the life cycle of an entity in the form of the Entity Life History or UML State Chart [14,75]. The accurate and formal description can be specified by Finite State Machines. The other aspect is when the alteration of several entities and their attributes is traced through the elements of the data collection; this flow of actions incorporates a long transaction, i.e., the chain of consecutive transformations [14,76]. The model of exerted impacts of an event can be represented in Finite State Machines. The two sets of models of Finite State Machines—namely the life cycles of entities and the long transactions of events— are orthogonal to each other. One of them meticulously pursues the fate of an entity, the other follows how an event affects the entities and their attributes within a data collection. These two sets of models embody the various integrity constraints of business processes that describe the behavior of the system and the state transitions of the constituents of the system. Due to the hierarchical

nature of the model of a data collection, the behavior of a "super-entity" and the interdependencies of its subsumed entities along with the event-coupled constraints emerges as integrity constraints, which should be consistent and reconciled. For the sake of unified and uniform representation and handling of constraints and distinct models of the system, a hyper-graph can incorporate the diverse models and their representation, including the Finite State Machines [77] (see Figure 7).

— The system responds to a stimulus, i.e., an organizational-level event, that originates from the external environment of the organization and triggers the functions of the Information System through interfaces including User Interfaces (UI). The interfaces can be perceived as documents that are represented in document-describing languages (XML, XHTML, JSON, etc.).

— **Integrity rules** The entity integrity rules that ensure identity are implemented in the various database management systems efficiently and effectively so that the enforcement of these integrity rules is straightforward in any representation, for instance, hyper-graph representation of data collections that are realized in any database management system. The entity integrity concentrates on the validity and correctness of the "tuples", the data items, and the values of the attributes in the data-containing structure. The referential integrity depicts the data dependencies among tuples and entities in a hierarchy within data collections (Figure 8). The specification of business- or enterprise-level integrity constraints can be realized at the data-collection level as static rules or as business rules of activities of processes that are initiated by events.

— The *realization* of integrity constraints happens finally in the Conceptual Model of the Systems (Figure 9). The consistency of the system can be achieved through the maintenance and enforcement of constraints. The hyper-graph makes it possible that the various models that contain the integrity constraints can be represented in a uniform theoretical environment. Furthermore, the hyper-graph representation offers the opportunity for the unified management of model-checking exercises.

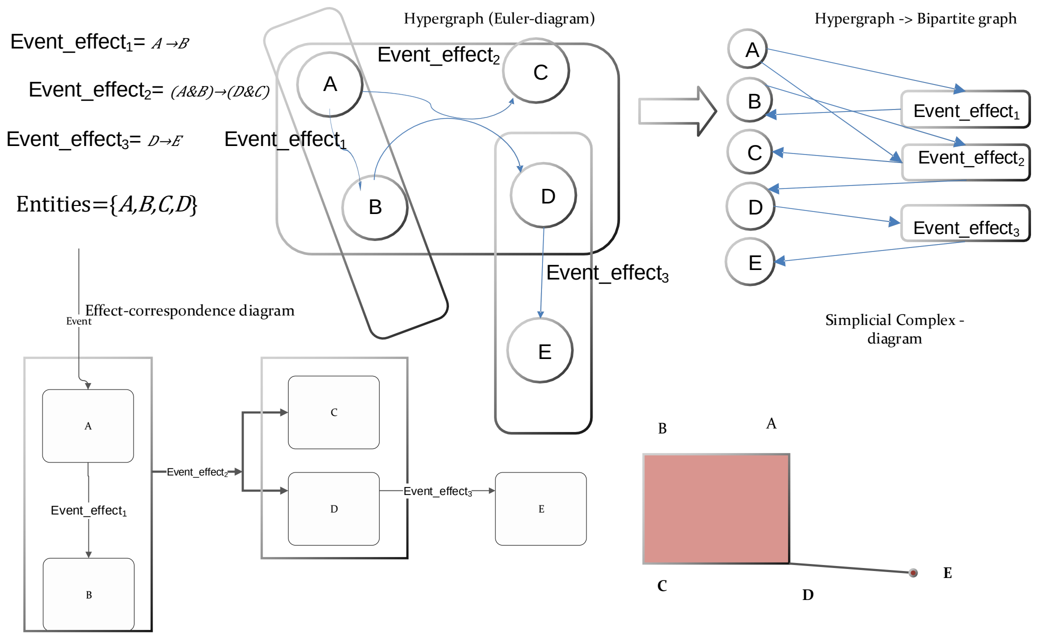


Figure 7. Navigation path: correspondence network of effects of events.

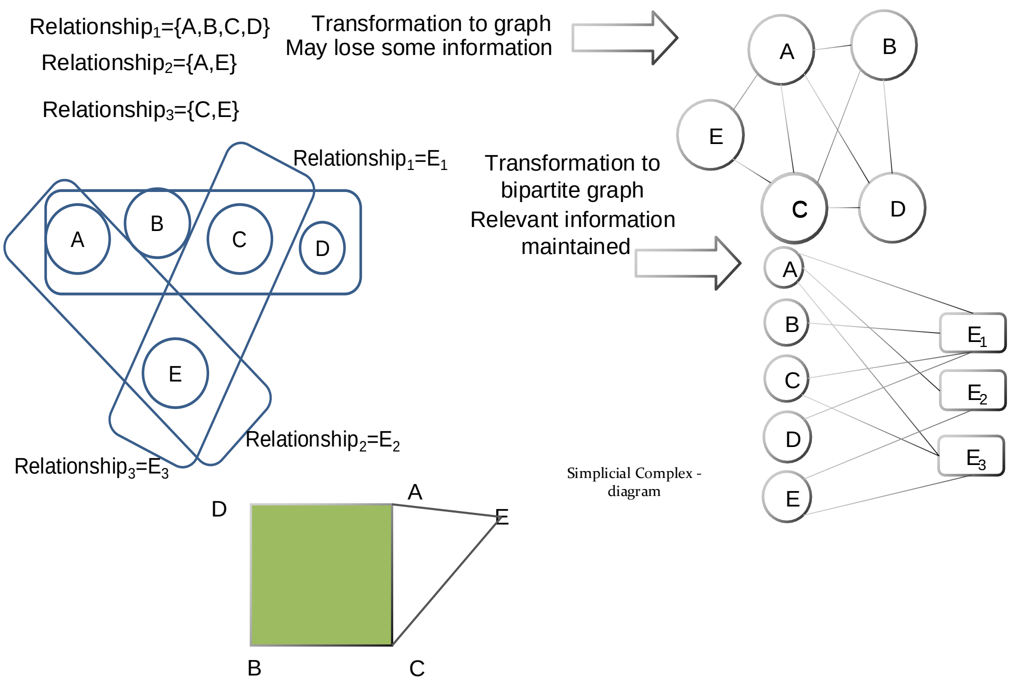
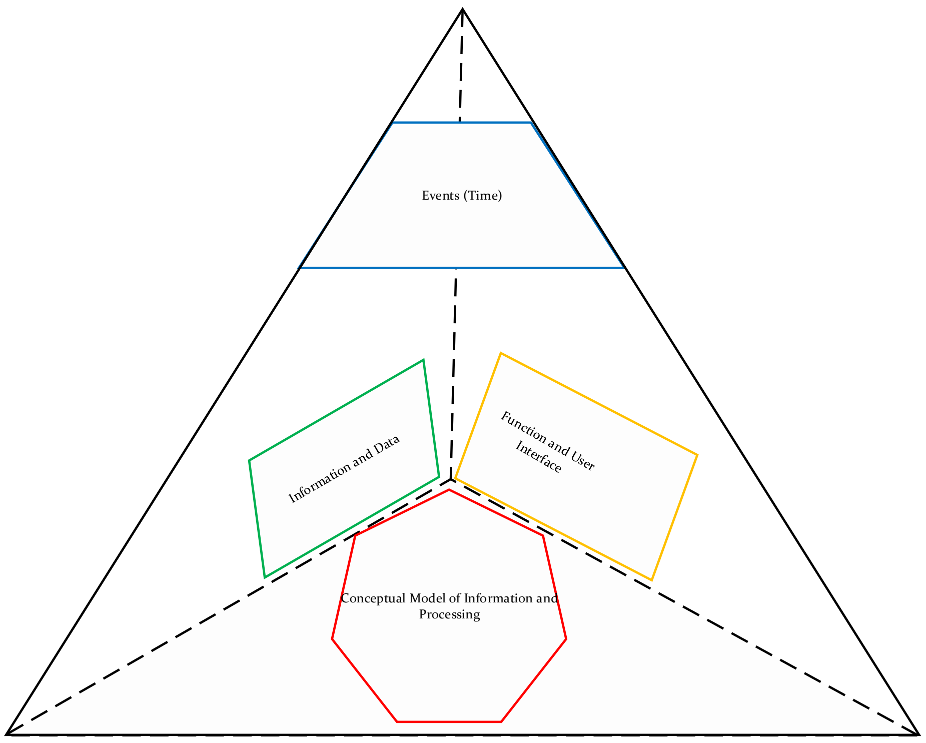


Figure 8. Hyper-graph representation of essential building blocks of the architecture of an IS Relationships among entities (group of attributes of data collections).



Events (Time)
Information and Data
Function and User Interface
Conceptual Model of Information and Processing

Figure 9. Dichotomy among the Three Facets of Information Systems.

3. Literature Review—Related Works

In this section, we provide an overview of the possible applications of hyper-graphs. We survey the hyper-graph-based modeling in the field of document and Information Systems modeling. We provide an outlook of the issues of Enterprise Architectures and the graph-based modeling in this case too.

3.1. Hyper-Graph Application Domains

The fundamentals of representation of knowledge and data are grounded in hyper-graph theories in reality, since the description of the notion that is relevant to a domain contains complex relationships that can be depicted by hyper-networks or hyper-graphs. Bretto [44] provides a good overarching picture of hyper-graph theories that can be used in applied sciences with mathematical rigor. In recent years, hyper-graphs have been broadly used in different studies and different fields. In this section, we describe these distinct fields to explain the importance of the hyper-graph in the modeling of Information Systems. The authors of Ref. [78] used a hyper-graph for clustering by using a clique average to transform the hyper-graph into a simple graph. In addition, hyper-graphs are also used in the field management of data structures [78], in multi-label classification through hyper-graph spectral learning [79]. In image representation and segmentation, the researcher of Ref. [80] formulates the task of image clustering as a hyper-graph partitioning issue. Each piece of the image and its nearest neighbors constitute two separate types of edges that are defined by their descriptors of shape and appearance. Furthermore, hyper-graphs are used to solve many difficulties in the field of image processing [81]. In Computer Vision and Image Processing, high-order relations and patterns exist that can be depicted by hyper-graphs, where the patterns can be defined as similarities by a given metric between vertices and edges in hyper-graphs to detect edges and highlight noise. The researchers of Ref. [82] described a semi-supervised learning method called Hyper-Prior that uses labeling and edge-weighting methods in hyper-graph to find the optimal arrangement. The method exploits a priori biological knowledge as a constraint on achieving an optimal categorization.

3.2. Documents and Information Systems

Ref. [83] analyzes the use of semi-structured and active *documents* represented in XML format. The analysis is extended to cover a methodology for designing for web-based applications . Ref. [84] describes a design methodology for websites; this method follows a meticulous, disciplined, systemic design process that is based on the fundamental concept of documents. Rossi in Ref. [85] worked out a design methodology that is dedicated to *Web Information Systems (WIS)*. Both user interfaces of WIS and the other communication media for the information transfer between the core of WIS and the external environment realize the information interchange through various documents. The design methodology and the perception of WIS according to this paper help to grasp the behavior of WIS.

3.3. Enterprise Architecture and Information Systems

There are several Enterprise Architecture definitions and even standards; however, the Zachman ontology and TOGAF *de facto* standard are apt to describe Information Systems in an organizational environment. Moreover, the heterogeneous models, stakeholders' viewpoints, and views can be handled in a uniform framework [11,12]. The Service-Oriented economy led to the service orientation of business units and consequently the service orientation of Information Systems. The functions of Information Systems that support the activities of an organization are perceived as a service of Information Systems [3]. At the technology and software architecture levels the *Service-Oriented Architecture (SOA)* was developed. The realization of SOA at the platform architecture level pursues two directions. One of them is the industry standard definition (defined by the Object Management Group), the other is based on the REST/RESTFUL approach that is grounded in the HTTP protocol of the Internet. Moreover, the SOA embraces the recent technologies of microservices as well [86–88].

Therefore, SOA can be considered as a reference architecture, so that SOA can support the utilization of software technology at the platform architecture level in companies that should interchange information among partners. Thus, SOA is an architecture guideline for the design of services and their information interaction in Information Systems founded on the concept of "*service*" or "*Web service*" [89,90]. The software architecture paradigms of *SOA* and *Cloud Computing* use the concept of services as a common ground for interaction with end-users and with other partners/systems. In past decades, various input data formats have been elaborated that are usable for the interchange of data with services [91–93].

Some research has tried to arrange the disparate aspects of Information Systems into a unified framework to keep them under the umbrella of the concept of behavior of Information Systems [27,94]. The application of enterprise architecture frameworks offers a tool-set for the conceptual integration of disparate viewpoints [11,12]. The Blokdijk information system modeling framework provides guidelines to grasp the structural constituents of Information Systems[24]. The axiomatic design paradigm can be tailored for Information Systems. This design approach is mathematically grounded and provides guidance for theoretical modeling aspects but also supports the practice used to create Information Systems [95]. The Enterprise Architecture, along with disciplined Business Process modeling methods [66], are flexible tools to model business processes, concepts, entities, and data types. The correct representation of the Business Processes, Workflows, ontologies, structures of concepts, and data in Information Systems profoundly influences the success of the design and development phase. These frameworks help to understand the behavior of the systems and proffer opportunities to utilize the graph-theoretical approaches for model checking, verification, and validation. One of the fruitful approaches is *the document-centric* perception of all constituents since the graph-theoretical representation can be deduced easily from document structures that describe the interfaces of functions and services, the data collections and structures that are included in the information flow, and the data collections that are placed at the core of Information Systems [11,12,24,92,96].

4. The Application of Graph-Theoretic Approaches in the Context of Information Systems

In this section, we look over the experimental design and development that were carried out in the context of Information Systems, as well as document-oriented approaches to exploit the graph-theoretical foundations for model checking and to support the problem-solving in system analysis and design.

4.1. Model Checking for Dynamically Modified Business Processes

Recently, the companies that own Information Systems have been pursuing the agile management and system development paradigms [97,98]. Information technology infrastructure management prescribes disciplined IT service and systems development. However, the dynamically changing external environment enforces the violation of the system development rules in the short term. The correction and mitigation of possible errors may happen later after the modifications have been carried out. To support agility and to maintain integrity and consistency, a method and a related business process to realize it is needed to fulfill the requirements. To achieve that purpose, a model that is proposed by us has been elaborated. Recently, the models of Business Processes have been described in document format, typically in XML. The business processes describing documents are represented in a hyper-graph structure dedicated to business-process representation. The hyper-graph representation is able to reflect the complexity of the relationships among the components of Business Processes. To exploit algorithms that were developed for graphs, the hyper-graph representation transformed into bipartite graphs.

The important components of representations of Business Processes that describe the control and information/document flow are as follows: *multiple merge; multiple-choice; parallel; exclusive choice; AND split/join; XOR split/join; sequence; cycle; and compensation arc/flow*, (see Figure 10) [66,99]. These cover the events that trigger activities, e.g., ingesting information/document, modifying their actual content and the state of data, executing the inputting and outputting acts as *bulk feed; load; download; remove; create.* Besides the manipulation of data collection and the control- and information flow-properties, the process instances are described by the following features *process id; process type; process cost; execution time; and user role.*

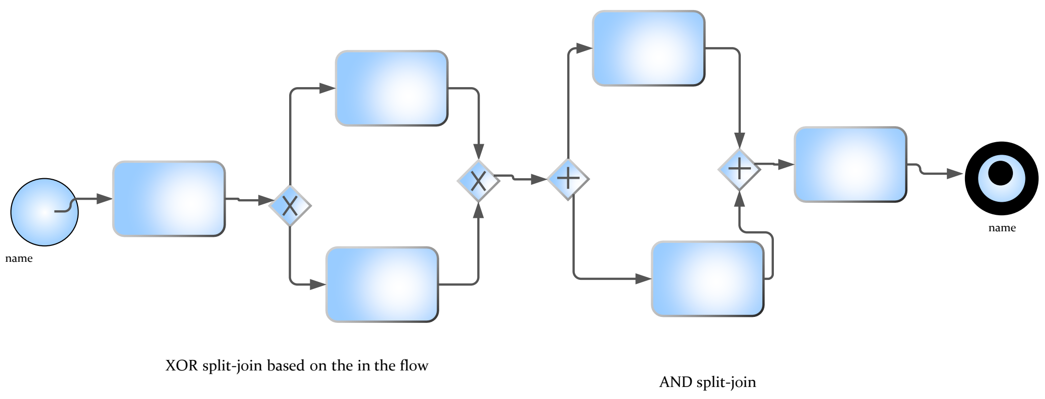


Figure 10. Some examples of possible constructions of information and control flow of Business Processes.

The analysis of the performance of the algorithm requires a reasonable amount of empirical data. One of the best practices is to generate synthetic data to have a sound foundation for the data analytics, in order to verify the model of the data analysis [100]. Therefore, it is a feasible approach to generate a set of data that represents Business Processes that can be categorized as well-formed or with erroneous behavior. The first step was to generate Business Process descriptions in XML. The second step was to transform the Business Process Description into a bipartite graph. This step was a technical one, as the initial transformation of the Business Process representation into a bipartite graph was much more comfortable from a computational point. Then, the bipartite graph was transformed into a hyper-graph. The reason behind this step is that the hyper-graph representation fits the formal analysis and representation of complex relationships, so, the domain of the analysis was placed into a theoretical framework that offers several tools. After the transformation of Business Processes into graph format, the Smith Normal for each representation of Business Processes was calculated (see Section 2.4). The similarity of incidence matrices in Smith Normal Forms indicates a similarity of the graphs we investigated to a degree that is satisfactory for our analysis [101] p. 93, [102,103]. We made use of accessible implemented algorithms to calculate the hyper-graph and simplicial complex representations of Business Processes [55,103]. To explore interesting similarities and dissimilarities, we considered the hyper-graph representation of Business Processes and simplicial complexes. Exploiting the available implemented algorithms, the starting point was the incidence matrix of the hyper-graph. Then, the calculation of the homology groups of simplicial complexes that describe hyper-graphs was the next step. The goal was to discover the properties of Business Processes that can be used for classification to highlight discrepancies, suspicious behaviors, and deviations from adequate standards.

Thus, for example, if the triple {A, B, D} is in a simplicial complex, so are all pairs {A, B}, {B, D}, {A, D}, all singular sets {A}, {B}, {D}, and for mathematical reasons also $\emptyset$; they form a simplicial complex Simplex (A, B, C, D) (see Figure 11). However, even if all three pairs (and thus all three singular sets) are in a simplicial complex, this does not imply that the triple{A, B, D} is also in the simplicial complex. If we think of the simplices in terms of geometric simplicial complexes, a triple may form a filled triangle, whereas the three pairs form the three sides, and the singles form the vertices. The triangle can be filled or not, depending on whether the triple belongs to it or not. Whether the triangle is filled in or not causes a topological difference, which is determined by the homology groups and the corresponding algebraic invariants, namely the Betti numbers (see Section 2.6). Holes and voids within abstract simplicial complexes of various dimensions refer to a structure where some discrepancies, anomalies, or a dearth of connections among elements may exist, i.e., between the representations of Business Processes. In a hyper-graph, subsets $|S| = k + 1$ are perceived as a vector space V_k, $\mathbb{Z}_2$; therefore, the content of the hyper-edges and relationships within hyper-edges is transformed into abstract simplicial complexes. The modulo 2 calculation is the same as creating the symmetric difference of the adequate sets. The *boundary* of k simplex is a collection of $|k-1|$ dimensional faces; a k chain is a formal sum of boundaries of the appropriate simplices. If we have an ordering relation of $|k-1|$ and $|k|$ simplices, the boundary of a $|k|$-chain can be computed by a linear transformation, namely, the linear mapping that is described by the incidence matrix is applied to the vector that describes the $|k|$-chain $vec(v_k)$. It can be seen that the $|k|$-chain is a formal structure, neither the vector components, nor the coefficients have geometrical meaning if we perceive the vector components' values as coefficients of the formal sum, e.g., in the matrix multiplication. In the case of $\mathbb{Z}$, the entries in the matrix and the vectors are integers. The $+1 \quad and \quad -1$, can denote the orientation of the abstract simplex. The boundary homomorphism can be formulated as $\partial(\vec{v}_k) = M_k\vec{v}_k$, where M_k is the incidence matrix that can be defined the following way in this case: $M_k[i,j] = 1$, when the i–the $|k-1|$ simplex is a face of the j–the $|k|$ simplex, otherwise $M_k[i,j] = 0$. This representation of the incidence relationships can be handled by the toolset of linear algebra, and there is no information loss during this kind of transformation. Some algorithms can be employed to calculate the Betti numbers of S abstract simplicial complex. One of the computational approaches is to transform the incidence matrix into the Smith Normal Form [55,56,104,105]. The Smith normal form of a matrix M is an invariant used to describe the incidence relations, as the representation of the simplicial complex that does not depend on any ordering. Two matrices with integer entries M_1 and M_2 are equivalent over $\mathbb{Z}$ if: $\exists A, B, \quad det(A) = \pm 1 \quad and \quad det(B) = \pm 1 \quad and \quad M_2 = AM_1B$ (A and B are invertible matrices).

$$AM_1B = \begin{bmatrix} m_1 & 0 & 0 & 0 & \cdots \\ 0 & \ddots & 0 & 0 & \cdots \\ 0 & 0 & m_r & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{3}$$

From the well-known theorem, $m_i|m_{i+1}$, where r is the rank of matrix M_1, and m_i-s, as invariants, are determined uniquely (apart from units) [106]. The Betti numbers can be read from the Smith normal form of the incidence matrix. The columns with zero entries represent the basis for k cycles, and the rows with non-zero entries constitute the basis of the $k-1$ boundaries. The number of zero columns minus the number of non-zero rows results in the Betti number β_k (the basis of the kernel and image of the mapping represented by the matrix: $Z_k(S) = ker\,\partial_k$ and $B_{k-1}(S) = im\,\partial_k$, the group of cycles and the boundary). Therefore, the homology groups can be computed [55]. Using these invariants, namely the incidence matrices, the boundary matrices of abstract simplices, Betti numbers, the *similar* groups of processes and the potential *discrepancies* in the dynamically changed processes can be discerned. A software and analytics experiment was carried out based on manually generated and synthesized data to prove the viability of the approach [107].

4.2. Hyper-Graphs for Modeling Information Systems from the Aspect of a Document-Centric Approach

The verification and checking of the models of Information Systems utilizes the formal, computational representation of the specific organization. Some components are essential constituents of the organization and IT/IS architecture, namely Business Processes, Workflows, and Data Collections. Data Collections include unstructured data, semi-structured data, and structured data that occur in the format of documents, of data stored in various database technologies, and of data storage architectures that make it possible for processing by organizations. The Business Processes and Workflows are represented within Information Systems as documents (typically in XML or perhaps JSON format). In [27], we outlined an overarching document-centric model that perceives the organization, the data/information, and the processing activities as one generic document that can be represented in a structured way in a hyper-graph. The Business Processes and Workflows can be considered to be acyclic directed graphs, hyper-graphs. Maintaining acyclicity at the highest level of Business Process representation is a reasonable design approach, since the internal loops within a workflow can be wrapped into sub-processes. The relevant aspect of the behavior, the input, and the output of the sub-process can be captured in sufficient detail to assess the overall behavior of the said Business Process. The past development for defining the standard notations for Business Processes made it possible to define compensation branches within Business Processes and Workflows, whereby cycles are generated within the graphs. However, this phenomenon can be handled by separating the compensation branches into a distinct exception-handling representation, i.e., when an exception occurs to trigger compensation branches, to handle the exceptional events, these specific events can be formulated as events that create control flows that lead to separate sub-processes or a stop event within the said business process, but the stop event initiates a distinct business process dedicated to compensation activities.

For the sake of unified and uniform handling, the representation language is XML, since Business Processes, Workflows, documents, and database schemes can be represented in XML [108]. Many workflows and business process description languages are based on XML [109]. Therefore, the XML descriptions of these artifacts can be represented in extended hyper-graphs and directed hyper-graphs. A description of a document in XML is a graph. It is not a strict tree structure because the XML Schema allows for cross-references between distinct branches of the XML document structure, i.e., these links are similar to foreign key relationships in relational algebra. The transformation of traditional graph representations into hyper-graphs is necessary to depict the complex structure of our document-centric model. The document-centric model embeds the comprehensive Enterprise Architecture that includes the following [11,16,17]:

4.2.1. Constituents of Information Systems

Enterprise Architecture
Business Unit An entity partly or wholly fulfills a business function to achieve strategic objectives.
Business Unit Type Based on certain properties of business units, equivalence classes are defined as business units types, and the business units are placed into these types.
Job is an elementary constituent of a Business Unit.
Job description describes the access rights, responsibilities, tasks, and obligations of a Job.
Job Type is a categorization of Jobs through certain properties into equivalence classes.
Actor is an employee who is associated with a Business Unit.
Actor Type is a categorization of Actors through some properties into equivalence classes.
Group may consist of either Actors or Business Units that have a common objective to be achieved.
Location This is a location in cyberspace or a geographical place where a business unit, actor, actor type, job, or group is situated.
Business Process Management and Models
Business Process is a set of activities or tasks that ingests various input items and then generates outcomes that are valuable for the human actors of stakeholders. The collection of activities and tasks are reconciled into the features of the Enterprise and Technology Architecture. The aim of the set of tasks and activities is to achieve business/organizational objectives. Every business process is initiated by a definite business function and may touch several other business processes and business units.
Business Process Model is comprised of a set of models of activities and tasks, and it contains the specification of constraints for operationalization. An *instance* of a business process is composed out of *instances of activities and tasks* that expound

the description of a particular case of the operation of the enterprise. A Business Process Model is a description and prescription for instances of a Business Process. A Business Process Model contains several models of tasks as sub-components. In the graph representation, a Business Process Model is composed of vertices and directed edges that represent the linkages among the vertices of the model. The vertices can represent tasks/activities, decisions, and the directed edges can represent information and control flows.

Workflow A *Workflow Management System (WfMS)* is a software package designed to manage business process design and execution of the business processes represented in a format that the computer can interpret. A *Workflow System (WfS)* is based on a *Workflow Management System* . The workflow system enables the set of business processes that have computer-supported process models to be operationalized [110]. Hence, the Workflow can be considered to be the subset of the Business Process and the Activities. The activities and tasks are coded either by a visual language or a formal language that can be interpreted directly by a computer system. Thus, a workflow is a directed graph of activities and tasks that receive inputs in the form of data and documents and then transforms them into outcomes that appear as documents (and data).

Activity/Task There could be human tasks to be carried out by an Actor through the utilization of computer-supported Activities. An activity can be grasped as a "black box" that embeds data transformation and business rules as its task, and the activity can be perceived as a triple that includes the task and a pair of information items that themselves represented documents. A task has a goal and execution steps of a well-defined algorithm.

Event Events represent those intangible phenomena that initiate changes of state within the real-world system and their IT representations that are relevant to Business Processes and Workflows and exert alteration in the data collection.

Logic gate/Decision In several Business Processes and Workflows, description language calls this component of the visual representation a "Gateway" that describes the control structure, checking mechanism, conditions, and flow of control.

Document-centric approach and Models (Section 4.2.6)

Document Model We consider all the constituents of an Information System to be components of a generic document. The overarching document incorporates all segments of the ingredients of Information Systems that are apt to be described in document representation languages, such as XML, JSON, etc.

Document The activities accept documents, and, after processing, the activities produce outcome documents.

Types of Documents We differentiate between generic, intensional, extensional, ground, and finalized documents.

Variables/Placeholders/Information items In this perception, every document—general-template-like or instance—is comprised of variables that, during processing, become bound [111].

4.2.2. Constituents of Generalized Hyper-Graphs

Pursuing a mathematical and logic-oriented approach means that we think of variables as bound variables, as opposed to the computational approach that allows continuous changes of values contained in variables. (The use of binding the variables and bound variables in a Computer Science analog way that is similar to mathematics is an important approach to apply the logic and mathematics in Informatics [112].) For several reasons, the immutable system and data architecture, especially in a distributed environment, emerged as one of the solutions to handle consistency problems. Even if the underlying computer architecture does not fulfill the expectation of immutability, there is a software-architecture-level solution to provide this feature, e.g., through blockchain [113–115]. For this reason, we assume the immutability of the information items after value assignments.

The above-outlined element of Information Systems represents a complex structure along with complex relationships that can be described by extended hyper-graphs [44]. The generalized hyper-graph makes it possible to obtain hyper-edges nested in each other to represent the complex interactions and interdependencies among the elements that are represented by vertices (Figure 4). As the above-listed constituents of Information Systems showcase, there are three realms that interact and are interdependent of each other even if we represent all constituents in the form of documents (Figures 7 and 11). We can define an appropriate labeling function that reflects the hierarchy of the nested hyper-edges (Definition 2).

$$member_v : E \to \mathcal{P}(V), so\ that\ member_v(e_i) \in \mathcal{P}(V) \quad \text{and} \quad member_v(e_i) = \{v_j \in e_i \in E\}, \tag{4}$$

where $\mathcal{P}(V)$ denotes the power set of V.

The *member* function defines the vertices that belong to a specific edge. As a shorthand notation, and abusing the notation a little, we can write $V(e_i) = \{v_j \in e_i \in E\}$

$$labeling : V \cup E \to Labels = \{l \in \{0,1\}^* | \quad |l| = n \in \mathbb{N}\}, \tag{5}$$

where *labeling* is a labeling function for vertices and hyper-edges, and labels are represented as binary codes for the sake of simplicity. We can assign attributes to edges and vertices, that represent any components of an Information System, i.e., documents, artifacts of the architecture, and elements of the Business Processes. (We can naturally use any alphabet in the labeling function $\Sigma = \{a_i | i \in \mathbb{N}, \}, w \in \Sigma^*$ and $|w| = n$, so that a label is a string of literals.)

$$attribute : V \cup E \to Attr = \{attr_i \in T_i \quad \text{, where } T_i \text{ is a type}, i \in \mathbb{N}\} \tag{6}$$

The *attribute* function assigns attributes to vertices and hyper-edges (cf. Equation (1)). The attributes can be used to attach semantic values to constituents of the generalized hyper-graph. The T type can be considered a semantic domain, and an attribute $attr_i \in T$ expounds a value to describe the semantic content of the said component. Besides the attribute assignment, semantic information can be given to constituents by labeling, too. Each vertex represents a constituent of the three realms within the generalized hyper-graph. We assign a unique identifier to each constituent as a *constituent identifier (cid)*. We assign the label and attributes to constituents to grasp the semantic meaning of the said constituent and, if necessary, other data values as well.

$$\begin{aligned} &value : V \cup E \to Values = \{values\_set = \{x_1, \ldots, x_k\}, x_i \in T_j, \\ &\text{where } T_j \text{ is any type of the permitted domains}, i \in \mathbb{N}\}. \end{aligned} \tag{7}$$

$$\begin{aligned} &attribute\_value : Attr \to Value = \{v_i \in T_j, \text{where } T_j \quad \text{is a type}, and \\ &i, j \in \mathbb{N}, \quad dom(T_j) \quad \text{is the underlying domain of } T_j\}. \end{aligned} \tag{8}$$

The nested hyper-edges can depict the subsumption of the elements to each other, i.e., the sub-element and element hierarchical relationships and the relationships as one constituent represented as a vertex belongs to another constituent represented as a hyper-edge. The generalized hyper-graph makes it possible to describe complex relationships of Information Systems that have three major realms of composition (see Section 4.2.1, Definition 3). The generalized hyper-edges support the representation of complex relationships that are necessary to depict both the static aspect and the dynamic aspect of Information Systems, i.e., data and event processing. A generalized hyper-edge can be considered a simple hyper-graph in its own right. We can depict the hierarchical relationships' subsumption through appropriate labeling and attribute value within the generalized hyper-graph. A vertex of the hyper-graph can be depicted by a set of attributes: $v_j \to \{\langle attr_i \rangle | attr_i \in Labels, \quad attribute\_value(attr_i) \in T_j \quad i,j \in \mathbb{N}\}$.

The hyper-edges can be characterized by a set of attributes, the vertices that are contained therein, and the nesting level; in the case of the directed, hyper-graphs are depicted by the source vertices contained in the tail and the target vertices that are included in the head; furthermore, to make a difference, a Boolean attribute is devoted to designating whether the hyper-edge is directed (i.e., a hyper-arc) or undirected (see Definition 2). In the case of a hyper-arc $\overrightarrow{e}_j \to \left\{\left\langle V(e_j) = \overrightarrow{e_j^+} = \left(e_j^+; j\right); \overrightarrow{e_j^-} = \left(e_j^-; j\right)\right\rangle, \langle\{attr_k\}\rangle \in Labels \quad i,j,k \in \mathbb{N}\right\}$, where a specific labelled *attribute* denotes that the hyper-edge is directed as $directed = true : T(Boolean)$. Naturally, when the hyper-edge is undirected, $directed = false : T(Boolean)$. Another attribute is dedicated to the nesting level or level of hierarchy: $level = p \in \mathbb{N}$, besides *constituent identifier (cid)*.

Thus, the generalized hyper-graph can depict the complex relationships of an Information System. The highest level of the complex relationships is situated in the document/information, processes/workflow, and event/time-related architecture. The architecture building blocks of an Information System are interrelated through their interactions [1]. A building block of an Information System is in unity with its environment and itself if it serves as a constituent element of higher-level systems. Elements of an architectural building block act as lower-level systems. The generalized hyper-graph can represent a system as a unity, i.e., a set of entities and a set of relationships among the entities. Moreover, the decomposition of the system means that every entity can be decomposed into a set of other components that can be regarded as individual system in their own right, and also that they can be perceived as subsystems that can be clearly separated out. This approach makes it possible for a business and/or a system analyst to concentrate on either the system or on the subsystems, depending on the actual interests in the analysis in every specific case (see Figures 3 and 4).

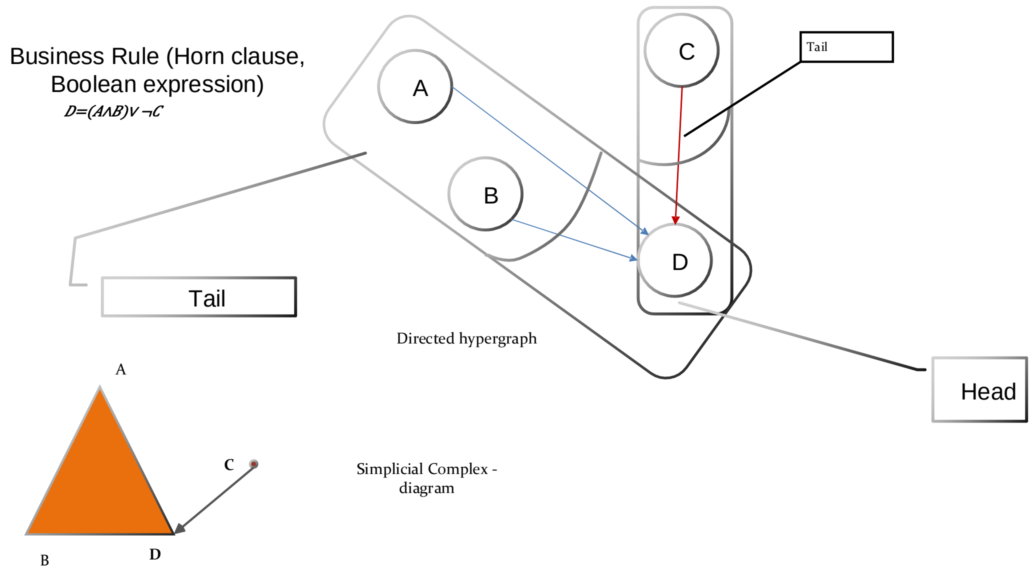

Business Rule (Horn clause, Boolean expression)
D=(A∧B)∨¬C
A
B
C
D
Tail
Tail
Head
Directed hypergraph
A
B
C
D
Simplicial Complex - diagram

Figure 11. Business Rules: logical inferences.

4.2.3. Hyper-Graph Representation for Storing in Databases

We can extend the representation of hyper-graphs in Database Management Systems (DBMS) as follows (see Section 2.4). A constituent of the hyper-graph representation can be a vertex or a hyper-edge in a hyper-graph $Const\_Hyp = Vertex|Hyperedge$. A constituent of the hyper-graph is a tuple:

$$Const\_Hyp = \Big\langle cid, name \in Labels, \langle \{attr_k\} \in Labels \rightarrow attribute\_value(attr_i) \in T_j \rangle, \tag{9}$$
$$\{rel\_link_l\}, cat\_const, \quad i, j, k, l \in \mathbb{N} \Big\rangle, \text{ where } cat\_const = Vertex | Hyperedge,$$

and rel_link_l may designate a logical relationship between the elements of generalized hyper-graphs or any other relationship that cannot be successfully encoded by nested hyper-edges or hyper-arcs. *cat_const* characterizes the category of the constituent. A set of operations that fit to the various search, traversing and graph walking algorithms is defined (see [19]).

4.2.4. Business Rules and Logical Statements

Through an illustrative example, we can look at how the behavior of an Information System can be determined. The data-intensive hyper-edges can depict the information content that can be represented in database schemes, meta-data of data collections, and semi-structured schemes. Each of them is represented as a (nested) hyper-edge that contains the vertices that store the ground-level facts, i.e., the variables and placeholders that contain the values assigned to them considering the data-types of related attributes and characteristics. We showcase a typical template of a workflow that is constructed from patterns of workflows [116,117]. The workflow patterns were defined to lay the foundation for building up complex workflows of business processes. The tasks, documents, data collections, and the related notion of specific business processes are abstracted away to yield a demonstrative example in Figure 12. The workflow model describes the sequential and temporal order of tasks; moreover, it describes the actors, artifacts, documents, and collections of data items that are linked to the tasks. A representation of a business process in a workflow has *one start event* and *one end event*. If there are *several end events*, we join them together into one end event by a minor transformation. The splits and joins of control that flow at gateways of AND/XOR (OR) are used to trigger parallel and alternative paths of execution. Between *the start* and *the end event*, any permitted constituents can occur that are allowed by the constraints expressed in logical statements and rules. The actors can be either carbon agents (human) or silicon agents (machine/application system). The carbon agents are persons, roles, or organizational/business units. The association between the tasks and actors can be described by documents and attributes of the task that characterize the type of relationship, for instance, 'is responsible for the execution, supervision, control, checking, etc. The artifacts are represented by documents or the collection of data items. Besides business processes and tasks, the artifacts in the workflow can consist of the following: (a) organizational goals, (b) products, (c) services, (d) markets, and (e) performance indicators, etc. [11]. The artifacts are stored in sub-documents for processing; therefore, they can be input and output too. The associations between sub-documents/collection of data items in the processing and the tasks are represented in the workflow through document association and explicit connection for designating the role of sub-documents, namely input, output, or both.

Hyper-edges can be defined for data collections, activities/tasks, and rules/inferences. Directed hyper-edges can express the directional relationship (i) between tasks and data collections, (ii) between tasks/activities, and (iii) task-to-task, governed by logical statements or rules. (see Figure 3).

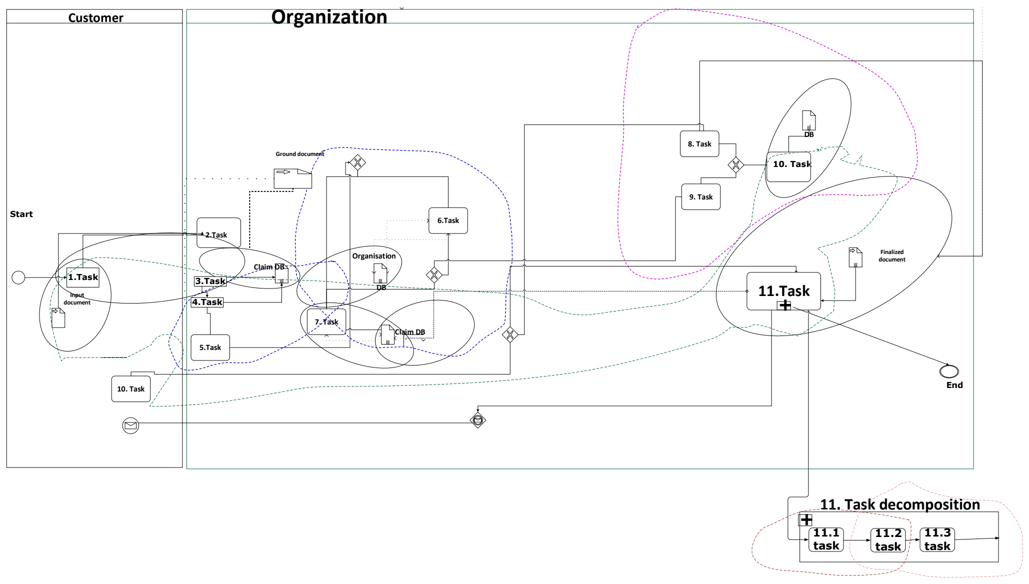


Figure 12. A real workflow; business process description abstracted away.

Such directed hyper-edges contain documents that are input and output to activities as vertices; moreover, the transformation and constraints between the input and output can be expressed by logical statements or rules that are associated with the directed hyper-edge. The representation exploits that vertices can belong to several hyper-edges. In the case of a generalized hyper-graph, hyper-edges can be embedded into other, distinct hyper-edges (Definition 3). The nesting of hyper-edges makes it possible to express complex hierarchical relationships among documents, data collections, and data items. In an analog way, the activities and tasks comprise businesses processes and workflows that contain sub-processes that are also considered hierarchical relationships. The description of business processes is manifested in documents, thereby, the dynamic behavior and the static facets of an Information System can be handled in a unified and uniform way, in a document-centric way. In Figure 12, the ellipses designate the activities/tasks and documents that are associated and wrapped into a hyper-edge. The groups with free-hand-style boundaries (colored and dashed) denote the interrelated activities that are linked together through logical gateways that contain conditions, constraints, and logical statements. These hyper-edges are represented as directed hyper-edges, since they depict directed relationships and inference rules between the elements. The representation of the whole business process can be perceived as a hyper-edge that contains the other hyper-edges. The entire business process consists of tasks that are numbered in Figure 12 from 1 to 11. The set of hyper-edges, where each hyper-edge encompasses a business process or a workflow description, reflects the behavior and usage patterns of the Information System. Any activity/task may enclose several sub-activities, and it can, therefore, be grasped as a sub-process. For the sake of simplicity in Figure 12, we provided an exemplification of a sub-process representation and its description in the form of directed hyper-edges. After a transformation of the decomposition of task 11, its components are configured into hyper-edges. The sub-process contains 11* (*wildcard denotation) tasks that are numbered from 1 to 3, and the sequential relationships among them are expressed by hyper-arcs.

After the transformation in Figure 13, some hyper-edges embrace the data and related tasks as the hyper-edge containing tasks 1 and 2 along with an input document, furthermore task 8 or task 10 associated with a dataset manifested in a database format. Another example is that tasks 6 and 7 are logically related through a logic gateway to task 9; hence, they are embedded in a hyper-arc. The attributes of hyper-edges and vertices, and especially the link attribute, express the directed relationships between the components (see Equation (9)). Furthermore, the attributes can explicate that the dataset in the hyper-edge associated with a task can be of input, output, or both roles. A database, which stores the data and documents that flow, can be a data source and a data sink too, as in the case of tasks 8 and 10. An attribute $data\_flow\_dir = afferent|efferent|afferent\&efferent$ designates what the roles of a document or a set of data items linked to a task are. Tasks 6 and 10 use the same document that is mediated through the database.

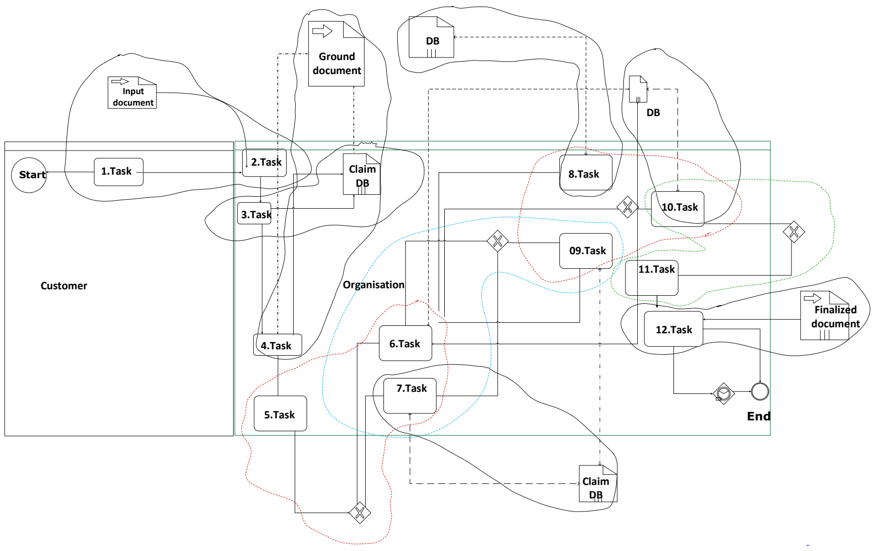


Figure 13. Transforming into adequate hyper-graph components: logical inference and document processing.

This model transformation of business processes and the related collections of data yield a *high degree of flexibility*. The uniform concept of hyper-edges and vertices of hyper-graphs makes allowances for unified and flexible representations of data, documents, business processes, and workflows, and all of these artifacts can be perceived as documents used to handle them in a unified way. The objective of the behavior of an Information System is to achieve its targets, which have been formulated as the requirements. The requirements against an Information System can be formulated as a set of goals. In a *document-centric approach*, we perceive the *goals as documents* that should be in an end-state that is a 'finalized document', which occurs after a 'ground-document' state that is achieved through transformation steps that are carried out within single tasks. The aim of a workflow is the coordination of all constituents and artifacts that are embroiled in the performance of a business process. The coordination can be described by the dependencies among the activities. A workflow handles the data and document dependencies among the activities (one task relies on the outcome of other tasks in the form of data items and documents, see Figure 12) that are governed by control and data flows. The activities use shared resources (e.g., databases and documents), whose usage is controlled through the scheduling of tasks and staff allocation mechanisms. A workflow management system provides the automation of the coordination activities and the components of business processes to support the goals of the Information System [118]. The classical formulation of hierarchy of goals as follows:

$$\begin{aligned} &H\_G = \langle Set\_Goal, \prec \rangle \ \text{, where } H\_G \text{ is the hierarchy of goals,} \\ &Set\_Goal = \{g_j | j \in \mathbb{N}\} \text{ is the set of goals, and } \prec \text{ is a partial ordering relation.} \end{aligned} \tag{10}$$

The ordering relation $\prec$ is a partial ordering of multiple goals, since a lower level goal in the hierarchy, or more accurately in the network, according to $\prec$ the ordering relation, should be satisfied to fulfill the requirements of perhaps several higher-level goals. How to accomplish the goals is coded into the hyper-edges and hyper-arcs that come into existence after a transformation of the business processes that are represented in an adequate denotation, e.g., BPMN Business Process or UML Activity Diagram. The behavior of an Information System can be represented this way (see Equation(10)):

$Behaviour\_IS = \langle H\_G, B\_A \rangle$, where $Behaviour\_IS$ describes the overall behavior of the system, (11)
H_G is the hierarchy of goals and , $B\_A\_G = \{b\_a\_g_i | b\_a\_g_i = \langle g_j, e_k \rangle \; i, j, k = 1 \ldots n \in \mathbb{N}\}$

where $b\_a\_g_i$ denotes business activity interconnected to a goal, g_j the associated goal, and e_k the hyper-edge associated with the activity and consisting of the related *tasks and documents/data items*, altogether representing the activity. Thus, the goals that are organized into a forest of hierarchies depict the decomposition of the system behavior into elementary aims that can be realized by services and microservices [87]. The overarching generic document contains the description of business processes and sub-documents that are in the data-flow of data processing. The generic document contains the workflows at the highest hierarchy level that characterizes the behavior of the system. The workflows are interrelated and linked to each other, and they are represented by hyper-edges at the highest level.

4.2.5. Horn Clauses and the Hyper-Graph Representations

Workflows are composed of business processes that consist of tasks. This hierarchy can be described by nested hyper-edges (see Figure 13). The goals are represented as documents, a set of data items, or documents depicting states to be achieved. These goals are displayed as vertices or nested hyper-edges. The goals are attached to specific fragments of business processes that comprise certain tasks. These fragments are represented as hyper-edges as well. For the sake of simplicity, we assume that the workflows and the business processes involved are sound in the sense of either Petri Nets, UML, or BPMN [119,120] (see Figure 10). The nested hyper-edges contain AND/XOR graphs that depict various, specific fragments of business processes within a workflow [121]. The hyper-arcs may connect a parent vertex to a set of successor vertices, and vice versa, several vertices may be connected to one successor node. These hyper-arcs are the connectors between vertices. We can define the connectivity between two vertices as follows: two vertices are *connected to each* other if there is a directed hyper-path between them, the connectivity relationship is denoted by $\succ$. A hyper-path H_Path_{ot} with $length(H\_Path_{ot}) = n$ in $\overrightarrow{H}$ is a series of vertices and hyper-arcs $H\_Path_{ot} = (v_1 = o, \overrightarrow{e}_{i_1}, v_2, \overrightarrow{e}_{i_2}, \ldots, \overrightarrow{e}_{i_n}, v_{n+1} = t)$, where

$$o \in \overrightarrow{e}^{+}_{i_1} \text{ (tail)}, \; t \in \overrightarrow{e}^{+}_{i_1} \text{ (head), and } v_j \in \overrightarrow{e}^{+}_{i_{j-1}} \cap \overrightarrow{e}^{-}_{i_j}, j = 2, \ldots, n. \tag{12}$$

Vertex $o \in \overrightarrow{e}_i$ (*tail*) is the origin or starting point, and $t \in \overrightarrow{e_i^-}$ is the *terminus*. If $t \in \overrightarrow{e}^{+}_{i_1}$ (tail), then H_Path_{ot} is a cycle. If each hyper-arc in a hyper-path is different from the others, then a *simple path* exists; if every vertex is different from each other, then it is an *elementary path*. In the case of an ***L-hyper-path***, it is required for vertices and hyper-arcs in the series to all be distinct, see Equation (12) [122]. Moreover, it holds that $v_j \in \overrightarrow{e}^{+}_{i_j}$, $v_{j+1} \in \overrightarrow{e}^{-}_{i_j}$, and $\overrightarrow{e}^{+}_{i_j} \subseteq \{o\} \cup \overrightarrow{e}^{-}_{i_1} \cup \overrightarrow{e}^{-}_{i_2}, \ldots, \cup \overrightarrow{e}^{-}_{i_{j-1}}$. We can associate a sub-hyper-graph with $L\_P = (v_1, \overrightarrow{e}_{i_1}, v_2, \overrightarrow{e}_{i_2}, \ldots, \overrightarrow{e}_{i_n}, v_{n+1})$, an L-path from vertex v_1 to vertex v_{n+1}. Let H_{L_P} be the hyper-graph representation of L_P L-path, where $V(H_{L\_P}) = \{v_1\} \cup \overrightarrow{e}_{i_1} \cup \overrightarrow{e}_{i_2}, \ldots and \cup \overrightarrow{e}_{i_n}$ and $E(H_{L\_P}) = \{\overrightarrow{e}_{i_1} \cup \overrightarrow{e}_{i_2}, \ldots, \cup \overrightarrow{e}_{i_n}\}$.

A *B-arcis* a backward hyperarc (F-*arc* forward hyperarc) is a hyper-edge (13)
$\overrightarrow{e_i} = (\overrightarrow{e_i^+}, \overrightarrow{e_i^-}) = (T(e_i), H(e_i))$ (tail, head), where$|H(e_i))| = 1$ (respectively, $|T(e_i))| = 1$).

A directed B-*hyper-graph* (F-*hyper-graph*) is a directed hyper-graph $\overrightarrow{H}$, such that every (14)
hyper-edge in $\overrightarrow{H}$ is a B-arc (respectively, F-arc).

A B-*hyperpath* from o to t in $\overrightarrow{H}$ is a minimal directed subhyper-graph $\overrightarrow{H}'$, such that (15)
the hyperedges of $\overrightarrow{H}'$can be arranged into a sequence $(\overrightarrow{e_1} \ldots \overrightarrow{e_n})$, where
$\forall \overrightarrow{e_i} \in E(\overrightarrow{H}')$. It holds that $T(e_i) \subseteq \{o\} \cup H(e_1) \cup \ldots \cup H(e_{i-1}),\ t \in H(e_k))$.

A transversal of $\overrightarrow{H}$ is a set $T \subseteq V$, such that T intersects all hyperedges of (16)
$T \cap \overrightarrow{e_i} \neq \emptyset\ \overrightarrow{e_i} \in \overrightarrow{H}$.

This is a minimal transversal, T, if it does not include other transversal as a subset. (17)

In a transversal hyper-graph $\mathcal{TR}(\overrightarrow{H}) = \bigcup\limits_{T\ minimal}$, T is a set of all minimal transversals T. (18)

The B-arc connects several vertices of the tail to one vertex in the head, and, vice versa, the F-arc connects one vertex in the tail to several vertices in the head. A BF-graph is a hyper-graph which depicts directed hyper-edges, either B-arcs or F-arcs. This representation of a hyper-arc and a directed hyper-graph is apt to the representation of the AND/XOR graphs of business processes and workflows. A task can be described in a sub-document format by variables, state variables, and attributes. A task can be formally defined such that $task = \langle In, Ou, Constraint, State, var_1, \ldots var_n \rangle$.

Definition 8. *where:*

1. *'In' represents inputs that could be sub-documents or collections of data items.*
2. *'Ou' describes the outputs that could be sub-documents or collections of data items.*
3. *'Constraint' depicts behavioral constraints and goals to be achieved (see Equation (11)).*
4. *'var_i' contains the actual valuation of the variable, and, moreover, whether it is free or has a value, it is denoted as well through the meta-attribute of the variable valuated.*
5. *'free-document' is a document with free variables.*

The AND/XOR (OR) graph related to a fragment of a workflow displayed as hyper-edge can be conceptualized as a predicate according to the dependencies between tasks and conditions that are captured in the logical gateways. A predicate that describes a rule for dependency between tasks can be given by the following:

$task_1 \wedge task_2 \ldots \wedge task_k \rightarrow task_n$ (dependency rule), (19)

equivalently it can be written $\neg task_1 \vee \neg task_2 \ldots \neg \vee task_k \vee task_n$, (20)

$task_i \otimes task_j \rightarrow task_h$ (dependency rule), (21)

$task_i \otimes task_j \Leftrightarrow (task_i \vee task_j) \wedge (\neg task_i \vee \neg task_j)$,

$task_i \otimes task_j \Leftrightarrow (task_i \wedge \neg task_j) \vee (\neg task_i \wedge task_j)$,

an *exlusive-or logical statement* can be formulated as a system of equations ,

$Ax = b\ mod\ 2$ where A is matrix with element 0-1, $\overrightarrow{b}$ is a 0-1 vector, (22)

and x represents the variables.

The system of equations can be perceived as a conjunction of clauses, where the single equations can be considered to be clauses (Equation (22)). These variables are the input/output variables of the tasks and the state attribute of variables of single tasks that designate whether the variable is valuated or not. The XOR relationship can be converted into CNF (Conjunctive Normal Form), since any propositional logic formula can be reformulated as a CNF. The selection structure that chooses the next task can be represented by the logical formulae, if it is necessary, the splitting (XOR gateway) branches can be separated into two distinct statements. The implication can be formulated as a Horn clause [123] (Equation (20)). An atomic formula is either a variable or a predicate with variables, x_i or $P(x_1, \ldots, x_n)$. A Horn clause is the disjunction ($\vee$) of *atomic*and *negated-atomic* formulae, where at most one formula is atomic, i.e., it is not negated. An implication can be considered a Horn clause if it contains exactly one atomic formula and at least one negated-atomic formula [124]. A CNF/DNF Transformation can be carried out automatically [125]. An alloy makes it possible to apply an efficient SAT solver [126–128]. We can formulate a goal of the Information System and a fact, for instance, a finalized document or a collection of data items represented in document format in Horn clauses as an end-state as follows Equation (11).

$$\rightarrow \ doc_i \qquad \text{(fact)}, \tag{23}$$

$$doc_i \ \rightarrow \qquad \text{(goal)}. \tag{24}$$

The logical formulae are associated with the hyper-edges. The goals of the Information Systems and the single Business Processes are expressed in logical formulae, too (see Equation (10)). The *goals* are manifested as such documents that represent *sub-documents or collections of data items in the flow of processing*. The goals can be formulated in an enterprise as follows: (i) specific, (ii) measurable, (iii) achievable, (iv) relevant, and (v) time-framed [129], p. 506. For each end-goal of the Information System—a certain head of a hyper-arc, at the highest level according to the partial-ordering relation—a chain of hyper-arcs should be built up that is based on the ordering relation '$\prec$' (Equations (10) and (11)). The antecedent goal, according to the partial ordering relation, is placed into the head of the hyper-arc; the consequent or subsumed goals are placed into the tail of the hyper-arc. This step is repeated recursively until the lowest level, the leaves, is reached. The partial ordering relation of the goals is transitive, non-reflexive, and antisymmetric. *Goal hierarchy* can be represented in this way in an AND/OR tree. We can describe these features in the form of first-order predicates and implications and transform them into Horn-clauses.

$$cons\_goal(goal_1, goal_2) \wedge con\_goal(goal_2, goal_3) \rightarrow cons\_goal(goal_1, goal_3) \qquad \text{(transitivity)}, \tag{25}$$

where $cons\_goal$ designates a consequent or a sub-goal,

$$\neg cons\_goal(goal, goal) \qquad \text{(non reflexive)}, \tag{26}$$

$$cons\_goal(goal_1, goal_2) \rightarrow \neg cons\_goal(goal_2, goal_1) \qquad \text{(anti-symmetric)}. \tag{27}$$

4.2.6. The Document Model in Hyper-Graphs

A sub-document or the collection of data items in processing can be perceived as a set of variables and attributes, so the goals can be depicted by a set of variables or attributes of a sub-document, respectively. In an analog way, the activities/tasks are represented by sub-documents and the variables of these sub-documents (Definition 8). The activities/tasks and the associated logical rules indicate which variables are valuated and stored in the sub-document or linked sub-documents. The dependency rules express *the valuation relationship* between activities/tasks and sub-documents. The dependency rules between activities/tasks signify whether a variable will be valuated if it occurs in the sub-document format of the activities/tasks.

The document model can be formalized (Figures 14 and 15):

(1) A finite set of variables (attributes) that are represented by vertices in a hyper-edge, $doc_i = \{x_{i_1}, \ldots, x_{i_n}\}$;
(2) A finite set of documents that are represented by hyper-edges ***DOC*** = { $doc_1, \ldots, doc_n$};

(3) The variables that are contained in documents belong to *attribute types* ***Attr*** = $\{T_1, \ldots, T_n\}$;
(4) The finite set of domains is ***DOMSET***=$\{D_1, \ldots .D_k\}$ that contains the domain of each single type, T_i, *attribute type*;
(5) The relationship between an ***OGDT*** *generic document type* (overarching document of organization) and its constituents that are the document types that belong to a ***DTH*** *(document type hierarchy)* can be described by hyper-arcs representing *is-a* relationships; the hierarchy is a mapping of super-type—subtype relationships between document types. The relationships can be deduced from the variables, their attributes, and the types of attributes. A document type is realized either by a DTD or an XML Schema [130];
(6) The relationship between a document, doc_i, and a *document type*, ***DT***, can be described by a hyper-arc representing the *instance-of* relationship.

The document types comprise the document model (Figure 15). The assignment of document types to documents denominates the actual state of their variables in the document flow. The variables receive values by the *valuation* function (Section 4.2.7).

The actual state of documents in the process flow can be implicated from the fact of how many variables are already assigned to specific values. A generic document consists of sub-classes and super-classes of documents that compose a forest of trees of the constituents. Finalization of a document instance results in the fact that all free variables take a certain value. The free documents similar to free rows in tableaux queries [131] can be perceived as documents that contain unvalued variables. While the document progresses through the flow of document processing, the variables are valuated incrementally. Valuation of a free variable may require external data; these data are shipped by system roles outside the enterprise, i.e., outside the Information System. The assignment of values to the variables is determined by the business rules of the enterprise. For that reason, we create differences between the states of *finalized* and *ground documents*. A finalized document is perhaps not yet a finished document in processing. It may still encompass free variables, moreover, error signaling variables that earmark the requirement for further activities by roles in the enterprise. The problem and conflict resolution of documents with defects happens typically by roles in the organization, i.e., outside the Information System. Applying an automated approach for defect removal makes necessary a kind of procedure that can be regarded as intensional treatment from a logics points of view, thereby the utilization of *intensional documents* takes place. Intensional documents are instances of generic document types that are based on business rules implemented by particular tasks. Such a task constructs documents based on business rules. These documents are extensional instances of the said generic, intensional document types. The instances of the extensions with free variables are created from an intensional document through performing the business rules embedded into the intensional document. Then, the variables of these extensional documents and the documents themselves can be manipulated as free-variables and free-documents.

During the document evolution stages, through the impact of activities, the generic document type and document types that are contained in the document hierarchy are instantiated into documents with free variables (*free-document*). Then, the variables are valuated until the documents reach a finalized status; however, the finalized state (*document-to-be-finalized*) does not mean an end-state in an enterprise environment in practice (Definition 8). The finalized document should be modified according to specific requirements until it achieves the end state that may be called a *ground-document*. When all the free variables are valuated, and no change can be executed on the document, then it is in a frozen status. Any modification can be carried out only on a new instance of the document type. This means that the life cycle of a new instance of the document type starts that continues the document development of the said document at a certain point of its life cycle.

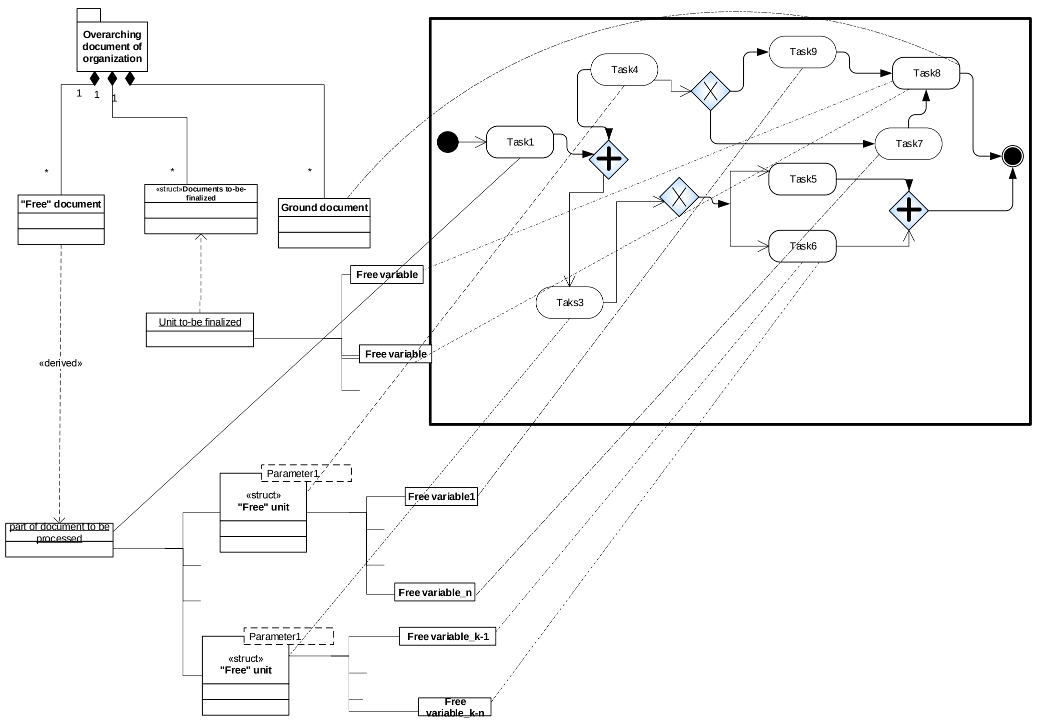


Figure 14. The interrelationships between sub-documents in processing and tasks.

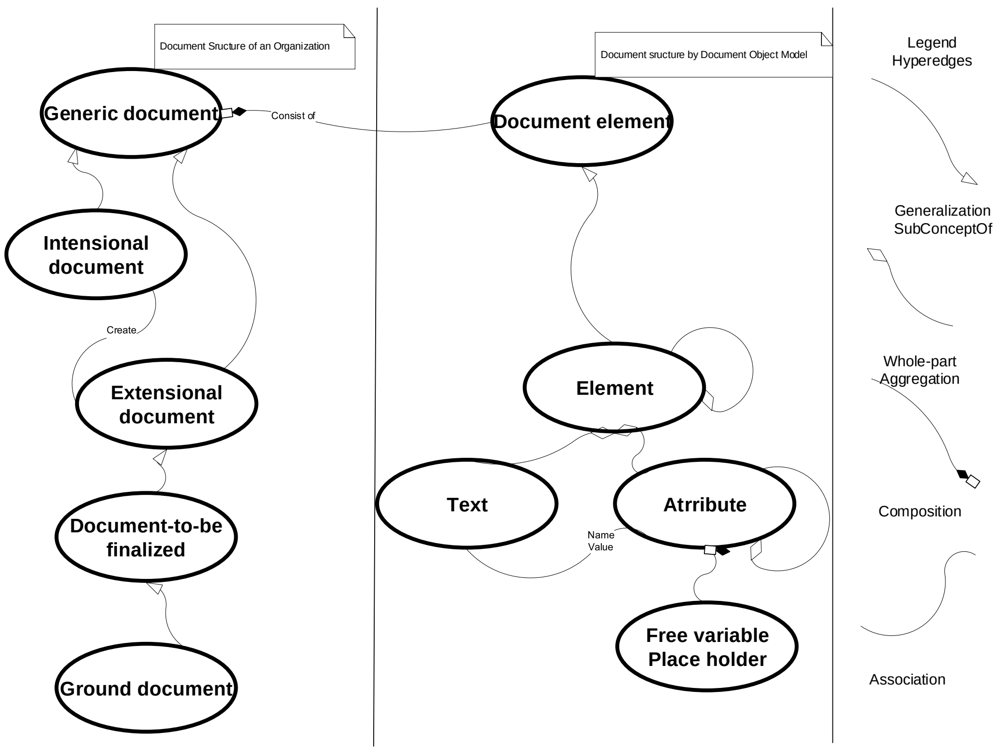


Figure 15. Hyper-graph model of documents.

4.2.7. Exploring the Dependency Rules in the Hyper-Graph Model

The function ν is the valuation function that assigns constants to each variable. If $A = P(x_1, \ldots, x_n)$ is an atomic formula, then $A\nu$ is the result of the substitution when every x_i is substituted by $\nu(x_i)$. A goal can be described by the variables of the associated sub-document in the process flow. When a static analysis is performed, whether a goal is attained is investigated, i.e., the sub-document representing the goal accomplishing the *ground-document status* that means that all variables are valuated is investigated. This fact can be checked by the meta-attribute of the variables, i.e., "*valuated*" (Definition 1). The dependency rule between the variables of a task is represented as a sub-document (inputs) and the variables of the impacted sub-documents (outputs). The starting event of a workflow yields output variables for the consequent tasks, while the ending event demands to receive the output variables of the antecedent tasks. If the goal of the business process or workflow is attained, then all variables are valuated, at least. The sub-document, as a result of the workflow, will be at least in a finalized document state or perhaps in a ground document state, i.e., in an immutable status.

$$\begin{aligned} & task_i(x_{i_1}, \ldots, x_{i_k}),\ doc_j(x_{j_1}, \ldots, x_{j_k}), \\ & x_{i_1} \wedge \ldots \wedge x_{i_k} \rightarrow x_{j_1} \wedge \ldots \wedge x_{j_k}\ , \\ & \text{when the tasks'variables are outputted into documents}\ , \\ & x_{j_1} \wedge \ldots \wedge x_{j_k} \rightarrow x_{i_1} \wedge \ldots \wedge x_{i_k}\ , \\ & \text{when the documents variables are inputted into tasks}\ , \\ & \text{(dependency rule between task's and sub-document's variables)}\ . \end{aligned} \tag{28}$$

The dependency rules induce a hyper-graph (see Equations (5), (19) and (25)):

Definition 9. *Dependency hyper-graph*

(i) *for each variable, $\forall x_{i_j} \in doc_i \in \boldsymbol{DOC}$, there is a vertex in the hyper-graph;*

(ii) *for each dependency rule, there is a hyper-edge that contains the variables from both the left-hand side and right-hand side of the rule $\forall x_1, \ldots, x_n, A_1, \ldots, A_n \rightarrow B$, where $n \geq 1$ where each variable occur at least once in A_i or B;*

(iii) *The hyper-edges that contains the variables —either of the left-hand side or right-hand side of a rule or of a sub-document or a task —are labeled as composite hyper-edges (hyper-vertices);*

(iv) *The hyper-edges that represent the dependencies are labelled by attribute functional dependency, and the content of the attribute is the dependency rule in Horn-clause format.*

The reason that the composite hyper-edge notion is introduced —besides to exploit the properties of the generalized hyper-graph —is that we can handle the composite hyper-edge as a "hyper-vertex", which is represented and substituted as a vertex in a hyper-path in one of the representation views of the adequate sub-hyper-graph, that consists of variables. If we consider the dependency rules as functions, the dependency graph can be regarded as a primal graph (Definition 1). For each fragment of a workflow, we can associate a hyper-edge that can be considered, in the generalized hyper-graph, as a hyper-graph itself.

Step 1 For each task, locate the task or tasks that are indicated by the logical rule (Equation (19)).

Step 2 Then, identify the linked sub-documents and collection of data items.

Step 3 For each logical implication between tasks, create an incidence matrix for a directed graph $A = [a_{ij}]$. This directed graph is manifested as a directed hyper-edge $\overrightarrow{e}_i$ (Equation (2)). A task is displayed as an embedded hyper-edge that represents the sub-document describing the task and contains the variables as vertices in the sub-document that are related to the said task and manipulated by this task. The tasks that are on the left-hand side of the implication will be embedded into the tail of hyper-edge, $task_k \in \overrightarrow{e_i^+} = T(e_i) \rightarrow a_{ki} = -1$. The tasks that are on the right-hand side of the implication will be included in the head of the hyper-edge $task_h \in \overrightarrow{e_i^-} = H(e_i) \rightarrow a_{hi} = 1$. This way, there is a hyper-arc between the hyper-edges that contains the variables of the left-hand and right-hand sides of the dependency rules and represents the dependency rule that is formulated in a Horn clause.

Step 4 For each document or collection of data items, create an incidence matrix for the representation of the related hyper-edges $D = [d_{ij}]$. The directed incidence matrix describes the directed relationships between tasks and documents (data items). When the document or collection of the data items doc_j is input for a task or for tasks, then the documents or the collection of data items is placed in the tail of the hyper-edge, and the related task or tasks are placed in the head of the hyper-edge: $doc_j \in \overrightarrow{e_i^+} = T(e_i) \to d_{ji} = -1$ and $task_h \in \overrightarrow{e_i^-} = H(e_i) \to d_{hi} = 1$. When the document or collection of the data items, doc_j, is output for a task or tasks, then the documents or the collection of data items are placed in the head of the hyper-edge, and the related task or tasks are placed in the tail of the hyper-edge: $doc_j \in \overrightarrow{e_i^-} = H(e_i) \to d_{ji} = 1$ and $task_k \in \overrightarrow{e_i^+} = T(e_i) \to d_{ki} = -1$. In an analog way, we can build up an incidence matrix, $Var = [var_{ij}]$, for every variable contained in a document to track the fate of the variables.

A process chain, an overarching workflow that embeds several business processes, can be represented in a generalized hyper-graph as a chain of business processes that can be depicted as a directed acyclic graph. The single business processes or workflows are constituents of the generalized hyper-graph, and they are represented as embedded hyper-edges that contain other hyper-edges, that can be perceived as stand-alone hyper-graphs; the hyper-arc connects the tasks and the sub-documents, the inputs/outputs, to each other [132] p. 314. The single business processes are connected by directed hyper-edges, the output of the antecedent process in the tail, and the inputs of the successor in the head. Thereby, the sequence of business processes as a chain can be depicted, so that they appear as one of the overarching workflows. Exploiting the flexibility of the generalized hyper-graph and the linked labeling functions, various problems can be investigated. The typical question is whether all variables of a ground document in its end-state are valuated. The issue can be reformulated so that whether each variable in a ground-document is valuated. This formulation makes it possible to check individual variables for whether they have been valuated, and a separate issue is the variables in the logical expressions and whether they are all valuated, i.e., the logical expression is true according to the property of valuation. Another question is whether a variable in an end state of a business process can be reached from a starting state ((*reachability*)). For each document, i.e., set of data items, we identify which task/activity produces the specific document or the set of data items that assigns values to the related variables, respectively, we can determine which task/activity the document expects and the possible valuated variables as input. For each task that has a dependency relationship, we can pinpoint which document plays the role of input and which the role of output. If $[D]_{ij} = d_{ij} = -1$, then the document is input; if $[D]_{ij} = d_{ij} = 1$, then the document is output for the respected task (see Step 4). In an Information System, we have several business process chains. A huge ERP system contains roughly 800 process chains [132], p. 314. Every process chain consists of numerous workflows; one workflow can be comprised of a few business processes, a business process is composed of tasks/activities. We describe these complex, partly hierarchical, structures by hyper-edges that are embedded in each other. Hence, we can investigate whether a single process chain achieved the required end-result that can be formulated in such a way whether the variables and data items in the output documents are valuated and connected to input data items variables in input documents. The analysis can be recursively repeated down to every single task/activity to perform the model checking.

Step 6 *B-hyper-graph and L-path computation* (see Equations (12) and (14)). We locate an end-state and the document that represents the attained state of a business process chain. Each variable in this document is represented as a vertex. We select one variable/vertex, then we identify the tasks that are related to that variable through the incidence matrix that describes the input/output variables and tasks (Step 4) and start building up an *L-hyper-path* through B-arcs. Then, we pursue the tasks and their input variables. These input variables are the output variables of antecedents tasks that can be determined again by the incidence matrix. Since an L-hyper-path is an alternating sequence of vertices and hyper-edges containing tasks and a set of variables, an L-hyper-path connects the variables in an end-state of a business process or chain to the variables of one of the start-states of another business process or the business process chain. The B-arcs are determined by the incidence matrix of the variables and the tasks, so that the matrix makes it possible to establish several continuations of the respected L-hyper-path. The collection of L-hyper-paths constitutes a B-hyper-graph. It is the set of domains of data types that cares for the values of variables. For every process chain and each of their end-states, the collection of L-hyper-paths and B-arcs that are encompassed by their related B-hyper-graphs should be constructed. This procedure should be repeated for each workflow and business process that is a constituents of a business process chain, then for every task within a business process.

The business process chains and the business processes that constitute them can be represented by acyclic directed graphs. The loops and the cycles can be hidden in sub-processes without abusing the integrity of the processes themselves. A business process can be regarded as a long transaction, since the requirement of interaction with other systems, batch services, and human actors can take hours or even days. In the case of any failure, it is a necessary kind of rollback, although it differs from a typical database transaction that is performed in a short time [133]. In business process modeling, to make the rollback mechanism operable, every task should have a related compensation activity that can reverse the impact—in the sense of the data and the state of the system—of the task [66,134]. Nevertheless, the modeling properties of BPMN can be exploited so that the compensation transactions coupled to a specific business process can be separated into distinct process. In this way, the coupled business processes communicate to each other through status messages to reverse the effects in the case of any faults of the business process and the underlying transactions. Thus, we avoid cycles in the hyper-graphs. Nevertheless, there is a duplication of the business processes. The compensation process is carried out in reverse order according to the original respective business process. Thereby, model checking can be executed for the hyper-graph representation of the business process whether all required variables within the compensation process are valuated in the related documents and set of data items. Using this representation approach, we can eliminate the cycles within the graph that would be built in by design and the representation transformation. We apply the generalized hyper-graph representation to depict the complex, partly hierarchical relationships among the numerous relevant constituents of an Information System. Exploiting the labeling function and the attributes assigned by the labeling function to hyper-edges and vertices, we can distinguish directed hyper-graphs describing single business process chains, single workflows, and single business processes. Thus, we can associate a hyper-graph with a business process chain, a workflow, or a business process. We call a directed hyper-graph, for instance, a hyper-graph associated with workflow $w\text{-}f_i$, a hyper-graph associated with a business process chain $b\text{-}p\text{-}chain_j$, or hyper-graph associated with a business process $b\text{-}p_k$.

Step 7 *B-hyper-path computation*. After identifying the B-hyper-graphs that are associated with business process chains, workflows, and business processes, we can calculate the B-hyper-path. In the case of B-hyper-paths, we start again with one variable in some end-state, then we create a B-hyper-path Equation (15). We repeat this procedure for all variables that can be found in documents representing end-states in business process chains, workflows, and business processes.

Step 8 *Transversals and path computations from hyper-vertices* [44], p. 37. After discovering the possible B-hyper-graphs that are associated with business process chains, workflows, and business processes, we can calculate the transversals for each identified B-hyper-graph. In the case of transversal computation, we start with the hyper-edge that represents the documents in some end-state and contains a set of variables.

A minimal transversal T of B-hyper-graph $\overrightarrow{H_{b\text{-}p\text{-}chain_j}}$ / $\overrightarrow{H_{w\text{-}f_i}}$ / $\overrightarrow{H_{b\text{-}p_k}}$ is a collection of the tasks whose output variables can be used as input variables of $b\text{-}p\text{-}chain_j$, $w\text{-}f_i$, $b\text{-}p_k$ (Equation (18)). We repeat this procedure for all documents that can be found in documents representing the end-state. We compute the *L-hyper-paths* repeatedly in such a way that the set of variables in a document within an end-state, which is enclosed in an adequate hyper-edge, is replaced in the representation by *one vertex* (hyper-vertex = hyper-edge identified with one compound vertex) that contains all variables with their properties. Thereby, we acquire another collection of L-hyper-paths.

The *L-hyper-paths*, *B-hyper-paths*, and *B-hyper-graphs* describe a possible connection between a variable stored in a document representing an end-state and a possible input variable or variables that play a role of value-setting to give an initial value to the variables. After this, we built up all possible *L-hyper-paths*, *B-hyper-paths*, and *B-hyper-graphs* that can be explored from the single variables and sets of variables in documents representing the end-state of business process chains and single business processes. Thus, we have the opportunity to investigate several questions regarding the integrity and consistency of the Information System. All minimal transversals of a hyper-graph $\mathcal{TR}(\overrightarrow{\mathcal{H}})$ can be computed, where $\overrightarrow{\mathcal{H}} = \overrightarrow{H_{b\text{-}p\text{-}chain_j}} | \overrightarrow{H_{w\text{-}f_i}} | \overrightarrow{H_{b\text{-}p_k}}$. If the hyper-graph size is reasonable in the sense of hyper-edges, there are efficient algorithms to calculate the minimal transversals [135,136].

Step 9 *Consistency checking of generated dependency rules*. Each minimal transversal, $\overrightarrow{T} \in \mathcal{TR}(\overrightarrow{\mathcal{H}})$, is a representation of dependency rules that can be identified from the models. For every edge $\overrightarrow{T}$ and the minimal transversal $\mathcal{TR}(\overrightarrow{\mathcal{H}})$ containing it, we check whether, when starting from the head part of some end-state, the associated attributes and logical statements are consistent (see Equation (9)) .When a new workflow or new business process is inserted into the system, the attainable goals are formulated as variables of documents stored in some end-state, either of the business process chain, particular workflow, or the said business process. If transversals T, in a minimal transversal $\mathcal{TR}(\overrightarrow{\mathcal{H}})$ (that is associated with $b\text{-}p\text{-}chain_j$, $w\text{-}f_i$, $b\text{-}p_k$) fulfills the behavior and other constraints, then the tail $T(\overrightarrow{T})$ containing logical conjunctions of tasks and the head $Head(\overrightarrow{T})$ containing either variables of documents or a task represent a valid dependency rule (Equation (11)).

The *B-hyper-graphs* constructed during the procedure contain the *B-hyper-paths*. A B-hyper-path connects one variable or a set of variables in the document (*hyper-vertex*) stored in an end-state to variables in the documents in one or more start-states. In the generalized hyper-graph, there may exist numerous B-hyper-graphs (Figure 16). A B-hyper-path can be used to check whether one variable or a set of variables can be linked to input variables. Since the B-hyper-path is created backward from the vertex that represents a variable, the path connects the said variable to a set of variables to input variables of tasks within business processes. The L-hyper-path is strictly linear, the B-hyper-path allows branching. Thereby, a B-hyper-path connects a variable to a set of input variables of various tasks according to the dependency rules. Alternatively, a hyper-path may start backward from a *hyper-vertex*—a set of variables contracted to one vertex that represents one document—and locate the potential input documents in the form of a hyper-vertex that is valuated within the backward walk. The *L-hyper-paths* are strictly linearized sequences of vertices and edges (Equation (15)), so that they connect a variable in some end-state to a variable in some start-state or a set of variables in the end-state to a set of variables in a start-state. It depends on which type of vertices are selected for constructing an L-hyper-path, a simple one representing one variable or a compound one that represents a set of variables as a hyper-vertex.

4.2.8. Model Improvement through Model Checking

We explore the conceivable application of the hyper-graph model to find either a business process, business processes, or a workflow to address a particular issue to solve problems. We constructed the imaginable hyper-paths, which are chains of hyper-edges that can be deduced from the models through transformation into hyper-graphs in the previous section (Section 4.2.7). The incidence matrices associated with hyper-graphs and sub-hyper-graphs are a suitable basis for dealing with these problems, besides the hyper-paths. There are two realms of the potential investigation; one is the examination of the existing system, and the other is the analysis of the incrementally enhanced system by new services. We should look at these issues as follows: (a) seeking out the appropriate business processes (chains or workflows) to figure out which would produce the anticipated output in the form of a variable or data item, (b) finding the appropriate business processes (chains or workflow) that generate the expected document or documents, and (c) and recognition of the problematic or "critical" business processes (chains or workflow). Since the tasks, workflows, business processes, business process chains, and documents are represented in XML, the meta-attributes and properties of the variables for navigation and walking through the hyper-graph representations can be denoted by elements or attributes of elements in the related XML document (tagging) [109]. For example, if we select a single output variable, x_j, then we can identify the related document and the incidence matrix, $Var = [var_{ij}]$ (see Step 4). Then, there are some options:

Option 1. $\forall i, [Var]_{ij} = var_{ij} = 0$, there is no hyper-edge representing a task that contains that variable in its head. It should be investigated whether this variable is necessary and, if yes, which business process and its task should produce it as an outcome.

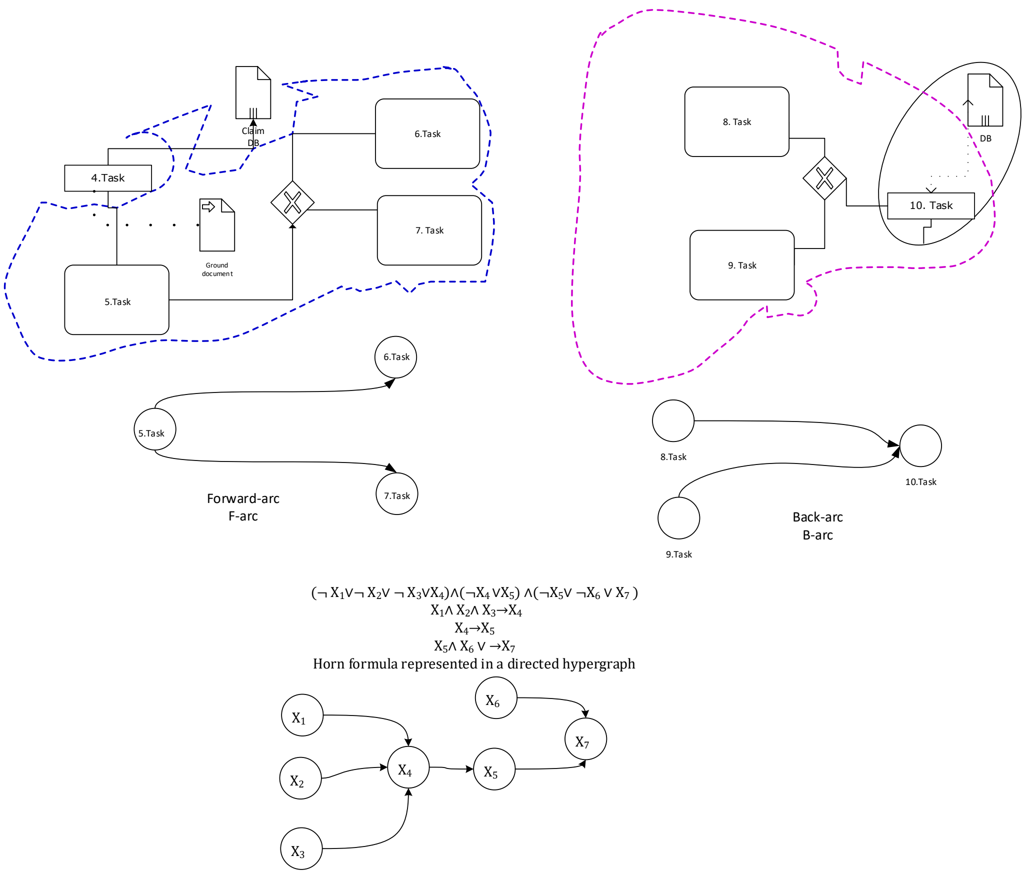
Option 2. $\exists i, [Var]_{ij} = var_{ij} = 1$, then the meta-attribute of this variable should be checked to see which task gives value to this variable. If there is at least one task that sets the meta-attribute of the variable to "*valuated*", then the relevant business processes and tasks can be found through an *L-hyper-path* and a *B-hyper-path*. If there does not exist a task that valuates the variable, then it should again be investigated whether this variable is necessary and, if yes, which business process and its task should produce it as an outcome, i.e., it seems to be that a valid input does not exist, or it did not succeed in constructing a hyper-path that connects the resultant variable to any valid input variable or set of input variables. The causes should be investigated, and the possible solution should be identified.

Option 3. If $[D]_{ij} = d_{ij} = 0$, then there is no hyper-edge representing a task that contains that document—and the set of variables in the document—in its head. Whether this document is necessary should be investigated and, if yes, which business process and its task should produce it as an outcome, if this situation occurred.

Option 4. $\exists i, [D]_{ij} = d_{ij} = 1$ then the meta-attribute of these variables in the document should be checked to see which task gives value to these variable. If there is at least one task that sets the meta-attribute of the variable "*valuated*" to *true*, then the relevant business processes and tasks can be found through an *L-hyper-path* and a *B-hyper-path*. If a task that valuates one of the variables does not exist, then it should again be investigated whether this variable is necessary and, if yes, which business process and its task should produce it as an outcome, i.e., it seems to be that a valid input does not exist, or it did not succeed in constructing a hyper-path that connected the resultant variable to any valid input variable or set of input variables. The causes should be investigated, and the possible solution should be identified.

Option 5. $\exists i, [Var]_{ij} = var_{ij} = 1$ and the meta-attribute "*valuated*" of this variable is *true*, then at least one or more *L-hyper-path* and *B-hyper-path* exist that yield viable sequences of variables and tasks in business processes that lead back to valid start-states and associated input variables. If there are several allowed sequences, which path is worth retention should again be investigated. Other factors can be contemplated in the selection procedure: e.g., the length of the path of the alternating tasks and input variables; the complexity of the paths investigated; and, moreover, the reliability and trustability of tasks and contained business processes; furthermore, the cost factors in an IT sense should also be investigated.

The *L-hyper-path* that connects a single variable through the dependency rules to another single variable when the variables are perceived as single vertices is strictly linearized. Instead of a single variable, we can select a composite hyper-edge, a hyper-vertex that exhibits a document, then we can build up an *L-hyper-path* that represents an alternating path between document vertices and hyper-edges, including tasks. The construction of a hyper-path happens while backward walking, starting from the designated t *terminus* to the o *origin*. A *B-hyper-path* permits a more loose connection; the starting point for building up is either a single variable or a document in a backward walking. We may eventually acquire a set of tails that contains a set of variables or documents that are connected to a *terminus*. The terminus can be either a single variable or a single document containing a set of variables.



Claim DB
6.Task
4.Task
7. Task
Ground document
5.Task
8. Task
DB
10. Task
9. Task
6.Task
5.Task
7.Task
Forward-arc
F-arc
8.Task
10.Task
9.Task
Back-arc
B-arc
$(\neg X_1 \vee \neg X_2 \vee \neg X_3 \vee X_4) \wedge (\neg X_4 \vee X_5) \wedge (\neg X_5 \vee \neg X_6 \vee X_7)$
$X_1 \wedge X_2 \wedge X_3 \rightarrow X_4$
$X_4 \rightarrow X_5$
$X_5 \wedge X_6 \vee \rightarrow X_7$
Horn formula represented in a directed hypergraph
X_1
X_2
X_3
X_4
X_5
X_6
X_7

Figure 16. Workflow fragments and their relationships with the transformed B-arc and F-arc.

Option 6. $\exists i, [D]_{ij} = d_{ij} = 1$ and the meta-attribute *"valuated"* of all variables in the document is *true*, then at least one or more *L-hyper-paths* and *B-hyper-paths* exist that produce a feasible series of variables/documents and tasks in business processes that lead back to valid start-states and associated input variables. As in Option 5, we should investigate which path is viable, reasonable, and consistent with the intentions of the system.

Option 7. If either $\exists i, [D]_{ij} = d_{ij} = 1$ or $\exists i, [Var]_{ij} = var_{ij} = 1$ and the meta-attribute *"valuated"* of all variables in the document is *true*, however, neither *L-hyper-path* nor *B-hyper-path* exist that provide a feasible sequence of variables/documents and tasks in business processes that lead back to any valid start-states and associated input variables, the required input variables are not known or not reachable.

When new requirements in the form of new goals are specified, then we translate the requirements into documents and variables in the documents so that the same approach can be applied as before (Equation (10)).

In the case of Option 7, the unavailable or unreachable input variables should be investigated additionally for every potential *L-hyper-path* or *B-hyper-path* to figure out whether the necessary input variables can be generated through an adequate path, i.e., through a valid and reasonable path of tasks and business processes. During the computation of *B-hyper-paths*, *B-hyper-graphs* come into existence, and the issues can be investigated within these hyper-graphs. The investigation of the issues that emerged is a *supervised operation*, i.e., a recursive procedure, because there is no guarantee that the solutions can be found automatically if there is a specification error. During the analysis, we can use the theorem from Ref. [122].

Theorem 1. *Let $\overrightarrow{H} = (V, \mathcal{E})$, $\mathcal{E} \subseteq \mathcal{P}(V)$ be a directed B-hyper-graph. Let $U \subseteq V$, and let $F \in \mathcal{E}$. Then, there is at most one L-hyper-path in $\overrightarrow{H}$ with the set of vertices that are exactly U and with the set of hyper-edges that are exactly F.*

The consequence of the Theorem 1 is that a polynomial-time greedy algorithm exists. The algorithm makes it possible to check whether there is a path between an outcome variable and a required input variable. This checking is interesting, especially in the case of a new goal formulation. We tried to close out possible cyclic behavior inherent in business process modeling and design that appear because of compensation processes, so that the compensation processes were separated into distinct processes. Despite that, some hidden cycles may exist that may cause the issues mentioned in Option 7. Ref. [137] analyzes and discusses the various notion of acyclicity of hyper-graphs and their interrelationships, namely alpha, beta, and gamma acyclicity. That paper provides a rule-based characterization for checking the acyclicity of hyper-graphs in all three reasonable acyclicity definitions.

Proposition 1.

1. *alpha (α) acyclicity:*
   - 1.1 *The hyper-graph H is α acyclic if it is conformal, where conformal means that a hyper-graph H is conformal if every clique of the hyper-graph is included in a hyper-edge, and free of cycles, i.e, cycle-free.*
   - 1.2 *The hyper-graph H is GYO reducible [138].*
2. *beta (β) acyclicity:*
   - 2.1 *The hyper-graph H is β acyclic if each subset of H (or each sub-hyper-graph of H) is α acyclic.*
   - 2.2 *The hyper-graph H can be reduced to an empty set by repeatedly removing nest points. A vertex v is a nest point if there is a linearly ordered sequence of hyper-edges that include each other consecutively and all contain v, $\{e_1, \ldots, e_n\}, e_{i+1} \subset e_i, \forall i < n$, and $v \in e_n$ [139].*
3. *gamma (γ) acyclicity:*
   - 3.1 *The hyper-graph H is γ acyclic, if H is β acyclic and there are no vertices v_1, v_2, v_3 so that $\{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_2, v_3\}\} \subseteq H[\{v_1, v_2, v_3\}]$, where $H[\{v_1, v_2, v_3\}]$ is the induced sub-hyper-graph, e.g., $W = \{v_1, v_2, v_3\}$, $W \subseteq V$ the vertices of H, $H[W] = \{e \cap W | e \in H\} \setminus \{\emptyset\}$.*
   - 3.2 *The hyper-graph H can be reduced to an empty set by repeatedly removing nest points.*
   - 3.3 *The hyper-graph is DM reducible [140].*

Corollary 1. *Alpha (α) acyclicity, beta (β) acyclicity, gamma (γ) acyclicity are in the complexity class of polynomial time [137].*

According to Ref. [141], the alpha acyclicity belongs to the complexity class of linear time. The GYO, DM, and beta elimination reduction algorithms for directed hyper-graphs can be employed to detect the cycles in single hyper-graphs (B-hyper-graphs) [36,142]. The issues that were conceptualized in Option 7 must be examined further through exploiting the cycle elimination algorithms and through scrutinizing the models and designs of the sequences of the inputs, tasks, and outputs. Detecting L-hyper-paths, B-hyper-paths, and hyper-graphs that contain a cycle provides the opportunity to investigate the cause of the cycle that remains after placing out the compensation processes. Since we applied an encoding for the model of Information Systems that alternately deposited the variables/data items and the manipulating tasks of the processes in the tail and head of the hyper-arcs, the hyper-arcs express dependency rules between the variables and the manipulating tasks between the tasks.If the cycle is justified from an organizational point of view, then the tasks and coupled variables can be inserted into a sub-process; thereby, the cycle is eliminated at the topmost hierarchical level of the processes, and it is replaced by composite sub-processes and their input and output variables. This solution makes it possible to continue the analysis of whether the necessary input variables are reachable or exist at all. In an agile development environment, or dynamically changing business process environment, this evaluation method assists in pinpointing problem areas. The incidence matrices for documents permit the investigation into whether a set of variables can be produced. If there are problems with some variables, then the incidence matrices for those variables can be used along with L-hyper-paths and B-hyper-paths. If there is no reasonable path between a specific variable and the required inputs, or suitable input variables cannot be found, then this situation compels model modification and intervention from the modelers.

5. Discussion

The Information Systems in an organizational environment are highly complex systems like the Enterprise Resource Planning systems (ERP) [143]. Hence, one of the opportunities to investigate the disparate phenomena of system behavior is to simulate the activities of Business Processes. The recent technologies make it possible to simulate a reasonable part of relevant business processes. The traditional rule of thumb in the case of Corporate and Enterprise Information Systems is that three separate software/system environments should be maintained, namely development, testing, and production [37] p. 7. The modeling and simulation of any parts or components of the operating Information System in a virtual environment through the exploitation of Model Checking technologies that are based on formal mathematical representation, i.e., hyper-graphs and simplicial complexes, and the above-outlined set of algorithms is a viable approach. Change Management in a quasi-real-time and dynamic environment becomes a critical issue, since the required modifications at any level of the Enterprise Architecture should yield advantageous alterations of the operating Information Systems and, at the same time, should care about the minimal disruption of Information System functions [144].

The described operationalized formal approaches can be placed into a unified framework. The framework consists of a simulation environment, a transformation system, and the development environment of Information Systems. The transformation environment translates the models into representations based on graphs and related theories. The simulation environment contains analytical methods and model-checking algorithms to investigate the models' consistency and integrity and also to highlight problems and explore similarities and dissimilarities between the changed and original components. A transformation method and algorithm has been created that can translate UML Activity Diagrams (then later Business Process Model Notation descriptions) into a Yawl workflow interpretation engine [145,146]. The reason for this transformation is that Yawl is an open access system that can operationalize the representation of Workflows and Business

Processes [147]. The representation of the Business Process can be examined by using transformations into Finite State Machines that are described in hyper-graphs and then performing Model Checking activities on the representations [77].

The Business Processes' description in hyper-graphs is transformed into bipartite graphs and presented in matrices according to the Smith Normal Form. The Smith Normal Form representation offers the opportunity to evaluate the dissimilarities between Business Processes, i.e., the violation of integrity and consistency in the case of dynamically changed Business Processes can be highlighted [107]. The hyper-graphs can be perceived as simplicial complexes or mapped onto simplicial complexes. The homology groups and Betti numbers (i.e., the ranks of homology groups) that can be calculated indicate the loops within simplicial complexes and, thereby, the phenomena in hyper-graphs that are worth investigating. The description logic statements can be represented by components of hyper-graphs, hyper-edges, and their labels [30]. The labeling functions make it possible to link the first-order and description logic statements to constituents of hyper-graphs representing single models of Information Systems. The description logic statements themselves can be represented in hyper-graphs, and proofs can be automated by adequate engines so that the model checking and the satisfaction of constraints and formulated business rules can be carried out as well [148,149].

The various transformations and applications of diverse representations focus on supporting model checking in terms of consistency, integrity, compliance, and conformity to the established rule set. The checking of the disparate models of Information Systems from distinct viewpoints requires different representations and sets of algorithms, since single specific representations and the linked theoretical backgrounds alongside the operationalizing methods yield only as many interpretable results as will cover specific aspects of Information Systems. Thus, the approaches described in this paper can highlight the potential problem areas in ongoing development projects. In tandem with human interactions, issues can be explored, and then mitigation actions can be initiated. The $P = NP$ or $P \neq NP$ concern does not cause any serious consequences nowadays regarding the problems regarding size being within the class of the problems that can be managed and handled by humans [38]. Especially in the field of Information Systems, the number of workflows and business processes are finite. The workflows, business processes, sub-processes, and tasks can be measured by their graphical representations in vertices and edges, these numbers—even in the cases of large ERP systems —are bounded so that, in a computation sense, model checking remains in reasonable complexity. Moreover, the practical investigation of a large set of Horn-clauses is computationally feasible since efficient SAT solvers are available [128].

We selected and used generalized hyper-graphs instead of other graph-based approaches, since the hyper-graph makes generalization and specialization both possible at once. Other graphical approaches, such as, e.g., meta-graphs, are feasible and viable approaches. Notwithstanding this, meta-graphs lack the flexibility of defining various layers of views and the capability of the superimposed viewpoints that can be handled simultaneously [150]. Moreover, the mathematical theory of hyper-graphs is broad, and the related mathematical theories involve various finely distinguished versions of hyper-graphs that have coupled a rich set of computer science approaches, methods, and algorithms. The computer science-based algorithms make it possible to operationalize the models coded in hyper-graphs; therefore these methods provide application opportunities in numerous fields, including Information Systems.

6. Future Directions of the Research

The Virtual Twin of Operating Information Systems within an Enterprise is such an important approach to mimic the actual system in such a way that it becomes a mapping of the real system onto a simplified representation (Figure 17).The representation contains those essential components and models taking into account the essential constituents from the viewpoint of the current analysis. The Virtual Twin approach can follow the same thinking line as the Digital Twin approach in the context of Cyber-physical Systems [151]. The major ERP systems have models and representations of their business processes. They use one of the standard notations, e.g., SAP [143]. In an industrial environment, the representation of those models is accessible so that they can be uploaded into in-memory databases. The available libraries within in-memory databases possess advanced graph algorithms and data analytics functions. The new business processes are designed, typically by agile or lean methods, to roll out rapidly. Then, the new business or dynamically changed business processes can be placed into this environment, and model checking can be efficiently and effectively executed through exploiting the available function of the in-memory database development environment.

The future research question is how the differing representation described in the paper can be integrated at a higher level. The Virtual Twin approach provides a viable and feasible approach, since an apt architecture for a Virtual Twin configuration can be placed in the testing environment of Information Systems that is prescribed in the disciplined system development methods [37]. Despite that fact, that our approach uses the mathematical basis of a hyper-graph and graph theories more generally in the interpretation, explanation, and transparency of issues requires disparate specific methods, algorithms, and procedures. The proposed representation and encoding of the Information System model make it possible to apply further mathematical analysis tools for problem detection and optimization, as well as the exploration of "critical" constituents (business processes, tasks), etc. The augmentation of the services of this representation requires further encoding of the properties of the Information System Model and also requires experimental design, development, and measurement of the effectiveness of various possible approaches to be carried out.The uniform basis of representation, however, offers the chance for integration among the various aspects and distinct algorithmic approaches.

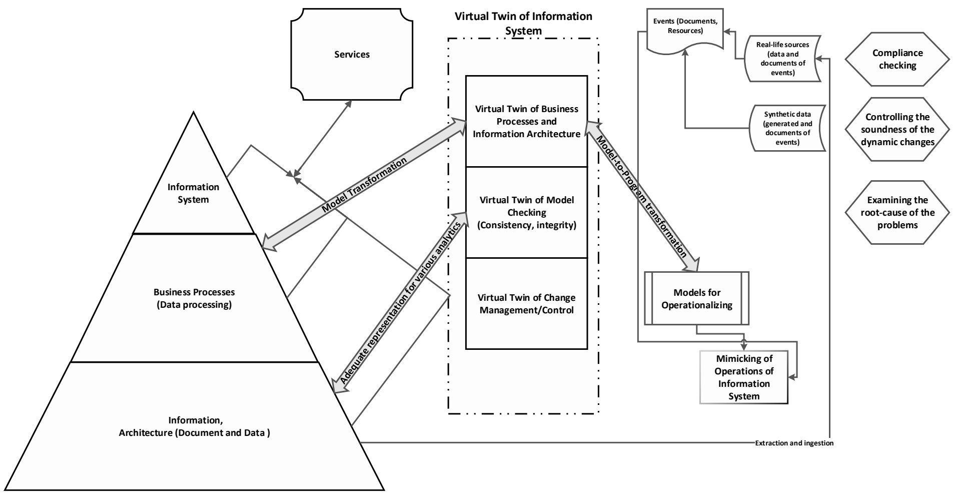

Services
Virtual Twin of Information System
Events (Documents, Resources)
Real-life sources (data and documents of events)
Compliance checking
Synthetic data (generated and documents of events)
Controlling the soundness of the dynamic changes
Examining the root-cause of the problems
Virtual Twin of Business Processes and Information Architecture
Virtual Twin of Model Checking (Consistency, integrity)
Virtual Twin of Change Management/Control
Model Transformation
Model-to-Program transformation
Adequate representation for various analytics
Information System
Business Processes (Data processing)
Information, Architecture (Document and Data)
Models for Operationalizing
Mimicking of Operations of Information System
Extraction and ingestion

Figure 17. The Virtual Twin Framework for Mimicking the Behavior of Information Systems.

7. Materials and Methods

Our research group developed methods and customized algorithms that can interpret and analyze specific aspects of the models in Information Systems. The programs and codes can be accessed at sites that are listed below:

— Hyper-graph visualization; source code of implementation:
  - https://github.com/stsoor/MScFinalProject (accessed on 30 Augest 2021).
  - https://github.com/stsoor/MScThesis(accessed on 30 Augest 2021).

— Clustering Business Processes for Evaluation of Similarities and Dissimilarities:
  - https://github.com/KumundzhievMaxim/ELTE-EFOP-2020(accessed on 25 July 2021).

— Operationalized transformation of UML AD into YAWL:
  – https://github.com/ahmadmukashaty/An-Operationalized-M2P-Transformation-for-Activity-Diagram-into-YAWL (accessed on 26 January 2022).

8. Conclusions

The described and proposed combination of methods, algorithms, and checking mechanisms can provide a viable and feasible approach even in a dynamically changing organizational environment. The transformation and model-checking methods can be exploited in agile project and organization development environments. The methods can be used when a well-defined part of an Information System needs dynamic alteration, maintenance, or further and incremental development. When the task is to check the consistency and integrity of the system which undergoes modifications, the proposed approach proves very useful. Furthermore, when the soundness of the particular aspects of the Information System, e.g., behavior, functional, and information facets and models, should be analyzed, the proposed method is fruitful. The encoding of the Information System model by the proposed method is not only a representation tool, but is also a tool for profound analysis. This mathematical structure, which is based on hyper-graphs and exploits several properties of the generalized hyper-graphs, yields a foundation for rich analytical methods. The various transformations into distinct representations are necessary because of the different capabilities of the specific single representation solutions to be interpreted. This is similar to other mathematical disciplines, where adequate representation that can be applied to gain insights into problems to be solved should be sought.

Author Contributions: B.M. and A.B. worked on the conceptualization of the raised issue; they wrote the original draft version, then they carried out the editing and revision. B.M. and A.B. proofread the draft and revision. B.M. supervised the process. B.M. acquired funding to support the creation of the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by grants of "Application Domain Specific Highly Reliable IT Solutions" project that has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research was supported by grants of "Application Domain Specific Highly Reliable IT Solutions" project that has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Article specific	
IS	Information System
IT	Information Technology
IT/IS	Information Technology and Information System
ERP	Enterprise Resource-Planning System
ITIL	Information Technology Infrastructure Library
TOGAF	The Open Group Architecture Framework
ASC	Abstract Simplicial Complex
XML	Extensible Markup Language

JSON	JavaScript Object Notation
DBMS	Database Management System
BPMN	Business Process modeling Notation standard version 2.0
CNF	Conjunctive Normal Form
DNF	Disjunctive Normal Form
DTD	Document Type Definition
GYO	non-deterministic algorithm described by Graham, Yu, and Özsoyoglu
DM	DM-reduction algortihm originates from D'Atri and Moscarini
DOM	Document Object Model

References

1. Bertalanffy, L.V. *General System Theory: Foundations, Development, Applications*, revised ed.; George Braziller, Inc.: New York, NY, USA, 2015.
2. Alter, S. Defining information systems as work systems: Implications for the IS field. *Eur. J. Inf. Syst.* **2008**, *17*, 448–469. [CrossRef]
3. Cardoso, J.; Fromm, H.; Nickel, S.; Satzger, G.; Studer, R.; Weinhardt, C. *Fundamentals of Service Systems*; Springer: Berlin/Heidelberg, Germany, 2015.
4. Demirkan, H.; Spohrer, J.C.; Krishna, V. *The Science of Service Systems*; Springer: Berlin/Heidelberg, Germany, 2011.
5. Mattyasovszky-Philipp, D.; Molnár, B. Adaptive/cognitive Resonance and the Architecture Issues of Cognitive Information Systems. In Proceedings of the 2020 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Mariehamn, Finland, 23–25 September 2020; IEEE: New York, NY, USA, 2020; pp. 000479–000484.
6. Molnár, B.; Mattyasovszky-Philipp, D. Cognitive Information Systems-artificial Intelligence & Management Decisions. In Proceedings of the 12th IADIS International Conference Information Systems, Vilnius, Lithuania, 10–12 October 2019; Volume 2019, pp. 290–294.
7. Spohrer, J.; Maglio, P.P.; Bailey, J.; Gruhl, D. Steps toward a Science of Service Systems. *Computer* **2007**, *40*, 71–77. [CrossRef]
8. Mattyasovszky-Philipp, D.; Molnár, B. Cognitive Resonance and the Architecture Issues of Cognitive Information Systems. In *Accentuate Innovations in Cognitive Info-Communication in Series: Topics in Intelligent Engineering and Informatics Infocommunications and Human Centred Engineering (Working Title)*; Springer: Berlin/Heidelberg, Germany, 2021; in press.
9. Molnár, B.; Mattyasovszky-Philipp, D.A. An Architectural Approach to Cognitive Information System. In Proceedings of the 10th IEEE International Conference on Cognitive Infocommunications, Naples, Italy, 23–25 October 2019; IEEE: Manhattan, NY, USA, 2019; pp. 459–462.
10. Ogiela, L.D. *Cognitive Information Systems in Management Sciences*; Academic Press: Boston, MA, USA, 2017.
11. Open Group. Togaf: The Open Group Architecture Framework, Version 9.2. 2010. Available online: http://www.opengroup.org/togaf/ (accessed on 26 January 2022).
12. Zachman, J.A. A Framework for Information Systems Architecture. *IBM Syst. J.* **1987**, *26*, 276–292. [CrossRef]
13. CCTA. (Ed.) *Ssadm Version 4 Reference Manuals (Volumes 1–4)*; NCC Blackwell: Oxford, UK, 1990.
14. Skidmore, S.; Farmer, R.; Mills, G. *Ssadm Models and Methods, Version 4*; Blackwell Pub.: Oxford, UK, 1992.
15. Duncan, J. *Ssadm in Practice: A Version 4 Text*; Macmillan International Higher Education: London, UK, 1995.
16. Molnár, B.; Őri, D. Towards a Hypergraph-based Formalism for Enterprise Architecture Representation to Lead Digital Transformation. In *European Conference on Advances in Databases and Information Systems*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 364–376.
17. Őri, D.; Molnár, B. A Hypergraph Based Formal Description Technique for Enterprise Architecture Representation. In Proceedings of the 2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI), Yonago, Japan, 8–13 July 2018; IEEE: New York, NY, USA, 2018; pp. 799–804.
18. Kozák, M.; Stárka, J.; Mlýnková, I. Schematron Schema Inference. In *Proceedings of the 16th International Database Engineering & Applications Sysmposium*; Association for Computing Machinery (ACM): New York, NY, USA, 2012; pp. 42–50.
19. Molnár, B.; Béleczki, A.; Sarkadi-Nagy, B. Storing Hypergraph-based Data Models in Non-hypergraph Data Storage and Applications for Information Systems. *Vietnam. J. Comput. Sci.* **2021**, *8*, 375–395. [CrossRef]
20. Krogstie, J.; Opdahl, A.L.; Brinkkemper, S. *Conceptual Modeling in Information Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2007.
21. Evans, A.; Sammut, P.; Willans, J.S. *Metamodeling for Mda*; First International Workshop: York, UK, 2003.
22. Zhang, Z. *Big Data Mining for Climate Change*; Elsevier: Amsterdam, The Netherlands, 2020.
23. Aubin, J.P.; Désilles, A. *Traffic Networks as Information Systems*; Springer GmbH: Berlin/Heidelberg, Germany, 2016.
24. Blokdijk, A.; Blokdijk, P. *Planning and Design of Information Systems*; Academic Press: London, UK, 1994.
25. Kipling, R. *Just So Stories for Little Children*; Oxford Paperbacks: Oxford, UK 1998. Available online: https://amzn.to/34yuKka (accessed on 26 January 2022).
26. Flood, R.L. I Keep Six Honest Serving Men: They Taught Me All I Knew. *Syst. Dyn. Rev.* **1994**, *10*, 231–243. [CrossRef]
27. Molnár, B.; Benczúr, A. Issues of Modeling Web Information Systems Proposal for a Document-centric Approach. *Procedia Technol.* **2013**, *9*, 340–350. [CrossRef]

28. Gewertz, M. *Defining Enterprise: A Systems View of Capability Management*; Marc H. Gewertz. 2016. Available online: https://www.eabooks.net/ (accessed on 26 January 2022).
29. Olivier Curé, G.B. *Rdf Database Systems: Triples Storage and Sparql Query Processing*; MORGAN KAUFMANN PUBL INC.: Burlington, MA, USA, 2014.
30. Molnár, B.; Béleczki, A.; Benczúr, A. Information Systems modeling Based on Graph-theoretic Background. *J. Inf. Telecommun.* **2017**, *2*, 68–90. [CrossRef]
31. Sawyer, S.; Crowston, K.; Wigand, R.T. Digital assemblages: Evidence and theorising from the computerisation of the US residential real estate industry. *New Technol. Work. Employ.* **2014**, *29*, 40–56. [CrossRef]
32. Sassen, S. *Territory, Authority, Rights: From Medieval to Global Assemblages*; PRINCETON UNIV PR: Princeton, NJ, USA 2008.
33. Latham, R. *Digital formations: IT and New Architectures in the Global Realm*; Princeton University Press: Princeton, NJ, USA, 2005.
34. Shanahan, J. *Soft Computing for Knowledge Discovery: Introducing Cartesian Granule Features*; Springer: Boston, MA, USA, 2000.
35. Savić, M.; Ivanović, M.; Jain, L.C. *Complex Networks in Software, Knowledge, and Social Systems*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019. [CrossRef]
36. Abiteboul, S.; Hull, R.; Vianu, V. *Foundations of Databases*; Addison-Wesley: Singapore, 1995.
37. Cohrs, M. *Ein Architekturmodel Für SAP®-Anwendungen (An Architecture Model for SAP® Applications)*; Vieweg + Teubner Verlag: Berlin, Germany, 2011.
38. Fortnow, L. Fifty years of P vs. NP and the possibility of the impossible. *Commun. ACM* **2022**, *65*, 76–85. [CrossRef]
39. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Morgan Kaufmann Publishers: San Mateo, CA, USA, 1988.
40. Maier, D. *Theory Relational Databases*; Computer Science Press: Rockville, MD, USA, 1983.
41. Gottlob, G.; Pichler, R.; Wei, F. Tractable database design through bounded treewidth. In Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems—PODS, Chicago, IL, USA, 26–28 June 2006; ACM Press: New York, NY, USA, 2006. [CrossRef]
42. Marini, J. *Document Object Model*; McGraw-Hill, Inc.: New York, NY, USA, 2002.
43. Friesen, J. *Java XML and JSON*; Apress: Berlin/Heidelberg, Germany, 2019. [CrossRef]
44. Bretto, A. Applications of Hypergraph Theory: A Brief Overview. In *Hypergraph Theory*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 111–116.
45. Iordanov, B. HyperGraphDB: A Generalized Graph Database. In *Web-Age Information Management—WAIM 2010 International Workshops: IWGD 2010, XMLDM 2010, WCMT 2010, Jiuzhaigou Valley, China, 15–17 July 2010, Revised Selected Papers*; Shen, H.T., Pei, J., Özsu, M.T., Zou, L., Lu, J., Ling, T.W., Yu, G., Zhuang, Y., Shao, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2010, Volume 6185, pp. 25–36. [CrossRef]
46. Kobrix Software. Hypergraphdb—A Graph Database. 2010. Available online: http://hypergraphdb.org (accessed on 30 January 2022).
47. Michael, R.; Marcus, P.; Christof, B.; Wolfgang, L. The Graph Story of the SAP HANA Database. In *Datenbanksysteme für Business, Technologie und Web (BTW), 15. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 11.-15.3.2013 in Magdeburg, Germany. Proceedings*; Markl, V., Saake, G., Sattler, K., Hackenbroich, G., Mitschang, B.; Härder, T., Köppen, V., Eds.; Gesellschaft für Informatik e.V.: Bonn, Germany, 2013; Volume P-214, pp. 403–420.
48. Newman, M. The Smith normal form. *Linear Algebra Its Appl.* **1997**, *254*, 367–381. [CrossRef]
49. Dummit, D.; Foote, R. *Abstract Algebra*, 3rd ed.; Wiley: New York, NY, USA, 2004; p. xii + 932.
50. Scheinerman, E.R.; Ullman, D.H. *Fractional Graph Theory*; A Wiley-Interscience publication; Wiley: New York, NY, USA, 1997.
51. Ferrario, D. A Simple Algorithm for Computing the Smith Normal Form of a Matrix in Z. 2016. Available online: https://www.dlfer.xyz/post/2016-10-27-smith-normal-form/ (accessed on 8 June 2021).
52. Purvine, E.; Aksoy, S.; Joslyn, C.; Nowak, K.; Praggastis, B.; Robinson, M. A Topological Approach to Representational Data Models. In *Human Interface and the Management of Information. Interaction, Visualization, and Analytics*; Chapter A Topological Approach to Representational Data Models; Yamamoto, S., Mori, H., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; Volume 10904, pp. 90–109. [CrossRef]
53. Vick, J.W. *Homology Theory: An Introduction to Algebraic Topology*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 145,
54. Kahle, M. Topology of random clique complexes. *Discret. Math.* **2009**, *309*, 1658–1671. [CrossRef]
55. Praggastis, B.; Arendt, D.; Joslyn, C.; Purvine, E.; Aksoy, S.; Monson, K. PNNL HyperNetX. 2020. Available online: https://pnnl.github.io/HyperNetX/build/index.html (accessed on 20 March 2021).
56. Earl, J. Computing Homology of Hypergraphs. 2019. Available online: https://digitalcommons.calpoly.edu/star/561 (accessed on 30 January 2022).
57. Dumas, J.G.; Heckenbach, F.; Saunders, D.; Welker, V. Computing Simplicial Homology Based on Efficient Smith Normal Form Algorithms. In *Algebra, Geometry and Software Systems*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 177–206. [CrossRef]
58. Robinson, I. *Graph Databases*; O'Reilly Media, Inc., O'Reilly Media: Sebastopol, CA, USA, 2013.
59. Deka, G.C. (Ed.) *NoSQL*; Taylor & Francis Ltd.: Abingdon, UK, 2017.
60. Brien, P.M.; Poulovassilis, A. A Semantic Approach to Integrating Xml and Structured Data Sources. In *Advanced Information Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 330–345.

61. Meier, A.; Kaufmann, M. *SQL & NoSQL Databases*; Springer Fachmedien Wiesbaden: Berlin/Heidelberg, Germany, 2019. doi: [CrossRef]
62. Dietrich, S.W.; Urban, S.D. *Fundamentals of Object Databases. Object-Oriented and Object-Relational Design*; Morgan & Claypool Publishers: San Rafael, CA, USA, 2010; Volume 12, p. xxi + 151. [CrossRef]
63. Merunka, V.; Molhanec, M. Object Normalization as Contribution to the area of Formal Methods of Object-Oriented Database Design. In *Advances in Computer and Information Sciences and Engineering*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 300–304. [CrossRef]
64. Libkin, L. Normalization Theory for XML. In *Database and XMLTechnologies*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1–13. [CrossRef]
65. Lv, T.; Yan, P. XML Normal Forms Based on Constraint-Tree-Based Functional Dependencies. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 348–357. [CrossRef]
66. Fischer, L. *BPMN 2.0 Handbook Second Edition: Methods, Concepts, Case Studies and Standards in Business Process Management Notation*; Future Strategies: Lighthouse Point, FL, USA, 2012.
67. White, S. *BPMN Modeling and Reference Guide: Understanding and Using BPMN: Develop Rigorous Yet Understandable Graphical Representations of Business Processes*; Future Strategies Inc.: Lighthouse Point, FL, USA, 2008.
68. Reisig, W. *Understanding Petri Nets*; Springer: Berlin/Heidelberg, Germany, 2013. [CrossRef]
69. Davis, R. The Event-Driven Process Chain. In *Business Process Modeling with ARIS: A Practical Guide*; Springer: London, UK, 2001; pp. 111–139. [CrossRef]
70. Singer, R.; Teller, M. Process Algebra and the Subject-Oriented Business Process Management Approach. In *S-BPM ONE-Education and Industrial Developments*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 135–150. [CrossRef]
71. Wong, P.Y.; Gibbons, J. Formalisations and applications of BPMN. *Sci. Comput. Program.* **2011**, *76*, 633–650. [CrossRef]
72. Jensen, K.; Rozenberg, G. (Eds.) *High-Level Petri Nets*; Springer: Berlin/Heidelberg, Germany, 2012. [CrossRef]
73. Mutarraf, U.; Barkaoui, K.; Li, Z.; Wu, N.; Qu, T. Transformation of Business Process Model and Notation models onto Petri nets and their analysis. *Adv. Mech. Eng.* **2018**, *10*, 168781401880817. [CrossRef]
74. Eshuis, R.; Wieringa, R. Verification support for workflow design with UML activity graphs. In Proceedings of the 24th International Conference on Software Engineering, Orlando, FL, USA , 19–25 May 2002; ACM Press: New York, NY, USA, 2002; pp. 166–176. [CrossRef]
75. Larman, C. *Applying Uml and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3/e*; Pearson Education India: New Delhi, India, 2012.
76. Scheer, A.W.; Thomas, O.; Adam, O. Process Modeling using Event-Driven Process Chains. In *Process-Aware Information Systems*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2005; pp. 119–145. [CrossRef]
77. Bouafia, K.; Molnár, B. Formal Verification of Analysis Approach for Enterprise Information Systems Architecture Using Hypergraph Representation Based on Finite State Machines for Supporting Business Process Requirements. *J. Appl. Bus. Econ. (JABE)* **2020**, *22*, 265. [CrossRef]
78. Saligny, L.; Bouillé, F. *La Méthode Hbds: Hypergraph-based Data Structure*; *Information Spatiale Et Archéologie*; 2011; pp. 62–65. Available online: https://halshs.archives-ouvertes.fr/halshs-00959477 (accessed on 26 January 2022).
79. Sun, L.; Ji, S.; Ye, J. Hypergraph Spectral Learning for Multi-label Classification. In Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 668–676. [CrossRef]
80. Ducournau, A. Hypergraphes: Clustering, Réduction Et Marches Aléatoires Orientées Pour La Segmentation D'images Et De Vidéo. Ph.D. Thesis, Ecole Nationale d'ingénieurs, Saint-Etienne, France, 2012.
81. Rital, S. Hypergraphe De Voisinage Spatiocolorimétrique: Application En Traitement D'images. Ph.D. Thesis, Université de Bourgogne, Dijon, France, 2004.
82. Tian, Z.; Hwang, T.; Kuang, R. A Hypergraph-Based Learning Algorithm for Classifying Gene Expression and arrayCGH Data with Prior Knowledge. *Bioinformatics* **2009**, *25*, 2831–2838. [CrossRef]
83. Koppen, E.; Neumann, G. Active Hypertext for Distributed Web Applications. In Proceedings of the IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'99). *IEEE Comput. Soc.* **1999**, 297–302. [CrossRef]
84. Atzeni, P.; Merialdo, P.; Mecca, G. Data-intensive Web Sites: Design and Maintenance. *World Wide Web* **2001**, *4*, 21–47. [CrossRef]
85. Rossi, G.; Schwabe, D.; Lyardet, F. Web Application Models Are More Than Conceptual Models. In *International Conference on Conceptual Modeling*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 239–252.
86. Erl, T. *Service-Oriented Architecture: Concepts, Technology, and Design*; Pearson Education India: New Delhi, India, 2005.
87. Erl, T. *Service-Oriented Architecture*; Pearson Education: London, UK, 2017.
88. Wilde, E.; Pautasso, C. (Eds.) *REST: From Research to Practice*; Springer: New York, NY, USA, 2011. [CrossRef]
89. Chinnici, R.; Moreau, J.J.; Ryman, A.; Weerawarana, S. Web Services Description Language (wsdl) Version 2.0 Part 1: Core Language. *W3C Recomm.* **2007**, *26*, 19.
90. MacKenzie, C.M.; Laskey, K.; McCabe, F.; Brown, P.F.; Metz, R.; Hamilton, B.A. Reference Model for Service Oriented Architecture 1.0. Oasis Standard. 2006. Available online: http://angeldeacero.wdfiles.com/local--files/start/oasissoa.pdf (accessed on 26 January 2022).

91. Bernauer, M.; Schrefl, M. Self-maintaining Web Pages: From Theory to Practice. *Data Knowl. Eng.* **2004**, *48*, 39–73. [CrossRef]
92. Chiu, C.M.; Bieber, M. A Dynamically Mapped Open Hypermedia System Framework for Integrating Information Systems. *Inf. Softw. Technol.* **2001**, *43*, 75–86. [CrossRef]
93. Nam, C.K.; Jang, G.S.; Bae, J.H.J. An Xml-based Active Document for Intelligent Web Applications. *Expert Syst. Appl.* **2003**, *25*, 165–176. [CrossRef]
94. Molnár, B.; Benczúr, A.; Tarcsi, Á. Formal Approach to a Web Information System Based on Story Algebra. *Singidunum J. Appl. Sci.* **2012**, *9*, 63–73. [CrossRef]
95. Suh, N.P. *Axiomatic Design: Advantages and Applications*; Oxford University Press: New York, NY, USA, 2001.
96. Broekstra, J.; Kampman, A.; van Harmelen, F. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *The Semantic Web — ISWC 2002*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 54–68. [CrossRef]
97. Šmite, D.; Moe, N.B.; Ågerfalk, P.J. (Eds.) *Agility Across Time and Space*; Springer: Berlin/Heidelberg, Germany, 2010. [CrossRef]
98. Lankhorst, M. (Ed.) *Agile Service Development*; Springer: Berlin/Heidelberg, Germany, 2012. [CrossRef]
99. Berliner BPM-Offensive. BPMN 2.0 Poster—Business Process Model and Notation. Available online: http://www.bpmb.de/index.php/BPMNPoster (accessed on 12 September 2021).
100. Emam, K.E.; Mosquera, L.; Hoptroff, R. *Practical Synthetic Data Generation*; O'Reilly Media, Inc.: New York, NY, USA, 2020.
101. Edwards, H.M. *Linear Algebra*; Birkhäuser: Boston, MA, USA, 2013.
102. David, S.; Dummit, R.M.F. *Abstract Algebra*; WILEY: Hoboken, NJ, USA, 2003.
103. A Tool for Computing the Smith Normal Forms over Arbitrary Principle Ideal Domains. Available online: https://pypi.org/project/smithnormalform/ (accessed on 26 September 2021).
104. Peltier, S.; Alayrangues, S.; Fuchs, L.; Lachaud, J.O. Computation of homology groups and generators. *Comput. Graph.* **2005**, *30*, 62–69. [CrossRef]
105. Agoston, M. *Algebraic Topology: A First Course*; M. Dekker: New York, NY, USA, 1976.
106. Smith, H.J.S. Arithmetical notes. *Proc. Lond. Math. Soc.* **1976**, *4*, 236–253. [CrossRef]
107. Bouafia, K.; Molnár, B. Hypergraph Application on Business Process Performance. *Information* **2021**, *12*, 370. [CrossRef]
108. Geroimenko, V. *Dictionary of XML Technologies and the Semantic Web*; Springer: London, UK, 2013.
109. van der Aalst, W.; ter Hofstede, A.; Kiepuszewski, B.; Barros, A. Workflow Patterns. *Distrib. Parallel Databases* **2003**, *14*, 5–51. [CrossRef]
110. Fischer, L. *Workflow Handbook*, 2nd ed.; Future Strategies: Lighthouse Point, FL, USA, 2002.
111. Molnár, B.; Benczúr, A. Facet of Modeling Web Information Systems from a Document-centric View. *Int. J. Web Portals (IJWP)* **2013**, *5*, 57–70. [CrossRef]
112. Thompson, S. *Type Theory and Functional Programming*; Addison-Wesley: Wokingham, UK, 1991.
113. Sharvit, Y. *Data-Oriented Programming: Unlearning Objects*; MANNING PUBN: New York, NY, USA, 2022.
114. Perry, M.L. *The Art of Immutable Architecture*; Apress: New York, NY, USA, 2020. [CrossRef]
115. Lazuashvili, N.; Norta, A.; Draheim, D. Integration of Blockchain Technology into a Land Registration System for Immutable Traceability: A Casestudy of Georgia. In *Business Process Management: Blockchain and Central and Eastern Europe Forum*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 219–233. [CrossRef]
116. Malone, T.W.; Crowston, K.; Herman, G.A. *Organizing Business Knowledge: The MIT Process Handbook*; The MIT Press: Cambridge, MA, USA, 2003.
117. Russell, N.; van der Aalst, W.M.P.; ter Hofstede, A.H. *Workflow Patterns*; MIT Press Ltd.: Cambridge, MA, USA, 2016.
118. Fischer, L. *Workflow Handbook*, 1st ed.; Future Strategies: Lighthouse Point, FL, USA, 2000.
119. Kiepuszewski, B.; ter Hofstede, A.H.M.; Bussler, C.J. On Structured Workflow modeling. In *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*; Springer International Publishing: Berlin/Heidelberg, Germany, 2000; pp. 431–445. [CrossRef]
120. Dehnert, J.; Rittgen, P. Relaxed Soundness of Business Processes. In *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*; Springer International Publishing: Berlin/Heidelberg, Germany, 2001; pp. 157–170. [CrossRef]
121. Ye, Y.; Roy, K. Efficient synthesis of AND/XOR networks. In Proceedings of the ASP-DAC'97: Asia and South Pacific Design Automation Conference, Chiba, Japan, 28–31 January 1997; IEEE: New York, NY, USA, 1997.
122. Thakur, M.; Tripathi, R. Linear connectivity problems in directed hypergraphs. *Theor. Comput. Sci.* **2009**, *410*, 2592–2618. [CrossRef]
123. Chandra, A.K.; Harel, D. Horn clause queries and generalizations. *J. Log. Program.* **1985**, *2*, 1–15. [CrossRef]
124. Kowalski, R.A. Predicate Logic as Programming Language. Information Processing. In Proceedings of the 6th IFIP Congress 1974, Stockholm, Sweden, 5–10 August 1974; Rosenfeld, J.L., Ed.; North-Holland: New York, NY, USA, 1974; pp. 569–574.
125. Ligeza, A. *Logical Foundations for Rule-Based Systems*; Springer GmbH: Berlin/Heidelberg, Germany, 2006.
126. Gammaitoni, L.; Kelsen, P. F-Alloy: A relational model transformation language based on Alloy. *Softw. Syst. Model.* **2019**, *18*, 213–247. [CrossRef]
127. Gammaitoni, L.; Kelsen, P. F-alloy: An alloy based model transformation language. In *International Conference on Theory and Practice of Model Transformations*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 166–180. [CrossRef]
128. Jackson, D. Alloy: A language and tool for exploring software designs. *Commun. ACM* **2019**, *62*, 66–76. [CrossRef]
129. Armstrong, M. *A Handbook of Human Resource Management Practice*; Kogan Page: London, UK, 2006.
130. Ullman, J.D. *A First Course in Database Systems*; Pearson Education India: Upper Saddle River, NJ, USA, 2007.

131. Klug, A.; Price, R. Determining View dependencies using tableaux. *ACM Trans. Database Syst.* **1982**, *7*, 361–380. [CrossRef]
132. Curran, T. *SAP R/3 Business Blueprint: Understanding Enterprise Supply Chain Management*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2000.
133. Garcia-Molina, H.; Salem, K. Sagas. *ACM SIGMOD Rec.* **1987**, *16*, 249–259. [CrossRef]
134. Liu, L.; Özsu, M.T. (Eds.) *Web Services Business Process Execution Language*; Encyclopedia of Database Systems; Springer: Boston, MA, USA, 2009. [CrossRef]
135. Eiter, T.; Gottlob, G. Hypergraph Transversal Computation and Related Problems in Logic and AI. In *Logics in Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 549–564. [CrossRef]
136. Bailey, J.; Manoukian, T.; Ramamohanarao, K. A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns. Third IEEE International Conference on Data Mining. *IEEE Comput. Soc.* **2003**, [CrossRef]
137. Brault-Baron, J. Hypergraph Acyclicity Revisited. *ACM Comput. Surv.* **2016**, *49*, 1–26. [CrossRef]
138. Marc, H.; Graham. *On the Universal Relation*; Technical report; University of Toronto: Toronto, ON, Canada, 1979.
139. Brault-Baron, J.; Capelli, F.; Mengel, S. Understanding model counting for *beta*-acyclic CNF-formulas. *arXiv* **2014**, arXiv:1405.6043.
140. D'Atri, A.; Moscarini, M. On the recognition and design of acyclic databases. In *Proceedings of the 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems-PODS*; ACM Press: New York, NY, USA, 1984. [CrossRef]
141. Tarjan, R.E.; Yannakakis, M. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. *SIAM J. Comput.* **1984**, *13*, 566–579. [CrossRef]
142. Yu, C.; Ozsoyoglu, M. An algorithm for tree-query membership of a distributed query. In Proceedings of the COMPSAC 79 Computer Software and The IEEE Computer Society's Third International Applications Conference, Chicago, IL, USA, 6–8 November 1979; IEEE: New York, NY, USA, 1979; pp. 306–312. [CrossRef]
143. Ahituv, N.; Neumann, S.; Zviran, M. A System Development Methodology for ERP Systems. *J. Comput. Inf. Syst.* **2002**, *42*, 56–67. [CrossRef]
144. Yamakawa, P.; Noriega, C.O.; Linares, A.N.; Ramírez, W.V. Improving ITIL compliance using change management practices: A finance sector case study. *Bus. Process Manag. J.* **2012**, *18*, 1020–1035. [CrossRef]
145. Kherbouche, M.; Bouafia, K.; Molnár, B. Transformation of Uml State Machine to Yawl. In Proceedings of the Ninth IEEE International Conference on Intelligent Computing and Information Systems, Washington, DC, USA, 11–14 October 2009. [CrossRef]
146. Kherbouche, M.; Mukashaty, A.A.; Molnár, B. An Operationalized Transformation for Activity Diagram into YAWL. In *Developments in Computer Science, 17–19 June 2021, ELTE, Hungary*; Csuhaj Varjú, E., Ed.; Eötvös Loránd University of Budapest, Faculty of Informatics, Eötvös Loránd University of Budapest, Faculty of Informatics: Budapest, Hungary,2021. Available online: http://dcs.elte.hu/wp-content/uploads/2022/01/DCS_proceedings.pdf (accessed on 30 January 2022).
147. Hofstede, A.H.M.; Aalst, W.M.P.; Adams, M.; Russell, N. (Eds.) *Modern Business Process Automation*; Springer: Berlin/Heidelberg, Germany, 2010. [CrossRef]
148. Kherbouche, M.; Molnár, B. Formal Model Checking and Transformations of Models Represented in UML with Alloy. In *Modeling to Program*; Thalheim, B., Ed.; Springer International Publishing: Berlin/Heidelberg, Germany, 2021; pp. 127–136. [CrossRef]
149. Alrabbaa, C.; Baader, F.; Borgwardt, S.; Koopmann, P.; Kovtunova, A. Finding Good Proofs for Description Logic Entailments using Recursive Quality Measures. In *Automated Deduction—CADE 28*; Springer International Publishing: Berlin/Heidelberg, Germany, 2021; pp. 291–308. [CrossRef]
150. Amit Basu, R.W.B. *Metagraphs and Their Applications*; Springer GmbH: Berlin/Heidelberg, Germany, 2006.
151. Tao, F. *Digital Twin Driven Smart Manufacturing*; Academic Press: London, UK, 2019.