Volume 4 | Number 4 Article 5

2016

Formal approach to modeling of modern Information Systems

Bálint Molnár
Eötvös Loránd University

András Benczúr
Eötvös Loránd University

András Béleczki
Eötvös Loránd University

International Journal of
Information Systems and Project Management

ISSN (print):2182-7796, ISSN (online):2182-7788, ISSN (cd-rom):2182-780X

Available online at www.sciencesphere.org/ijispm

Formal approach to modeling of modern information systems

Bálint Molnár
Information Systems Department, Eötvös Loránd University
Pázmány Péter sétány 1/C, Budapest, 1117
Hungary
www.shortbio.net/molnarba@inf.elte.hu

András Benczúr
Information Systems Department, Eötvös Loránd University
Pázmány Péter sétány 1/C, Budapest, 1117
Hungary
www.shortbio.net/abenczur@inf.elte.hu

András Béleczki
Information Systems Department, Eötvös Loránd University
Pázmány Péter sétány 1/C, Budapest, 1117
Hungary
www.shortbio.net/bearaai@inf.elte.hu

Abstract:
Most recently, the concept of business documents has started to play double role. On one hand, a business document (word processing text or calculation sheet) can be used as specification tool, on the other hand the business document is an immanent constituent of business processes, thereby essential component of business information systems. The recent tendency is that the majority of documents and their contents within business information systems remain in semi-structured format and a lesser part of documents is transformed into schemas of structured databases. In order to keep the emerging situation in hand, we suggest the creation (1) a *theoretical framework* for modeling business Information Systems and (2) a *design method* for practical application based on the theoretical model that provides the structuring principles. The modeling approach that focuses on *documents* and their interrelationships with business *processes* assists in perceiving the activities of modern information systems.

Keywords:
formal modeling; document centric process and data modeling; information system architecture; information systems modeling.

DOI: 10.12821/ijispm040404

Manuscript received: 20 January 2016
Manuscript accepted: 28 November 2017

1. Introduction

The current Business Information Systems shows new behavioral properties. Namely, the documents, unstructured and semi-structured data, have high relevance beside the structured data. One of the directions within management sciences is the service-orientation. Being the business processes of companies organized by the service-orientation pattern, consequently, the structure of functions within Information Systems follows the model of IT (Information Technology) services, independently from the applied technologies as either Web, REST, Micro Services or other appropriate technology that are based on Services.

These two trends – document and service centric approaches – are slowly modifying the requirements against the modeling methods that are intended to describe the behavior of IS [1], [3]. The documents, interactive documents and the emphasis on Web interfaces led to the concept of modern Information Systems.

In the core of IS there is a set of data that delivers the required information either to the business activities or to the information processes. During analysis, the question that is investigated is as to whether what information should be kept in the system. The data and their collections exist independently from business documents that may or may not related to decision processes. The modeling of Information Systems focused on document should adhere to established practices of data and information modeling. The model should be perceivable by users at high level. The approach for modeling and analysis should be semantically powerful in order to serve as a basic model and be understandable by users. The document-centric model is unlike data models but they are interdependent on each other. The document model tries to gather the results of business activities and the transformations in order to extend information within documents with new facts (Fig. 1).

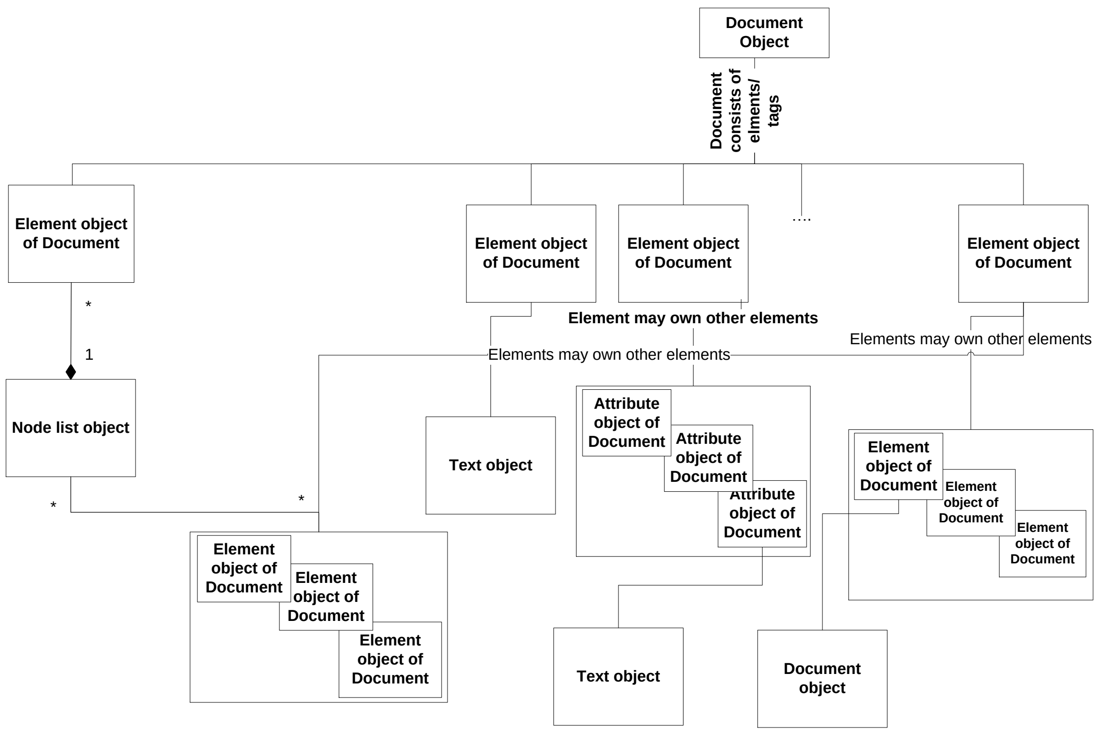
Document Object
Document consists of elments/ tags
Element object of Document
Element object of Document
Element object of Document
....
Element object of Document
*
1
Node list object
*
*
Element may own other elements
Elements may own other elements
Elements may own other elements
Text object
Attribute object of Document
Attribute object of Document
Attribute object of Document
Element object of Document
Element object of Document
Element object of Document
Element object of Document
Element object of Document
Element object of Document
Text object
Document object

Fig. 1. Levels of collections related to documents.

The manipulation of documents happens through business processes; moreover, the document model mirrors the structure of organization and events (Fig. 2).

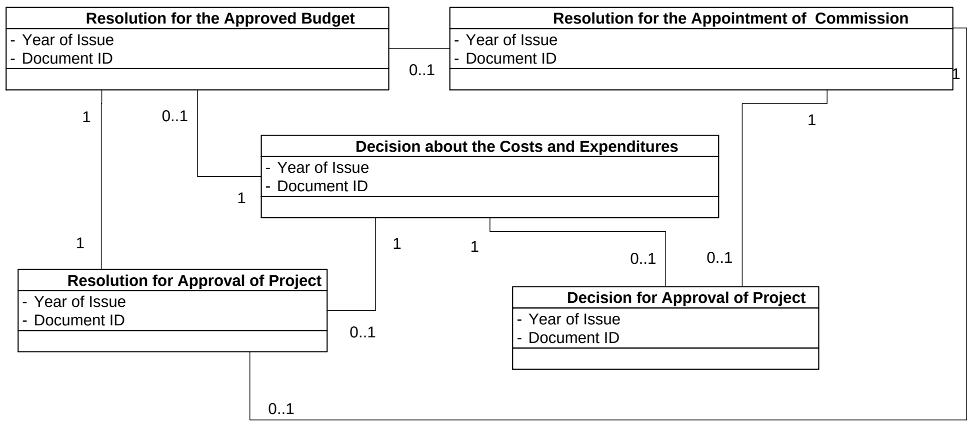
Resolution for the Approved Budget
- Year of Issue
- Document ID
Resolution for the Appointment of Commission
- Year of Issue
- Document ID
Decision about the Costs and Expenditures
- Year of Issue
- Document ID
Resolution for Approval of Project
- Year of Issue
- Document ID
Decision for Approval of Project
- Year of Issue
- Document ID
0..1
1
1
0..1
1
1
1
1
1
0..1
0..1
0..1
0..1

Fig. 2. Example of Class Diagram for Documents at “Document Conceptual Level” within a Case Study.

Within the data model, those alterations that change elements that are significant for business should be tracked, i.e. create new ones, transform existing ones, set up new dependencies or adjust existing relationships. The actor or role that performs the data conversion should be distinguishable during transformation processes; furthermore, the collection of data that are linked to documents and roles should be identifiable as well. In an E-government environment, a case study is planned and designed to verify and validate the results of proposed modeling approach with theoretical background.

In section 2 we present previous research reported in the literature. In section 3 we outline our method, making use of the previous approaches in a document centric approach. In section 4 it is presented the formal mathematical background. Section 5 presents the formalized document centric approach. Section 6 discusses the information architecture and documents. Finally, section 7 provides the summary and the conclusions.

2. Literature Review

Joeris [1] proposed a document based approach for modeling control and data flow for business activities and data interchange among them. Wewers et al. [4] presented a system that supports a framework for inter-organizational, document oriented workflow.

To help the perception of the complex behavior of IS the enterprise architecture approaches offer support, namely the Zachman ontology and TOGAF, both was developed for IS [5], [6], [7].

The artifact-centric business process model uses three basic concepts [8], [9]: artifact classes; tasks; and business rules. The tasks handle the artifacts, the business rules govern which tasks should be triggered and which artifacts will be manipulated [10], [11]. SOA (Service Oriented Architecture) and the related technologies as XML (Extensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) and UDDI (Universal Description, Discovery, and Integration,) permits that services to be available through the Web [12]. The Web documents typically in XML format can be considered the central notion of Information Systems on the Web. There are analysis and design problems that should be improved. Emphasizing the problems with IT rather than business processes hinders the modeling and abstraction of stable and reliable IS [32]. The approaches as *Service Oriented*

Computing (SOC), and *Cloud Computing* concentrates on services as a standardized and general information exchange interface towards users. There are different input data format for interchange between services [13], [14], [15]: (1) HTML pages; (2) SOAP messages (XML); (3) unstructured documents (XML). Unstructured documents may contain text, images, and other binary data, only the metadata may have formalized in XML using tags. Set of documents without uniform XLST (Extensible Stylesheet Language Transformations), DTD (Document Type Definition) or any other “style-sheets”, we consider them as set of unstructured files since there is no general principle that can be enforced on each single document. Unstructured documents are the typical office documents without pre-defined style-sheets for tagging as the meta-data of documents may be tagged but the textual information is not. SOAP messages as XML tagged data can be regarded as structured but it may transport unstructured data.

XML documents can be considered as application-relevant “things”, i.e. they can be perceived as new data objects to be stored and managed by a DBMS. This type of XML documents, in this sense, is document-centric, since their meaning depends on the document as a whole. The XML structure is more irregular in contrast to structured data, and data contained in them are heterogeneous. Chidlovskii [14], [16] provides a formal grammatical description of XML.

The alignment and fitting between business processes and organization can be analyzed on the base of ontologies and semantic approaches [17], [18]. The e-commerce, e-banking, e-tourism, Web-based Enterprise Resource Systems can be considered as typical Information Systems on the WEB.

To combine the previously referred approaches to model modern Information Systems, there are various proposals [2], [3], [19]. Blokdijk’s assembly of Information System Models [20] offers structuring principles; moreover, the axiomatic design approach [21] can be employed for the Information Systems provides clues for both theoretic and practical modeling point of view. The concept of generalized hypergraph [22] seems to be a proper mathematical formalism that fits to unifying all viewpoints, perspectives, artifacts and modeling elements.

3. Document centric approach

On modeling IS, the data model plays a central role traditionally (Fig. 3). To harmonize the traditional data model with document model, we generalize the concept of data models.

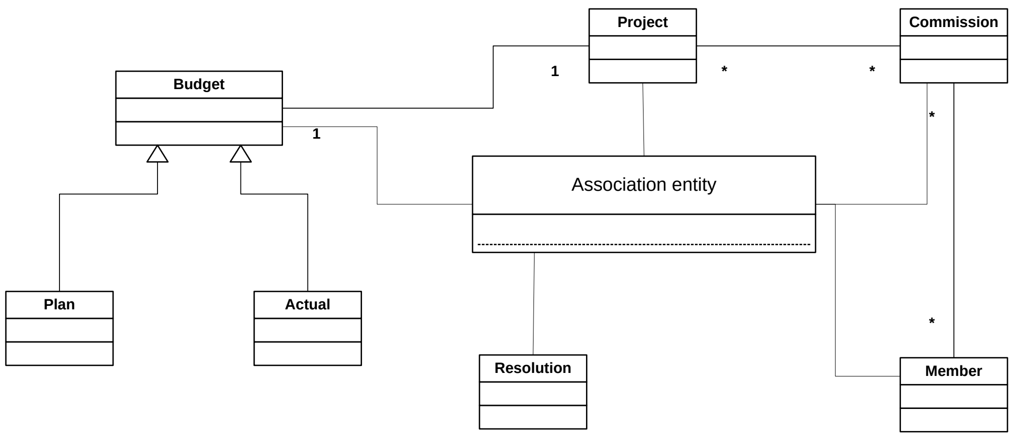

Fig. 3. Class Diagram for Documents Class Diagram for Documents at “Data Model Conceptual Level” within a Case Study.

In this representation, data models consist of **collections** (of data) so that each collection has a designation. The collections are *sets of data* or multi-set (bag) of data or *data types* with well-defined properties and structure; the most typical representation of *data model* is either relational data model or object-relational data model (However, there are several concurrent representation and implementation technology). The extension of data types as occurrences compose subsets of dataset that can be deduced from document structures. The *collections* include identified data that are significant as their modifications over time are linked to documents (but that is not the same as the logging of database activities, in opposite it depicts the impact of activities related to document manipulation).

3.1 The Document-centric Modeling

The proposed approach is unlike to the traditional database modeling methods and the recent fashionable artifact-centered approaches. The document-centric modeling should exist with a strong *correspondence* to the Enterprise Architecture of the given organization, with a definite emphasis on the business processes. The structure of documents within an organization can be mapped onto the organigram and structure of business processes through homomorphism (Fig. 4).

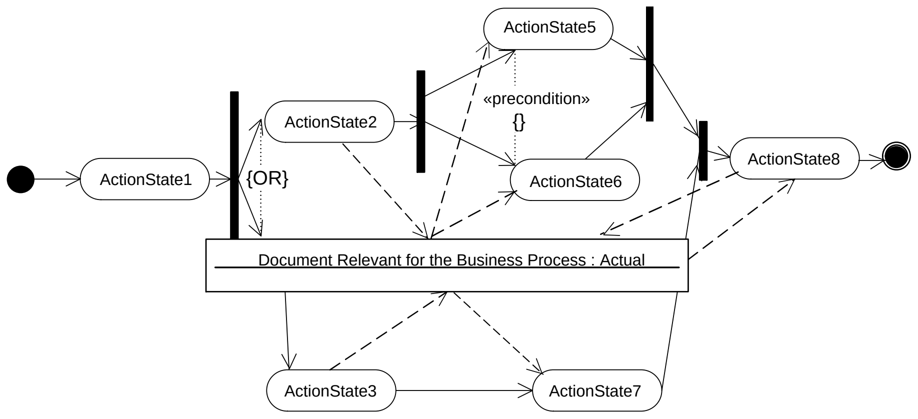

Fig. 4. Business Process Model - A Case.

The representation of both business processes and organization structures appear within Business Process Owner/Manager perspective of Enterprise Architecture [7]. The needs for flexible IS lead to tendencies that can be formulated as a *customer-centric* paradigm. The customer-centric paradigm can be partly captured by a highly flexible document structure at the User Interface level. The documents should be adaptable to changes both in their structure and in their related content. There is correlation between the software architecture and the project structure of the software development [23]. The document model should mirror the life cycle of documents, the manipulation, the events, and effects by business processes. The *modifications* that affect the data included in *documents* should be traced, i.e. creating, modifying data items, establishing new relationship; the *precedence* analyses are an available option for tracking the impacts [20].

The chain of events and processes can be monitored through *roles/actors* and their *handling* of identifiable data items within documents. The human roles stimulate data processing activities that affects documents, consequently the data items as well that are included in the documents. The *document-subdocument* structure (Fig. 1, Fig. 2, Fig. 3) is able to represent both the organization and the information model at the same time. Whilst the data model is not structured as set of *sub-data models* instead it displays rather uniform configuration [20] (Fig. 5).

Blokdijk’s model offers a structuring approach for perceiving Information Systems. The model’s major components are (Fig. 5): (1) **organizational model**; (2) **information model**; (3) **data model**; (4) **process model**. The process model provides the composition of business *activities* and it is strongly coupled to the *control structure*. However, the data model is not an exact representation of the organization structure. For the reason that *patterns of data model* reflect the relevant facts about the organization but it does not map the *organization structures*. The document and data model requires a common representational method in which the services, and functions of documents, the coupled business activities can be depicted in a uniform manner. Furthermore, the interrelationships between the data and document model can be shown as much the same way as possible. There is logic of inheritance to identify data items within documents. Within the document chain, data elements are inherited from the previous element of documents.

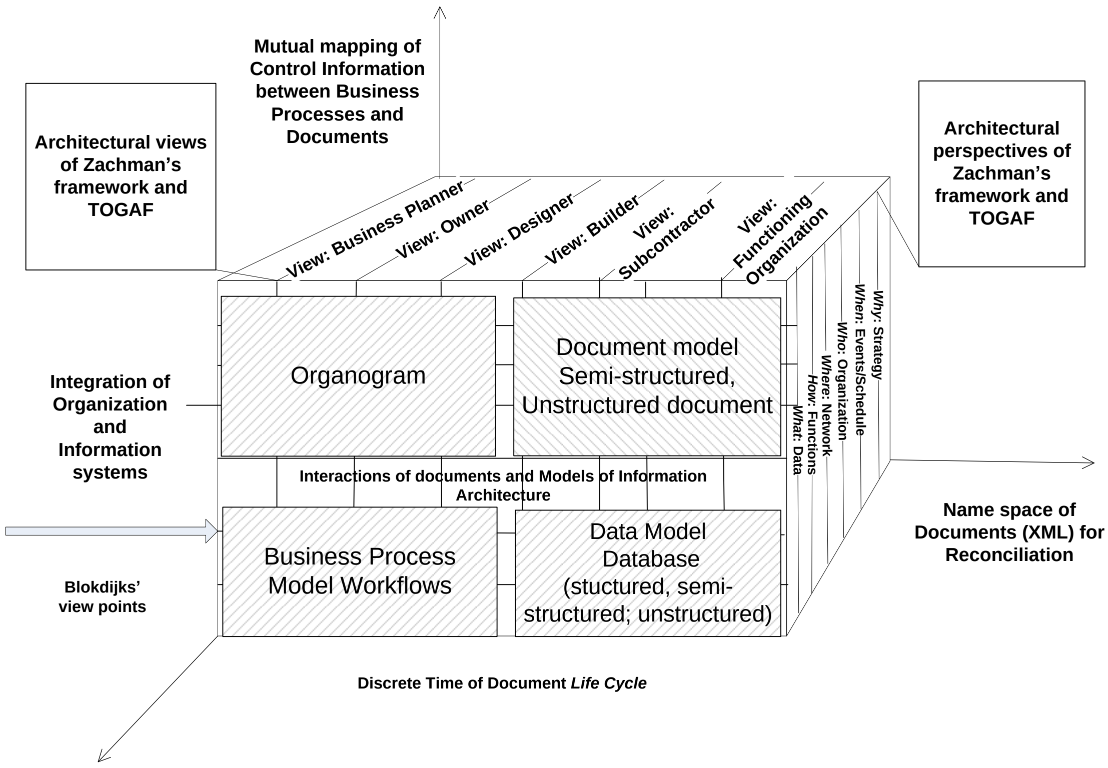
Mutual mapping of Control Information between Business Processes and Documents
Architectural views of Zachman's framework and TOGAF
Architectural perspectives of Zachman's framework and TOGAF
View: Business Planner
View: Owner
View: Designer
View: Builder
View: Subcontractor
View: Functioning Organization
Why: Strategy
When: Events/Schedule
Who: Organization
Where: Network
How: Functions
What: Data
Organogram
Document model Semi-structured, Unstructured document
Integration of Organization and Information systems
Interactions of documents and Models of Information Architecture
Business Process Model Workflows
Data Model Database (stuctured, semi-structured; unstructured)
Name space of Documents (XML) for Reconciliation
Blokdijks' view points
Discrete Time of Document Life Cycle

Fig. 5. A Multi-dimensional model for interaction of Information Systems and Documents.

3.2 Types of Documents

The document model is composed of **document types**. The types of documents designate the state of their variables. We define the concept of ***binding*** by this way: a free field within a document; or a free variable is set for a value, i.e. valuated. The status and the type of documents can be inferred from the bindings; i.e. how many variables are already set to specific values. A **generic document** is a hierarchy of classes of documents. Finalizing or finishing a document instance within a hierarchy of a generic documents leads to that all free variables/fields are set to a certain value step-by-step. The finalization of documents ensues from overarching business processes that can be linked to the flow of documents. The documents flow can be represented by data flow, Event Process Chain or Business Process Modeling Notation.

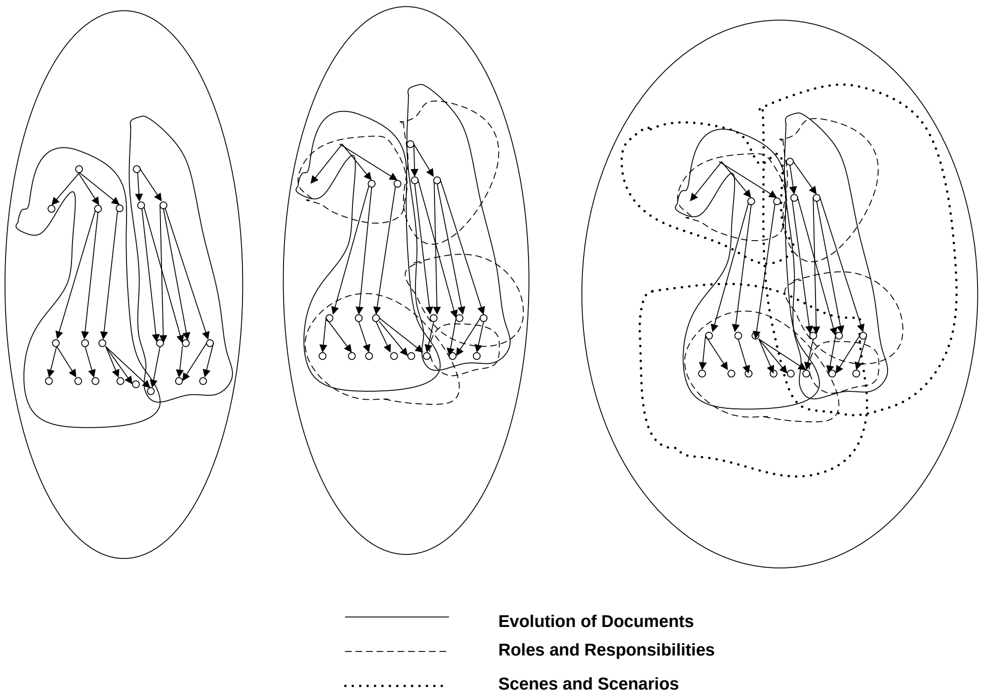

Fig. 6. Representation the Life of Documents by Hypergraph.

The **free-documents** – like free tuples from tableau queries – can be perceived as documents that contain unbounded variables. As the document evolves more and more variables valuated, finally the documents achieve a state in that the documents cannot contain any unbounded variables. We can call documents in this state as ground-documents. The parts of documents can be regarded a finalized one from the viewpoint of one of the system roles; however, the parts of the documents may still contain some free variables that require further processing by some other system roles. The external information is supplied by system roles out of the organization. i.e. outside of IS, the steps of the fulfillment process and their sequences are defined by business rules of the organization Valuation of a free variable needs external information what is supplied by system roles out of the organization, i.e. outside of Information Systems, the steps of the fulfillment process and their sequences are defined by business rules of the organization. For that reason, we make differences between the states of *finalized* and *ground-document*. However, the responsibility for recognizing the proper data items relates to the currently valid *system role* (human or business process). The previously published figure (see [2]) (Fig. 5) has been extended to designate the *name space* of document's DBMS, emphasize the mutual mapping between the information related to *control* and *business processes*, and to pinpoint to set of models that play a crucial role in integration of *enterprises* and *Information Systems*.

A finalized document may contain free-variables and/or error signaling variables /fields that designate the necessity for further processing by some certain roles. The defect resolution of documents happens typically by organizational roles, i.e. outside of the automated Information Systems. In the case of algorithmic approach for error handling, the further document processing requires an intensional treatment, and usage of intensional documents, i.e. generate document instances based on business rules that are fulfilled by the automated Information Systems to create extensional occurrences. A stable state of an instance of an overarching business process within an IS can be achieved in the case if all documents that were involved in it are already ground-documents. The document handling finally results in ground-documents, ground sub-documents and assembled documents through several stages of development of to-be-finalized documents. The initial state is an uppermost documents and derived (intensional) documents. The intensional documents may contain free variables at meta-data and data level at the same time. On finishing their processing, the ground-documents build up a network (Fig. 6). To establish interdependencies among ground-documents may require some extra information. The supplemental information may assist to finish building-up the network of documents.

3.3 Representation of Documents

The current standards for describing the structure of documents are the XML, DOM (Document Object Model), JSON [15], [31]. The conceptual data model is either represented by entity-relationship or object-oriented modeling methods. The interdependency between document model and data model can be represented by RDF.

To support enterprise architecture, the recent IT architectures (SOA, REST, etc.) offer procedures as orchestration and choreography to create complex services along with documents. The documents may belong to various categories as generic, intensional, to-be-finalized, and ground-document type. The architectures provide the opportunity to create protective, security and safety mechanisms [5], [24].

The document handling finally results in ground-documents, ground sub-documents and assembled documents through several stages of development of to-be-finalized documents. The initial state is an uppermost documents and derived (intensional) documents. The intensional documents may contain free variables at meta-data and data level at the same time. On finishing their processing, the ground-documents build up a network. To establish interdependencies among ground-documents may require some extra information. The supplemental information may assist to finish building-up the network of documents.

3.4 The proposed document model

A database-centric IS model that is based on an information theory approach [25] outlines a framework that describes the input, output and query processing. Fig. 7 contains the IS model indicated by the dashed line; the previously published version [2] was enhanced to express that the source code data outside of the system and the code generated by the information system towards destination are communicated through a *crust consisting of documents*. In computerized systems, interactive documents and Web services become visible on the source and the output side. Free documents appear at the *interface/façade* level. The system roles (either human or automated system) perform variable valuation, or binding at each single variable through simple tasks. The business activities consist of tasks; a task can be composed of elementary tasks. An elementary task can be coupled to specific variables and its manipulation. The end-users who typically use information can retrieve data through documents, e.g. querying and fetching data from database and then processing the obtained responses.

The two sides of the model, the input and potential output data are separated by the document model in the Fig. 7. Although the various possible states and instances of document types integrates both sides at the same time. The two sides show the same behavior but provide different services. The twofold behavior is actually either querying or alteration like.

In front of data model and its manifestation in the form of a database system, a new, document model should be placed in. Beside the logical formulation of data retrieval and modification, the model should contain the description for sequences of interaction among documents; moreover, they should deal with collection of documents.

We can make difference between documents as being static or dynamic. The structure of a dynamic document may change as the response that is triggered by the system or system roles indicate it. The response may create instances of a general dynamic document that results in a sequence of free documents. The free documents are gradually converted into ground-documents starting from generic ones through intensional ones to be finalized and ground-documents. The ground-documents do not include any free variables thereby the names of variables in the ground-documents can be placed into the name-space of database.

The system of documents – generic, intensional, free-documents, to-be-finalized and ground-documents – can be perceived as a meta-database. This meta-database encloses not only static structures, but it includes active component as well that can be realized by web services. The *active component* incorporates the potential *program code* for interactions among the system roles, documents etc. The active components encapsulate the codes for database management too. The above-mentioned techniques can be integrated into a unified framework (Fig. 3, Fig. 4). Although, there is a lack for a comprehensive and not too complex scheme that combines all elements required for modeling Information Systems from a document-centric viewpoint and it is computable. Our proposal takes a step into the direction that both theoretic modeling and engineering viewpoint can be vindicated in a unified approach. The hypergraph as an appropriate mathematical tool may serve as a unifying approach to reconcile the before-mentioned heterogeneous model into a unified schema.

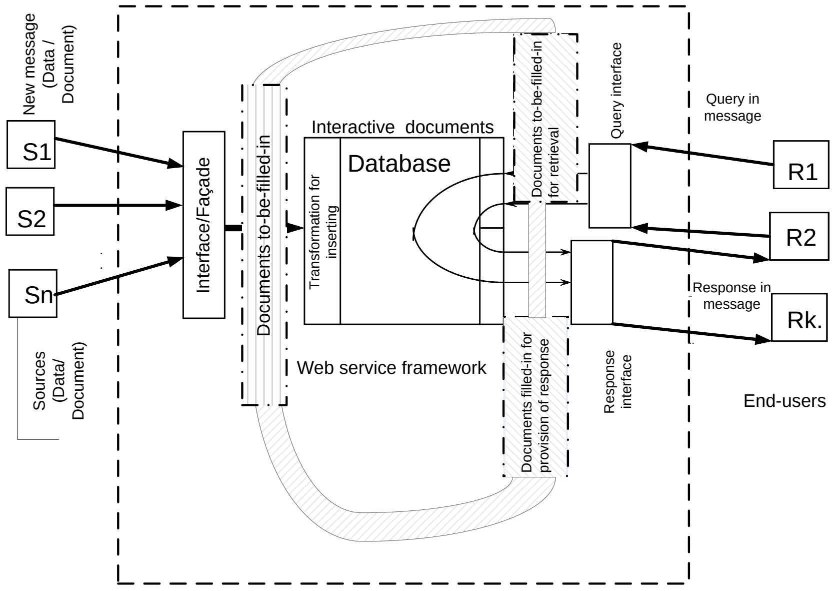

Fig. 7. Information Systems' model in a document-centric approach.

4. Formal mathematical background

Hypergraphs. As we have outlined previously, the problem to be solved can be described as a set of complex, heterogeneous relationships. The basic components that appear as constituent participate sometimes in hierarchical, sometimes rather network-like relationships that are different to each other. The hypergraphs as mathematical structure seems to be apt to representing the interrelationships among the models, views, viewpoints, perspectives, and the overarching documents and business processes [7].
We start with the basic definitions of hypergraphs in order to employ for depicting the before-mentioned complex relationships.

Definition 1. A *hypergraph H* is a pair *(V, E)* of a finite set $V = \{v_1,..., v_n\}$ and a set *E* of nonempty subsets of *V*. The elements of *V* are called vertices; the elements of *E* are called edges [22].

Definition 2. *Generalized* or extended hypergraph. The notion of hypergraph may be extended so that the hyperedges can be represented – in certain cases – as vertices, i.e. a hyperedge *e* may consist of both vertices and hyperedges as well. The hyperedges that are contained within the hyperedge *e* should be different from *e* [22].

The hypergraphs as a tool for describing Information Systems from various viewpoints yields a formal method to analyze the system, and to check the conformance, compliance and consistency of the set of models. The representation created by the above-mentioned way can be leveraged for design and operational purposes as well. Considering a document model, a particular document type hierarchy can be perceived as a “hierarchy” of hyperedges. The free variables or placeholders to be filled-in may occur as ultimate vertices within hyperedges that represents the instance of extension of particular document type. In a document subpart hierarchy, a specific subpart of document may be denoted by a vertex within a particular hyperedge that describes this document that contains the subpart, although that subpart as a vertex may include a document type hierarchy that can be depicted by a hyperedge.

Definition 3. A *directed hypergraph* is an ordered pair

$$\vec{H} = \left(V; \vec{E} = \{\vec{e}_i : i \in I\}\right) \qquad (1)$$

Where *V* is a finite *set of vertices* end $\vec{E}$ is a set of *hyperarcs (directed hyperedge)* with finite index set *I*. Every *hyperarc* $\vec{\mathbf{e}}_i$ can be perceived as an ordered pair

$$\vec{e}_i = \left(\vec{e}_i^{\,+} = \left(e_i^+, i\right); \vec{e}_i^{\,-} = \left(i, e_i^-\right)\right) \qquad (2)$$

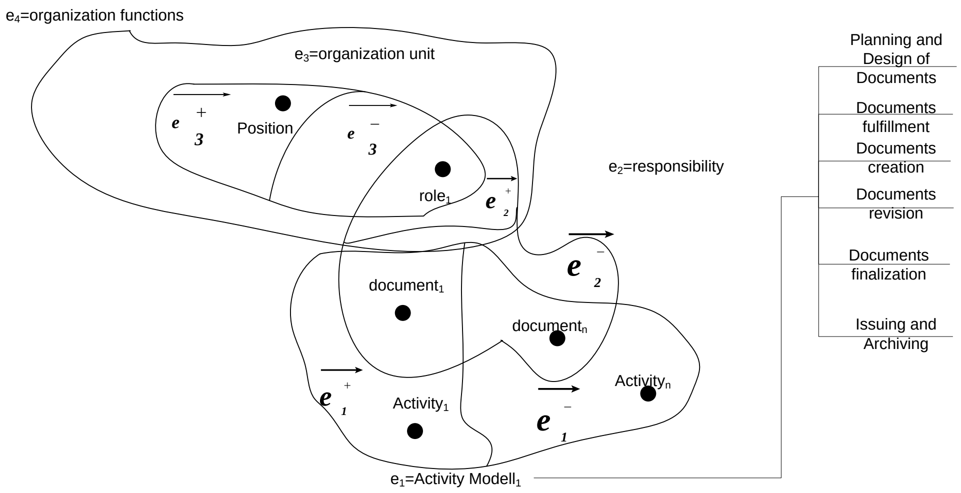

Fig. 8. Example for Directed Hypergraph Representing a Sample of Essential Relationships.

Where $e_i^+ \subseteq V$ is the set of vertices of $\vec{e}_i^{\,+}$ and $e_i^- \subseteq V$ is the set of vertices $\vec{e}_i^{\,-}$. The elements of $\vec{e}_i^{\,+}$ (hyperedges and/or vertices) are called *tail* of $\vec{\mathbf{e}}_i$, while elements of $\vec{e}_i^{\,-}$ are called *head* [22]. We may use as shorthand notation for ordered pairs, e.g. a vertex and a directed hyperedge as ordered pair <v_i, e_j>.

The underlying graph representation is based on the hypergraphs and directed hypergraphs. The potential implementations of hypergraphs in a hypergraph database make allowance for linking attributes to vertices, even more to hyperedges. The target domain, namely documents and model of Information Systems within organizations, contains complex n-ary relationships. The hypergraph provides the opportunity to depict recursive construction, to describe logical relations, to store compound structures along with their values [26], [27], [28]. As an illustration of the basic concepts of directed hypergraph, an example can be seen in Fig. 8. that makes sense of the representation for the domain by hypergraph. The essential characteristics is that vertices contain composite constituents that are themselves may be graphs; generalized hyperedge may contain other hyperedges but not itself and vertices.

Definition 4. *Architecture Describing Hypergraph* is a generalized hypergraph with undirected and directed hyperedges. It can be designated as a tuple $\langle V, A, E, E_U, E_D, Attr \rangle$:

- $\boldsymbol{V}$ is the set of *vertices*;
- $\boldsymbol{A}$ is the set of arcs, i.e. directed edges, an arc is an ordered pair $\langle i, j \rangle$, where $i, j \in V$;
- $\boldsymbol{E}$ is the set of hyperedges;
- $\boldsymbol{E}_U$ is the set of the *undirected* hyperedges, because of the properties of generalized hypergraphs, a hyperedge *e* is
  - either $e \neq \emptyset, e \subseteq V$, (*basic hyperedge*),
  - or a *bag* of hyperedges;
- $\boldsymbol{E}_U$ is divided up at a meta-level into partitions:
  - $\boldsymbol{E}_C$ consists of the *configuration hyperedges*. Each $h_i \in \boldsymbol{E}_C$ is a simple hyperedge, i.e. containing only vertices, not complex structures and other hyperedges. All $h_i \in \boldsymbol{E}_C$ can be labeled unambiguously. The configuration hyperedges manifests the structure of "things", the vertices within a hyperedge are the properties of the specific "thing". The properties can be perceived as variables or attributes (depending on the context) that can be valuated thereby they linked to an individual value (vertex in *D* (see **Definition 6**.)) or a set of values, e.g. to a grouping hyperedge;
  - $\boldsymbol{E}_E$ is composed of the e*xtensional hyperedges*. The extensional hyperedges can represent collections of data, the instances of *generic documents*. For example, the collections of data can be built up by tuples of data items, the instances of documents can be composed of certain bags of free variables that are contained in the particular documents' object structure. In these examples, the distinct elements, the vertices of these hyperedges can be considered as constituents of extensional hyperedges;
  - $\boldsymbol{E}_I$ comprises the *intensional hyperedges*. The intensional hyperedges show the logical and rule-based interrelationships among the vertices (models within the architecture), moreover configuration hyperedges;
  - $\boldsymbol{E}_G$ is made up of *grouping hyperedges* that embody various structuring principles on components, as e.g. view, viewpoint and perspectives etc. in architecture describing approaches; they symbolize interrelationships between certain models and pieces or parts of documents as e.g. business activity models, documents and responsibilities of roles within an organization unit. The hyperedge $h \in \boldsymbol{E}_G$ can be utilized for sorting the vertices (representing either documents or models) into organizational-related, document-related and activity related relationships.
- $\boldsymbol{E}_D$ is a set of *hyperarcs*, i.e. *directed hyperedges;* the hyperarc $\overrightarrow{e_i} \in \boldsymbol{E}_D$ can be as it follows (see **Definition 3**):
  - $\overrightarrow{e_i} =< v_j, \vec{h} >= \left(\overrightarrow{e_i^+} = (e_i^+ = v_j, i); \overrightarrow{e_i^-} = (i, e_i^- = h)\right)$ where $v_j \in V$, and $\vec{h} \in \boldsymbol{E}_G$;
  - $\overrightarrow{e_i} =< v_j, \vec{h} >= \left(\overrightarrow{e_i^+} = (e_i^+ = v_j, i); \overrightarrow{e_i^-} = (i, e_i^- = h)\right)$ where $v_j \in V$, and $\vec{h} \in \boldsymbol{E}_C$;
  - $\overrightarrow{e_i} =< v_j, \vec{h} >= \left(\overrightarrow{e_i^+} = (e_i^+ = v_j, i); \overrightarrow{e_i^-} = (i, e_i^- = h)\right)$ where $v_j \in V$, and $\vec{h} \in \boldsymbol{E}_E$;

– there does not exist two hyperarcs $\vec{e}_i =< v_j, \vec{h} >$ and $\vec{e}_k =< v_l, \vec{h}' >$ that either $\vec{h}, \vec{h}' \in \boldsymbol{E}_C$ or $\vec{h}, \vec{h}' \in \boldsymbol{E}_E$, i.e. every vertex $v_j \in V$ is linked, at most, to one configuration hyperedge ($\boldsymbol{E}_C$) and at most to one extensional hyperedge ($\boldsymbol{E}_E$). These conditions can be interpreted the following way: a vertex may belong to a configuration structure (either document or model), or it may belong to an extension that represents the instantiation of either a document or a model.

Description Logics. One of the most common approaches of formalization is the use of some mathematical-logical language. The Description Logics belong to mathematical logics, and their purpose is formal knowledge representation [29]. Compared to propositional calculus (or propositional logic), the expressiveness of description logic is higher, and it has a more efficient algorithm for the decision problem than the first-order predicate logic. On the other hand, the network like knowledge representation - where the elements of the network are vertices and links are relationships as e.g. the semantic net- work - can be related to the theory of hypergraphs. In both case, vertices can be used to define concepts, and links can be used to characterize the relationships among them. Bearing this in mind, it is obvious to apply description logic on a system based on the mathematical background of hypergraphs.

The knowledge representation systems based on Description Logics contains two main components: the TBox, and the ABox. The TBox introduces the terminology, i.e., the basic concepts, which denote sets of individuals (atomic and complex), and roles, which define binary relations between individuals. These are forming the vocabulary of an application domain. The ABox contains assertions among named individuals and the vocabulary.

There are many variations of the Description Logics (originated from the varieties of description languages) and there is an informal convention, where their name indicates which operators are allowed. For example, a basic logical language is the Attributive Language – AL, which allows: atomic negation; concept intersection; limited existential quantification; and universal restriction. This can be extended with other operators, as e.g. concept union (U), full existential qualification (E), cardinality restriction (N), or complex concept negation (C). The description language lays the groundwork for the description logic [29].

To illustrate Description Logic in document centric environment, we show some examples below:

- With ***Parameter*** ⊑ (***Free Variable*** ⊔ ***Bound Variable***) notation we describe, that a document parameter can be either free or bound variable.
- ***Parameter*** ⊑ ∃***is_part_of.*** (***Document Fragment***) means, that a document fragment consists of parameters, and ***Document Fragment***. ∃ ***is_derived.*** (***Free Document***) means, that the document fragments are derived from unprocessed free documents.
- ***State.P*** ⊑ ∃***has_successor.*** (***Action State.Q***) means, that *Q* action-state follows the *P* state.
- The following line describe, that an action-state needs free variables to work with: ***Action State*** ⊑ ∃***has_free_variables*** (***Document***); ***has_free_variables*** ≡ ≥ 1 ***is_free_variable*** ⊓ ***is_free_parameter***.***Parameter***

The output of a well-designed formalization of an information system that is depicted by description logics and – at the same time – represented by a hypergraph exists in machine-readable format thereby this formalization has the opportunity to use various frameworks and tools to evaluate the model. With this it is possible to effectively optimize the information system even in the early model-development phase.

5. Formalized Document Centric Approach

In the case of a particular organization, we can imagine that there is a comprehensive document that is a representation, in conceptual sense, all potential documents. This overarching document is composed of generic document types. Generic document types are hierarchical structures that can be described by configuration hyperedges that reflect the composition of documents. There are hierarchical relations among the members of a generic document. The hierarchical

relationships can be described by configuration hyperedges; the instances of a generic document member can be perceived as extensions and can be represented by extensional hyperedges.
There is an approach that recognizes documents as “a unit of business information exchanged in a business transaction’’, i.e. as a medium for message exchange between business partners [30]. Business can be perceived as a general notion in this context, namely the entirety of commercial and non-profit companies, public services and public administration as branches of economy and societal life use documents, and decisively electronic documents. Because of proliferation of computer literacy, the users’ requirements stated more frequently in the form of documents as e.g. word processors, calculation tables, etc. There are several sectors of IT applications where documents in various conceptual forms play important roles. A document can be described technically by *XML* schema, and additional or contextual information may be supplied by *DTD*, *XLink*, *XInclude*, *XSL/XSLT*. The Document Object Model (*DOM*) yields an object-centric representation for documents [31].

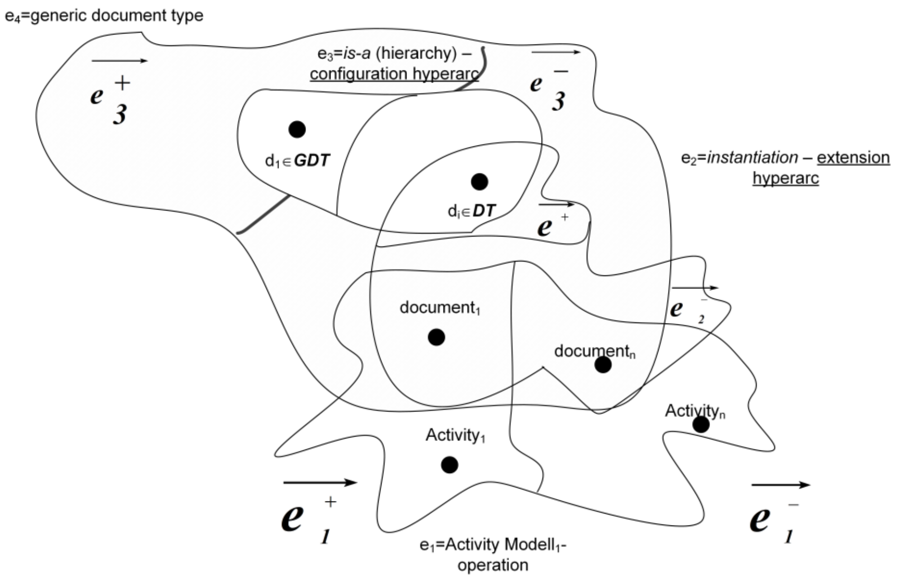
e4=generic document type
e3=is-a (hierarchy) – configuration hyperarc
e2=instantiation – extension hyperarc
d1∈GDT
di∈DT
document1
documentn
Activity1
Activityn
e1=Activity Modell1-operation

Fig. 9. Interrelationships of Documents Represented by Hyperedges.

In our approach, we emphasize the existence of overarching document as a projection of the embracing organization. The parts and subparts of documents and document hierarchies are the subject of operation that initiated by business processes, activities and tasks. The responsibilities for executing the operation linked to roles within the organization. The documents utilize the underlying collections of data and thereby the serve as media to facilitate the data flow within organization.

A generic document type ***GDT*** is a hierarchy of document types ***DTH***. The elements of ***DTH*** can belong to a configuration hyperedge e_{Ci} as vertices. The generalized hypergraphs allow that the vertices may appear as complex structures, as hyperedges. Therefore, a vertex can be a hyperedge that itself a configuration hyperedge that contains a hierarchy of document types. Thereby, the representation makes possible for a recursive definition of document types and gathering them into a generic document type.

The direction of the hyperarc shows whether a document plays the *input* or *output* role in a particular context (Fig. 9). The definition of the hyperarc is given above (see formula (2)) permits the differentiation between the *information* represented by the *head* and *tail* of a hyperarc, and the *information* that are represented in the form of vertices that are contained within the heads and the tails [22].

Definition 5. The *Document Subhypergraph* consists of:

- A finite set of documents that are represented by vertices ***DOC***= {doc_1, ..., doc_n};
    - The documents contain variables, the variables belong to certain attribute type of Attr= {T_1, …, Tn} that consist of the attribute types;
    - The finite set of domains is ***DOMSET***= {D_1....D_k} that contains the domain of each single type, T_i;
- The relationship between a generic document type ***GDT*** hierarchy and its constituents document types belonging to a ***DTH*** can be described by hyperarcs 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.
- The relationship between a document doc_i and a document type ***DT*** can be described by a hyperarc representing the *instance-of* relationship.

The instances of a document type can be linked to the particular document type through an extension hyperedge. The document instance contains typically free-variables; thus these document instances can be called as free-documents. Free documents as document instances and extensions of document types are the subject of manipulation by business processes. A value for a variable can be a new fact or a new free-document of appropriate types. The concept of generic document type offers possibilities for derivation of new document types from other document types that can be regarded as templates. The derivation rules can be formalized by logical statement that may create either a slightly different document type according to the structure of documents and then an instance of it or operate during the lifecycle of an instance of the document types. A document type may contain business rules in the form of predicates, data retrieving and calculation rules. Both cases demand operators that create documents through intension, i.e. logical inferences. To depict these relationships, the intensional hyperedges can be used. The common characteristics in both cases is that neither the creation of a new document type and its instance nor a document instance with more valuated variables require human interaction through business processes and activities, they should be fully automated. A fully automated business process may be described in BPEL (Business Process Execution Language), but the full automation raises several issues that should be handled if there is no direct human, external interaction at a certain point of time during the execution. During their lifecycles, the free variables of free-documents are valuated, i.e. a variable is set for a value. A document is modified during processing by business activities in the context of actual responsibilities (organization units, roles, actors). A document may achieve the finalized status but the policy and rules of organizations permits further processing in some cases. When a document is in such a status that it cannot be modified in any case then this document can be called ground document. This situation is typical in public administration as it manufactures document during the business process and ships a ground document to the customer. The time is important factor of life cycle of documents. The interplay between business activities and documents moves through the time dimension.

6. Information Architecture and Documents

As we have seen previously, the documents are strongly coupled to their embracing organization context, even defining the appropriate document types request referencing to the related activities. Beside the essential documents, IS can be described by various models that are ordered into a reasonable structure by Enterprise Architecture approach. To describe the document manipulation requires operators so that we can extend the definition:

Definition 6. *Architecture Describing Hypergraph* is a generalized hypergraph that can be extended by some functions and operations:

- $label_{node} : V \rightarrow L_{node}$; where L is a set of labels, it is a vertex labeling function;
- $label_{edge} : E \rightarrow L_{edge}$; where L is a set of labels, it is an edge labeling function;
- $source_E : E \rightarrow V;$
- $target_E : E \rightarrow V;$ these functions return the source and target vertices of an edge *E;*
- $attr \ : Attr \rightarrow V \ ;$ attribute assignment function;
- $source_{Attr} : Attr \rightarrow V;$ The vertex that owns the attribute is returned;
- $target_{Attr} : Attr \rightarrow D;$ The *data values* of attributes are yielded; *D* represents the set of data.
- *D* can be grasped (efficiency of the representation is left out of the investigation) again as vertices within the hypergraph and it can be interpreted as *variables*.
- Over *D* as a set of variables, set of operations (*OP*) can be defined that can be used to describe constraints and rules within formulas.

Table 1. Representation of Information Systems by Hypergraph.

Concept of Information System Theory	**Representation of concept in the domain of hypergraph theory**
Information System	A result of a system-development exercise that created a set of design artifacts. The set of elements and *relationships* among them can be represented as vertices and edges within the graph. We can map the model elements to a *hypergraph* that consists of vertices and hyperedges.
Node/vertex in a hypergraph	Each vertex corresponds to an element within an Information Systems, e.g. *documents*, elements of documents (constituting a tree structure), business processes, workflows, layers of workflows, web services, networks of web services, etc. The documents may represent one of the aspects for the information flow both inwards and outwards.
Edge in a hypergraph	*Edge* is a specific *hyperedge* with cardinality equal to two. Edge denotes binary relationships between two vertices, as e.g. free documents is processed by a certain Web service, a generic document is the ancestor of an intensional documents, a free-document resulted in a ground-document after binding, valuating of variables, etc.
Hyperedge	A hyperedge represents a relationship among a subset of vertices as e.g. Web services belonging to a specific workflow, business process containing workflows, etc.
System graph	A hypergraph that includes a disjoint vertex for modeling the environment of the system, plus all the vertices and hyperedges of the WIS.
Sub-system	A subset of vertices and their incident hyperedges. A vertex is *incident* to a hyperedge if the hyperedge contains the vertex. A sub-system may be composed of documents, Web services and related entities out of data model, etc.
Interconnecting sub-systems hyperedges graph of the generalized hypergraph	A graph consisting of all the vertices in a sub-system and all hyperedges connecting together subsystems.

Beside the documents, the various models that follow some architectural description and system design style are essential constituents of IS (Table 1). The hypergraph representation gives the chance to represent the complex interactions and interrelationships among models and documents that drives the behavior of systems. The object-oriented paradigm and UML visual language proliferated as specification language for models. For the uniform discussion, we presume that all of the models in line with the UML modeling and visual language standard, moreover their representations pursue the object-oriented, meta-data structure codified into standard. The models' descriptions appear usually in semi-structured document forms as XML and/or JSON that offers a chance for uniform treatment of documents and models of Information Systems. As structuring principal for models of IS, we can use Zachman ontology and/or TOGAF (Fig. 5) [6], [7]. A model is a description of specific properties of an IS and it represents an artifact of views, viewpoints and perspectives [7]; or it can be perceived as an architectural building block of the system [6]. The set of relations among models and the internal structure of models plays essential role.

The models can be arranged into three meta-groups namely *organization*, *documents* and *activities* related models. For modeling, the relationships and interactions among these three meta-groups and the underlying collections of data are significant. The models, documents and concepts of IS and a vertex representing the external environment compose a hypergraph that embraces all important parts of the application domain that may be called as *System Hypergraph*. The specific models can be considered as complex structures, and at the first cut, they can be represented as vertices containing the information about the model, possibly in the form of a hypergraph, because of generalized hypergraph permits to set up a hierarchy. We may structure the overarching hypergraph several sub-hypergraphs as documents, organization and its units, underlying data collections, business processes and their constituents.

We can exploit the flexibility of hypergraphs to describe relationships. A hyperedge, and a hyperarc (directed hyperedge) can depict various relationships. In the case of documents, a hyperarc can express the input and output roles of documents that they may fulfil within activities of business processes. The document may be attached to organization units and actors through a *responsibility hyperedge* (labelled directed hyperedge). The variables of documents may be connected to data vertices of *D* that is organized into reasonable partitions that are represented by vertices contained in hyperedges that can be mutually mapped to specific data collections. These sub-hypergraphs may be called *Sub-system Hypergraph*s. Between the models, a refinement relation can be identified within an architectural perspective (Fig. 5) and represented by a hyperarc (directed hyperedge) *is-a-refinement*. The documents and their structures can be described by documents model.

Definition 7. *Models* of IS represented in the Architecture Describing Hypergraph are:

- The set of vertices is divided up into two basic subsets V_{Doc} and V_{Model};
- $V_{Doc} \supseteq \{OGDT\}$ where *OGDT* signifies the *overarching generic document*, that is the super type of all other document types and their instances;
- $V_{Model} \supseteq \{EA, \{external\_evironment\}\}$, where *EA* designates the overall *Enterprise Architecture* consisting of models, the *external_evironment* refers to the outside world that is typically the source of *stimulus* that is generated by either humans or any other systems;
- $V_{Configuration} = \cup\, h_i$ where $h_i \in \boldsymbol{E}_C$, and $\cap\, h_i = \varnothing$ where $h_i \in \boldsymbol{E}_C$.

The expressions articulate the fact that the configuration hyperedges represents the structure of artifacts of models and documents in the form of structural constituents as vertices.

- The set of arcs (directed edges of graphs) $\mathbf{A}$ is partitioned into subsets $\mathbf{A}_{\text{Doc\_Target}}$, $\mathbf{A}_{\text{Model\_Target}}$, $\mathbf{A}_{\text{Interaction}}$, where $\mathbf{A}_{\text{Doc\_Target}} \subseteq \text{V}_{\text{Configuration}} \times \text{V}_{\text{Doc}}$, $\mathbf{A}_{\text{Model\_Target}} \subseteq \text{V}_{\text{Configuration}} \times \text{V}_{\text{Model}}$.

The directed edges, the arcs map a complex structure, a configuration of elements (vertices) to a vertex that represents either a document or a model.

- $\mathbf{HA}_{\text{Interaction}} \subseteq \text{V}_{\text{Model}} \times \text{V}_{\text{Doc}}$, $\mathbf{HA}_{\text{Interaction}} \subseteq \mathbf{E}_{\text{D}}$;

The interaction between certain models and specific documents can be expressed by a hyperedge $h \in \boldsymbol{HA}_{Interaction}$.

- $\mathbf{E}_{\text{C}}$ can be partitioned into two subsets $\mathbf{E}_{\text{Configuration\_Document}}$ and $\mathbf{E}_{\text{Configuration\_Model}}$.

The hyperedges $h_{i,\ cd} \in \boldsymbol{E}_{Configuration\_Document}$, $h_{j,\ cd} \in \boldsymbol{E}_{Configuration\_Model}$ represent an inheritance structure. The inheritance structure conforms to the object-oriented paradigm, i.e. the configuration of documents and models inherit the attributes of super-classes, and may have extra attributes as well. Each attribute of a certain configuration can be represented by a vertex of the hyperedge. An attribute linked to a vertex either in V_{Model} or in V_{Doc}, its value represented by a link to a $d \in D$ when it is valuated. If the attribute is multi-valued, then the attribute is connected to hyperedge $h \in Power\ (D)$ (the power set of D).

- The set of extensional hyperedges $\boldsymbol{E}_E$ is split into two subsets $\boldsymbol{E}_{Superclass}$ and $\boldsymbol{E}_{Extension}$
  - The hyperarc $h \in \boldsymbol{E}_{Superclass}$, if $h \in \boldsymbol{E}_E$, (h set of vertices)
    - Either $h \subset V_{Doc}$ and $OGDT \in h$
    - or $h \subset V_{Model}$ and $EA \in h$.
    - Given a vertex $v_i \in h$ and $h' \in \boldsymbol{E}_{Superclass}$, then either valid that $< v_i, h' > \in \boldsymbol{E}_{Super\_doc}$, then $h' \subseteq h$
    - Or $< v_i, h' > \in \boldsymbol{E}_{Super\_model}$. then $h' \subseteq h$
      - Notation: $\boldsymbol{E}_{Super\_doc} = V_{Doc} \setminus \{OGDT\}) \times \boldsymbol{E}_{Superclass} \subset \boldsymbol{E}_D$;
      - Notation: $\boldsymbol{E}_{Super\_model} = ((V_{Model} \setminus \{EA, \{external\_evironment\}\}) \times \boldsymbol{E}_{Superclass} \subset \boldsymbol{E}_D)$.

The hyperedges $h \in \boldsymbol{E}_{Superclass}$ provide the association between a class of objects (models or documents) and its super-classes in compliance to the object-oriented paradigm. For the reason for our modeling approach, we make distinction between the two top super-classes, namely *OGDT*, the *overarching generic document*, *EA* the overall *Enterprise Architecture*. The conditions above specify the transitivity of *is-a* relationship for the relation between class and its super-classes.

- The instances of models can be represented by $\boldsymbol{E}_{Instance\_model} \subset V_{Model} \times \boldsymbol{E}_E$ (extensional);
- The instances of documents can be represented by $\boldsymbol{E}_{Instance\_doc} \subset V_{Doc} \times \boldsymbol{E}_E$;
- $h \in \boldsymbol{E}_E$, (h set of vertices) is $h \in \boldsymbol{E}_{Attribute\_Set}$ if $h \subset D$. The following statement is valid as well: $\cup\ h_i = D$, $h_i \in \boldsymbol{E}_{Attribute\_Set}$. The hyperarcs $h \in \boldsymbol{E}_{Attribute\_Set}$ are used to represent the attributes domains, and the associated values;
- The hyperarc $h \in \boldsymbol{E}_{Extension}$, if $h \in \boldsymbol{E}_E$, (h set of vertices) and
  - Given a vertex $v_i \in h \subset V_{Doc}$ and $h \in \boldsymbol{E}_{Extension}$, then $< v_i, h > \in \boldsymbol{E}_{Instance\_doc}$, $< v_i, h' > \in \boldsymbol{E}_{Super\_doc}$, then for each $n \in h$ and each $d_t \in h'\ \exists\ ha \in \boldsymbol{E}_E$ (hyperarc) where $< d_t, ha > \in \boldsymbol{E}_{Instance\_doc}$;
  - Or
  - Given a vertex $v_i \in h \subset V_{Model}$ and $h \in \boldsymbol{E}_{Extension}$, then $< v_i, h > \in \boldsymbol{E}_{Instance\_model}$, $< v_i, h' > \in \boldsymbol{E}_{Super\_model}$, then for each $n \in h$ and each $d_t \in h'\ \exists\ ha \in \boldsymbol{E}_E$ (hyperarc) where $< d_t\ , ha > \in \boldsymbol{E}_{Instance\_model}$.

A hyperedge $h \in \boldsymbol{E}_{Extension}$ represents an extension for models and documents respectively as well. The above described statement formalizes the transitivity of *instance-of* relationship.

- The intensional hyperarc $h \in \boldsymbol{E}_I$, $< d, h> \in \boldsymbol{E}_{Intension}$ if $\boldsymbol{E}_{Intension} \subset V_{Doc} \times \boldsymbol{E}_I$, $d \in V_{Doc}$, $h \in \boldsymbol{E}_{Configuration\_Document,}$ $h \subset V_{Doc}$; the intensional hyperarc defines the hierarchical relationship between templates, rule-based document types and extensional document types that are instantiated.
- The set of hyperarcs (directed hyperedges) in $\boldsymbol{E}_D$ can be arranged into several subsets according to the notion of Enterprise Architecture:
  - The hyperarc $h \in \boldsymbol{E}_{View} \subseteq \boldsymbol{E}_G$, $h \subseteq V_{Model}$, represents a stakeholder's view that puts together models that describe the specific viewpoint of a role within organization. The hyperarc may be defined as $<r_i, m_j = (e_i^-; j \in \mathrm{I})>$, where r_i represents a vertex within an organizational model and it is mapped to a role of organization; $m_j \subset \boldsymbol{E}_{Instance\_model,}$ or $m_j \subset \boldsymbol{E}_{Configuration\_Model}$ before instantiation of models;

- The hyperarc $h \in \boldsymbol{E}_{Perspective} \subseteq \boldsymbol{E}_G$, $h \subseteq Powerset\ (V_{Model}\ )$, embodies a hierarchy of models according to a refinement hierarchy;
- The hyperarc $h \in \boldsymbol{E}_{Doc\_Life\_cycle} \subseteq \boldsymbol{E}_G$, $<d, h> \in \boldsymbol{E}_{Instance\_doc} \times \boldsymbol{E}_{Instance\_model}$, $d \in V_{Doc}$, that depicts the life cycle of document through the interactions with models.

7. Conclusion

We have described issues and problems of modeling IS. The recent evolution of technologies at user interface level and database handling raised questions that can be solved through new modeling approaches taking into account of ubiquitous documents as data holder.

Using of successful methods for single particular views, viewpoints and models, a framework for unifying the various approaches is outlined. To provide a theoretically sound but reasonable complex and comprehensive approach for description and research of IS a hypergraph based method is proposed (Table 1). The direction of future research is to exploit the hypergraph as mathematical model to formalize the IS' model from a document centric view.

In this paper we proposed an Architecture Describing Hypergraph as representation for Enterprise Architectures and related Documents. The suggested descriptive method takes advantages from the basic properties of generalized hypergraphs, i.e. unequivocal representation of complex relationships; moreover, there are some distinguished features:

- Uniform treatment of both intensional and extensional aspects of documents and models within Enterprise Architecture;
- Direct depiction of hierarchical relationships through instance-of, sub-class-of, super-class-of relationships.

The outlined approach can also be considered as a formal background to analyze and design IS. The documents play important roles in Information Systems in the time of analysis, design, specification and operation with strong coupling to roles of organizations. The unified framework provides an opportunity for uniform handling of models and documents on a formal foundation.

The hypergraph–based approach offers the chance to apply further mathematical tools for assistance in the design, verification and validation to maintain the integrity and consistency of IS.

References

[1] G. Joeris, "Cooperative and integrated workflow and document management for engineering applications," In *Database and Expert Systems Applications,* Toulouse, France, 1997, pp. 68-73.

[2] B. Molnár and A. Benczúr, "Facet of Modeling Web Information Systems from a Document-Centric View," *International Journal of Web Portals (IJWP),* vol. 5, no. 4, pp. 57-70, 2013.

[3] B. Molnár, A. Benczúr and Á. Tarcsi. "Formal Approach to a Web Information System Based on Story Algebra," *Singidunum Journal of Applied Sciences: Economy Management Tourism Information Technology and Law* vol. 9, no. 2, pp. 63-73, 2012.

[4] T. Wewers and C., Wargitsch, "Four dimensions of interorganizational, document-oriented workflow: a case study of the approval of hazardous-waste disposal," In: *System Sciences, The 3lst Hawaii International Conference*, Hawaii, USA, 1998, vol. 4, pp. 332–341.

[5] OASIS, "*A reference model for service-oriented architecture*," White Paper, Service-Oriented Architecture Reference Model Technical Committee, Organization for the Advancement of Structured Information Standards, Billerica, MA, February, 2006.

[6] Open Group 2010, *“TOGAF: The Open Group Architecture Framework, TOGAF® Version 9,”* Available http://www.opengroup.org/togaf/

[7] J.A. Zachman, “A Framework for Information Systems Architecture,” *IBM Systems Journal Volume*, vol. 26, no. 3, pp. 276--292, 1987.

[8] K. Bhattacharya, C. Gerede, R. Hull, R. Liu and J. Su, “Towards Formal Analysis of Artifact-Centric Business Process Models,” in *BPM 2007*, LNCS, vol. 4714, G. Alonso, P. Dadam, M. Rosemann, Eds., Heidelberg, Germany: Springer, 2007, pp. 288–304.

[9] S. Yongchareon and C. Liu, “A Process View Framework for Artifact-Centric Business Processes,” In *OTM 2010*. LNCS, vol. 6426. R. Meersman, T.S. Dillon, P. Herrero, Eds., Heidelberg, Germany: Springer, 2010, pp. 26–43.

[10] R. Hull, “Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges,” in *On the Move to Meaningful Internet Systems: OTM 2008*, R. Meersman, Z. Tari, Eds., Berlin, Heidelberg, Germany: Springer, 2008, pp. 1152-1163.

[11] R. Hull, "Data-Centricity and Services Interoperation" in *International Conference on Service-Oriented Computing*, B. Samik, P. Cesare, Z. Liang, F. Xiang, Eds., Berlin, Heidelberg, Germany: Springer, 2013, pp. 1-8.

[12] W3C 2001, “*Web Services Description Language (WSDL) 1.1*.” Available: http://www.w3.org/TR/wsdl

[13] M. Bernauer and M. Schrefl, “Self-maintaining web pages: from theory to practice,” *Data & Knowledge Engineering*, vol. 48, pp. 39-73, 2004.

[14] B. Chidlovskii, “Schema extraction from XML collections,” In *The 2nd ACM/IEEE-CS joint conference on Digital libraries*, Portland, Oregon, USA, 2002, pp. 291-292.

[15] C.-K. Nama, G.-S., Jang and J.-H. Ba, “An XML-based active document for intelligent web applications,” *Expert Systems with Applications*, vol. 25, pp. 165-176, 2003.

[16] B. Daum, *Modeling business objects with XML schema*, San Francisco, USA: Morgan Kaufmann, 2003

[17] A. Gábor, A. Kő, I. Szabó, K. Ternai and K. Varga, “Compliance Check in Semantic Business Process Management,” in *On the Move to Meaningful Internet Systems: OTM 2013 Workshops*, Graz, Austria, 2013, pp. 353-362.

[18] A. Kő and K. Ternai, “A Development Method for Ontology Based Business Processes,” in *eChallenges e-2011 Conference*, Florence, Italy, 2011.

[19] B. Molnár, Z. Máriás, Z. Suhajda and I. Fekete, “Amnis-Design and Implementation of an Adaptive Workflow Management System,” in *9th International Symposium on Applied Informatics and Related Areas* - AIS2014, Székesfehérvár, Hungary, 2014.

[20] A. Blokdijk and P. Blokdijk, *Planning and Design of Information Systems,* London, UK: Academic Press, 1987.

[21] N.P. Suh, *Axiomatic Design: Advantages and Applications*, New York, USA: Oxford University Press, 2001.

[22] A. Bretto, *Hypergraph Theory: An Introduction*. Cham, Switzerland: Springer International Publishing, 2013

[23] J. D. Herbsleb and R. E. Grinter, “Architectures, Coordination, and Distance: Conway's Law and Beyond,” *IEEE Software*, vol. 5. no 16, September, 1999, Available: http://dx.doi.org/10.1109/52.795103

[24] J. Webber, S. Parastatidis and I. Robinson, *REST in Practice: Hypermedia and Systems*, Sebastopol, CA, USA: O'Reilly Media, Inc, 2010.

[25] A. Benczúr, “The Evolution of Human Communication and the Information Revolution – A Mathematical Perspective,” *Mathematical and Computer Modeling*, vol. 38, pp. 691-708, 2003.

[26] G. Gallo, G. Longo, S. Pallottino and S. Nguyen, “Directed hypergraphs and applications,” *Discrete applied mathematics*, vol. 42, no. 2, pp. 177-201, 1993.

[27] G., Ausiello, P. G. Franciosa and D. Frigioni, “Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach,” in *Theoretical Computer Science, 7th Italian Conference, ICTCS 2001*, Torino, Italy, 2001, pp. 312-328.

[28] B. Iordanov, “Hypergraphdb: a generalized graph database,” in *Web-Age Information Management*, pp. 25-36, Springer Berlin Heidelberg 2010.

[29] F. Baader, *The description logic handbook: theory, implementation, and applications*, Cambridge, UK: Cambridge University Press, 2003.

[30] IDA, *IDA e-procurement protocol XML schemas initiative*. IDA working document. 2004.

[31] J. Marini, *Document Object Model: Processing Structured Documents*, New York, NY, USA: McGraw-Hill, Inc., 2002.

[32] Y. Baghdadi, “A business model for deploying Web services: A data-centric approach based on factual dependencies,” *Information Systems and e-Business Management*, vol. 3, no. 2, pp. 151-173, 2005.

Biographical notes

Bálint Molnár
He is Associate Professor (Dr. habil., PhD in Technical Informatics) at Eötvös Loránd University of Budapest, he teaches: Methodologies of Information System Development, ERP and Integrated Systems, Web technologies for Enterprise Information Systems, Database Management Systems, Theoretical Background of Information Management, Enterprise Architecture and Security Architectures. He is Associate Professor at Corvinus University of Economic Sciences, he teaches: Development of Information Systems, Project management, Knowledge-based systems development. His research area: Information System Modeling, ERP systems, Business Process Modeling, Semantic Web, Enterprise Architectures, SOA. His research interest covers topics that were before-mentioned as teaching subjects. He has published several scientific and professional papers and been engaged as a consultant and project manager at the Hungarian Public Administration. He is a member of the editorial board of Journal of Information Technology & Politics, The Electronic Journal of Knowledge Management (EJKM), and Singidunum Journal of Applied Sciences.

www.shortbio.net/molnarba@inf.elte.hu

András Benczúr
He is Professor Emeritus at Eötvös Loránd University of Budapest. He teaches Database Management Systems and its theoretical and formal backgrounds. His research interest covers the modern Data and Document Management Systems, Big Data, Cloud, Modeling of Information Systems. He has published several scientific and professional papers in Mathematics and Informatics. He is Doctor of Sciences at the Hungarian Academy of Sciences.

www.shortbio.net/abenczur@inf.elte.hu

András Béleczki
He is PhD student at Eötvös Loránd University of Budapest. His research interest covers Information Systems and Modeling, Big Data, Data Mining. He teaches Database Management Systems. He has published his first paper in December, 2015 about formal modeling of document-centric Information Systems.

www.shortbio.net/bearaai@inf.elte.hu