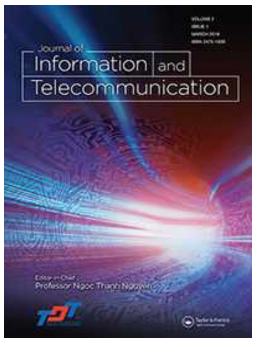


**Journal of Information and Telecommunication**

ISSN: 2475-1839 (Print) 2475-1847 (Online) Journal homepage: https://www.tandfonline.com/loi/tjit20

# Information systems modelling based on graph-theoretic background

B. Molnár, A. Béleczki & A. Benczúr

**To cite this article:** B. Molnár, A. Béleczki & A. Benczúr (2018) Information systems modelling based on graph-theoretic background, Journal of Information and Telecommunication, 2:1, 68-90, DOI: 10.1080/24751839.2017.1375223

**To link to this article:** https://doi.org/10.1080/24751839.2017.1375223

JOURNAL OF INFORMATION AND TELECOMMUNICATION, 2018
VOL. 2, NO. 1, 68–90
https://doi.org/10.1080/24751839.2017.1375223

# Information systems modelling based on graph-theoretic background

B. Molnár, A. Béleczki and A. Benczúr

Information Systems Department, Eötvös Loránd University, Budapest, Hungary

**ABSTRACT**

The recent tendency in analysis and design of information systems is that the emphasis is placed on the documents that are ubiquitous around information systems and organizations. The proliferation of computer literacy led to the general use of electronic documents. To understand the anticipated behaviour of information systems and the actual operation of an organization, the analysis of documents plays increasingly an important role. The behaviour of information systems can be interpreted in a framework of Enterprise Architecture and its models that are contained in it. Certain parts and entirety of various types of documents are connected to business processes, tasks, roles, and actors within an organization. The tracking of life cycle of documents and representing the complex relationships are essential at both analysis and operation time. We propose a theoretical framework that makes use of previous results of modelling and well-founded mathematical techniques. The basic idea is that the very flexible mathematical structure, the hypergraph, provides a sound groundwork on which a formal structure can be built up through mapping the essential concept, construction, components, and constituents of information systems. Thus, the representations of models for information systems that mapped onto a hypergraph can be analysed by either using more traditional tools as logic and inference rules or by a set of tools belonging to data science later. The paper describes the mapping of the important concepts onto hypergraphs as documents, processes in *cases*, their models and some rules for verification and validation; the hypergraph description can be interpreted as a concept hypergraph to be subjected for logical reasoning.

**ARTICLE HISTORY**
Received 15 April 2017
Accepted 31 August 2017

**KEYWORDS**
Information system modelling; document-centric modelling; information system architecture; Zachman framework; hypergraphs; ontology; description logics; concept hypergraph

## 1. Introduction

The information systems become more complex for several reasons. The use of various electronic document types is commonplace in organizations. The interactive forms, Web pages, the structured and semi-structured documents, on the one hand, are the stimulus to start the chain of activities within an organization; on the other hand, they are the end

**CONTACT** B. Molnár molnarba@inf.elte.hu, molnarba@gmail.com Information Systems Department, Eötvös Loránd University of Budapest, Pázmány Péter sétány 1/C, Budapest 1117, Hungary

of business processes. The Business and System Analysts typically encounter documents during the process of analysis, the conceptualization of requirements by users. The documents emerge in the disguise as requirements, subject of actual business processes and tasks, essential component of data processing internal to the information systems.

There are several modelling and descriptive approaches for modelling information systems. Some of them make use of formalism grounded in mathematics. It seems to be beneficiary and feasible if we elaborate a method that focuses on documents and their life cycle and places them into the context of Information Architecture of Information Systems. The adequate mathematical formalism that is capable to represent the complex relationships is the generalized *hypergraph* (Bretto, 2013).

In a contemporary information system, a phenomenon that can be called document-centric can be discerned. Much of the recent information systems can be typified as business-oriented information systems, i.e. the business processes, the workflows play an important role in the environment in which the information system is embedded. The business processes and documents are intertwined so that a document-centric approach with a formal grounding provides a helping hand for understanding the system, maintaining the consistency in a dynamic environment, sustaining the data security and privacy, assisting in verification and validation of the model at analysis, design, and operational season (Bell, 2008).

The scientific *contribution* of our research is the mapping of the concepts that are related to documents, business processes, and models of information systems onto a hypergraph structure that offers the opportunity for both static verification and operationalization in a graph database environment. On the created structure, a set of logic rules specified that provide an initial basis for formal verification, validation, and maintaining the consistency of the models. In the preceding conference paper (Molnár, Benczúr, & Béleczki, 2016b), the way for the application of description logics and a brief set of the defined logical axioms were described. In the recent paper, our scientific *contribution* is that the outlined approach in the previous paper is pursued further on and extended in the following areas: (1) Taking advantage of case studies (Molnár, Béleczki, & Benczúr, 2016; Molnár, 2017), the document-centric approach is examined in a public administration environment in the light of *Adaptive Case Management* (Swenson, 2011); (2) the mapping onto the proposed hypergraph structure of the components of *cases* (processes, procedures, input and output documents) is formalized; (3) the hypergraph structure for describing *cases* and relevant documents was reformulated as a concept hypergraph to enable the reasoning mechanism of description logics; (4) the logical axioms are augmented that are formulated in description logics; (5) the concept hypergraph and the strongly coupled logical rules made possible the conceptualization of matching between the components of models in *cases* and defining a qualitative measurement for the degree of matching; and (6) the first formal steps are taken towards modelling of dynamically changing business processes in all three seasons of their life cycles.

The implementation of the framework is taken place firstly on a hypergraph database (Iordanov, 2010). Although the typical problems of an open-source software occurred, there is an ongoing attempt to exploit the properties of graph databases and to work out a homeomorph mapping onto a bipartite graph to represent a bipartite, concept hypergraph.

In Section 2, we present the previous researches reported in the literature; in Section 3, we describe the required mathematical background; in Sections 4 and 5, we outline our method making use of the previous approaches in a document-centric approach, and Section 6 provides a summary and conclusions.

## 2. Literature review

The web-based and *Web Information Systems* (*WIS*) are the typical examples that make extensive use of the various document formats. The emphasis on Web technologies slowly diminished as the application of Web technology, at user interfaces, became commonplace. The WIS contains traditional information systems element and semi-structured XML documents, which behave actively beside other important parts of the system. The XML documents make use of active hyperlinks and other features of Web and Web services (Isakowitz, Bieber, & Vitali, 1998). A WIS can be considered as an information system that has an underlying database, or at least sets of data collection. The WIS can be accessed on the web from anywhere in the cyberspace via web browsers. The data and information that flow through the web interfaces and web pages are incorporated into documents. The documents as appropriate format for structuring both information and data can serve to query data from the information system or actualize the underlying data structure.

A systematic design approach to construct web-based applications was discussed in Köppen and Neumann (1999). The shown method made use of semi-structured and interactive documents represented by XML. Another paper presented an approach for a well-founded, concept-based modelling process for a website. For designing of WIS, Rossi presented a design procedure (Rossi, Schwabe, & Lyardet, 1999). To grasp the complex behaviour of information systems, the notion of the Enterprise Architecture gives a helping hand, namely Blokdijk's perception of information systems, Zachman's ontology, and TOGAF, all of them was created for information systems (Blokdijk & Blokdijk, 1987; Clark et al., 2001; MacKenzie et al., 2006; Zachman, 1987).

An information system supports business processes (Business Process Modelling, BPM) and is usually tightly coupled to another information system. A standard way to model business processes is the application of BPM methods. The information systems can also be perceived as a structure with underlying databases for structured, semi-structured, as well as unstructured documents (XML-based). The documents play an important role at the interface, interaction level, and at core activities of data processing. The integration level and the degree of reconciliation between business processes and the organization were analysed on the basis of ontologies and semantic approaches (Kő & Ternai, 2011). A model-driven and formal comprehensive approach for information systems modelling was depicted in Lukovic, Ivancevic, Celikovic, and Aleksic (2013). There are various input data format for communication to services: (1) HTML pages, (2) SOAP messages, (3) semi-structured and unstructured documents (XML-based) (Bernauer & Schrefl, 2004; Chiu & Bieber, 2001; Nam, Jang, & Bae, 2003).

In the contemporary information systems, the analysis and design problems can be formulated as follows: (1) the weight of communication with information systems shifted from the structured data towards semi-structured and unstructured data in the form of documents; (2) the data exchange between information systems and the organizational environment happens through services provided by some logical component of the information system that ingest the data that have been inputted. The input data, document, and the service that handles them are strongly interconnected at the level of business process management, furthermore at a logical level of modelling too so that these compounds of elements were considered as one unit by several researchers. Some approaches called that perception as *Business artifact* (Cohn & Hull, 2009; Hull, 2008). In this approach, the events, data, and services that process the data compose one entity so that these artefacts were the subject of formal modelling and operationalization of the activities. Furthermore, there were similar concepts that have been defined by disparate names, such as *adaptive documents* (ADoc) (Kumaran, Nandi, Heath, Bhaskaran, & Das, 2003) and *adaptive business objects* (ABO) (Nandi & Kumaran, 2005). However, none of them tried to connect the documents as inputs to the various architecture layers that contain design, implementation, and operation models; moreover, neither of them attempted to exploit the duality and dichotomy between business processes and documents that exist in models contained in the Enterprise Architecture. At the human–machine interface, the efficient and effective usage of documents was investigated that may provide a better understanding and perception of documents, whereby a suitable personal information system was created for the comfortable management of documents for the end-user (Indratmo & Vassileva, 2006).

There were some previous papers that tried to put the before-mentioned approaches into a unified framework by a semi-formal way (Molnár, 2014; Molnár & Tarcsi, 2011; Molnár & Benczúr, 2013). The Enterprise Architecture framework such as Zachman and TOGAF provided a supporting environment (Bent, Sante, Kerssens, & Kemmeren, 2008; MacKenzie et al., 2006; Zachman, 1987). Blokdijk's collection of Information System Models yielded a structuring guideline (Blokdijk & Blokdijk, 1987); moreover, the axiomatic design approach employed for the Information System environment offered an opportunity for a method that was interesting not only for theoretic modelling point of view but provided a chance to support practical design methods (Christensen, Curbera, Meredith, & Weerawarana, 2001).

## 3. 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 constituents participate sometimes in hierarchical and sometimes network-like relationships. These two kinds of relations are different from each other. The hypergraphs – as mathematical structure – seem to be apt to representing the interrelationships among the models, views, viewpoints, perspectives, and the overarching documents and business processes (Zachman, 1987).

We start with the basic definitions of hypergraphs to employ for depicting the before-mentioned complex relationships.

**Definition 3.1:** A *hypergraph* $H$ is a pair $(V, E)$ of a finite set $V = \{v_1, \dots, v_n\}$ and a set $E$ of nonempty subsets of $V$. The elements of $V$ are called vertices (nodes) and the elements of $E$ are called edges (Bretto, 2013).

**Definition 3.2:** *Generalized* or extended hypergraph. The notion of hypergraph is broadened so that some of the hyperedges are denoted – in certain cases – as vertices, thereby a generalized 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* (Bretto, 2013).

Considering a document model, a 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 represent the instance of the extension of a document type. In a document subpart hierarchy, a specific subpart of document may be denoted by a vertex within a specific 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 3.3:** A *directed hypergraph* is an ordered pair

$$\vec{H} = (V; \vec{E} = \{\vec{e_i} | i \in I\}), \tag{1}$$

where $V$ is a finite *set of vertices* and $\vec{E}$ is a set of *hyperarcs* with finite index set $I$. Every *hyperarc* $\vec{e_i}$ can be perceived as an ordered pair

$$\vec{e_i} = (\overrightarrow{e_i^+} = (e_i^+; i); \overrightarrow{e_i^-} = (e_i^-; i)), \tag{2}$$

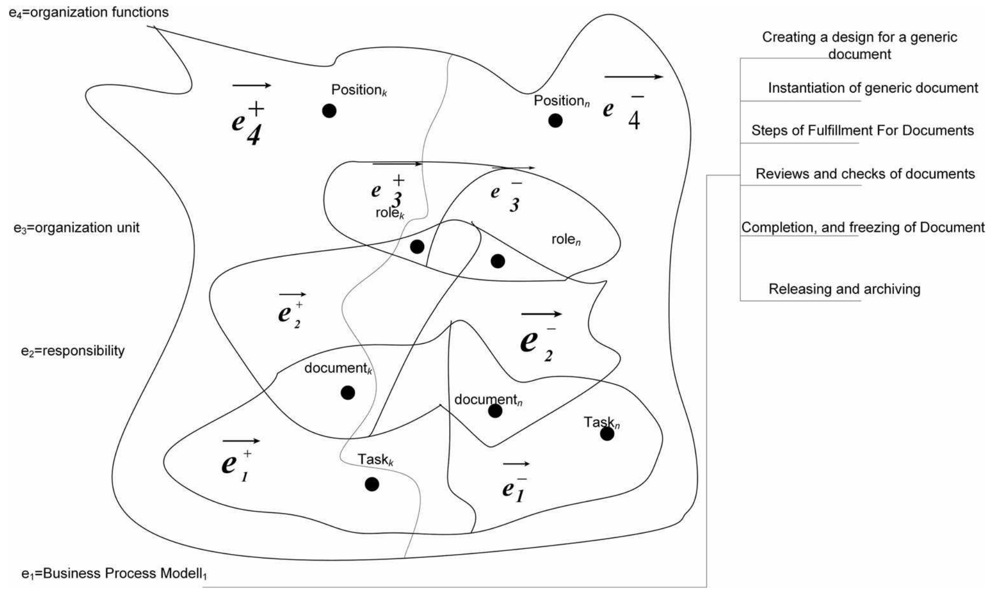
where $e_i^+ \subseteq V$ is the set of vertices of $\overrightarrow{e_i^+}$ and $e_i^- \subseteq V$ is the set of vertices $\overrightarrow{e_i^-}$. The elements of $\overrightarrow{e_i^+}$ (hyperedges and/or vertices) are called *tail* of $\vec{e_i}$, while elements of $\overrightarrow{e_i^-}$ are called *head* (Bretto, 2013). We may use as shorthand notation for ordered pairs, e.g. a vertex and a directed hyperedge as ordered pair $\langle v_i, e_j \rangle$.

The underlying graph representation is based on the hypergraphs and directed hypergraphs. The potential implementations of hypergraphs in a hypergraph database allow for linking attributes to vertices and to hyperedges too. 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, and to store compound structures along with their values (Ausiello, Franciosa, & Frigioni, 2001; Gallo, Longo, Pallottino, & Nguyen, 1993; Iordanov, 2010).

As an illustration of the basic concepts of directed hypergraph, an example can be seen in Figure 1 that makes sense of the representation for the domain by hypergraph. The essential characteristic is that vertices contain composite constituents that are themselves may be graphs; generalized hyperedge may contain other hyperedges – but not itself – and nodes. A detailed description about the *Architecture Describing Hypergraph* can be found in Molnár, Benczúr, and Béleczki (2016a, 2016b).

## 4. Formalized document-centric approach

In the case of an organization, we can imagine that there is a comprehensive document that is a representation – in conceptual sense – of 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



e4=organization functions
e3=organization unit
e2=responsibility
e1=Business Process Modell1
Positionk
Positionn
rolek
rolen
documentk
documentn
Taskk
Taskn
Creating a design for a generic document
Instantiation of generic document
Steps of Fulfillment For Documents
Reviews and checks of documents
Completion, and freezing of Document
Releasing and archiving

**Figure 1.** Directed hypergraph describing the relations among documents and tasks of business processes.

generic document. The hierarchical relationships can be described by *configuration hyperedges* ($\boldsymbol{E_C}$); the instances of a generic document member can be perceived as extensions and can be represented by extensional hyperedges.

A *generic document type* ***GDT*** is a *hierarchy of document types* ***DTH***. The elements of ***DTH*** can belong to a configuration hyperedge $e_{Ci} \in \boldsymbol{E_C}$ as vertices. The generalized hypergraphs allow that the vertices may appear as complex structures, as hyperedges. Therefore, a node 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.

Figure 2 shows a sample for explanation of the structure for recursive document structure through a case study example (Molnár, Béleczki et al., 2016). The hypergraph structure provides the opportunity to create a complex structure that can reflect flexibly the documents during input, processing, and output with connection to the tasks of the workflows. The data elements, the variable (either free or bounded), and the types of the elements of XML structures can be represented as nodes embedded into hyperedges and hyperarcs. The hyperedges and hyperarcs along with their interrelationship can describe the roles that are played by nodes and hyperedges because the generalized hypergraphs allow for inserting recursively hyperedges into each other. By this way, the documents can be described as an interconnected network of specific objects that depicts the static structure. However, the generalized hypergraph makes possible to represent the operations on each single object through recursively embedded hyperedges, whereby it mirrors the dynamical aspect of the interactions between documents and processes.

The direction of the hyperarc shows whether a document plays the *input* or *output* role in a context. The definition given above (see formula (2)) permits the differentiation between the *information* represented by the *head* and *tail* of a hyperarc and the

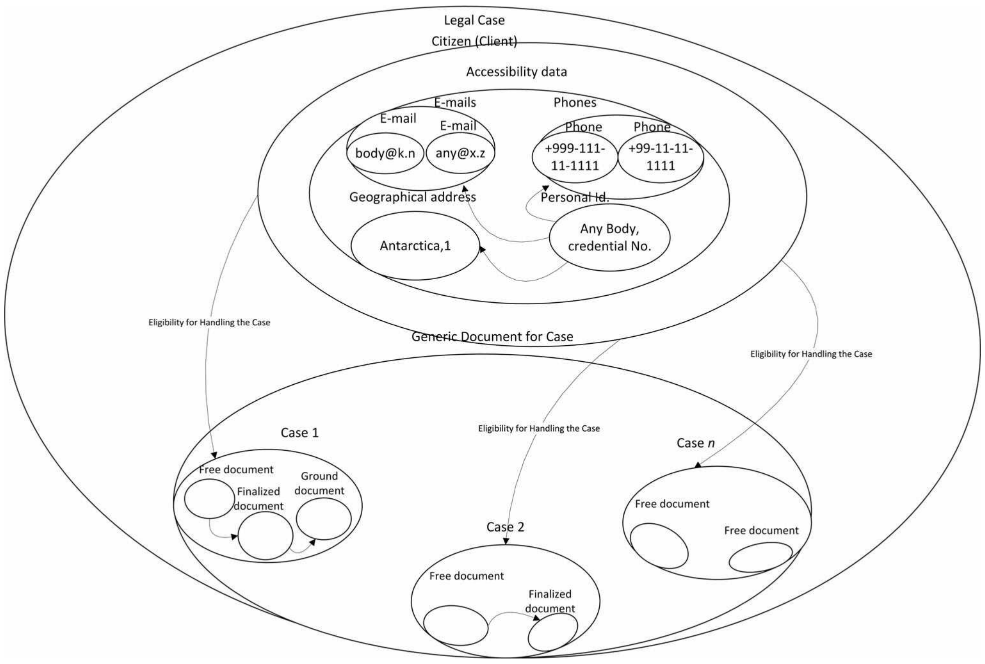


**Figure 2.** Hypergraph representation for documents in case management within public administration.

*information* that are represented in the form of nodes that are contained within the heads and the tails (Bretto, 2013).

**Definition 4.1:** The *Document Subhypergraph* consists of:

(1) A finite set of documents that are represented by vertices ***DOC*** = {$doc_1$, … ,$doc_n$};
(2) The documents contain variables, the variables belong to *attribute types* ***Attr*** = {$T_1$, … , $T_n$};
(3) The finite set of domains is ***DOMSET*** = {$D_1$, … ,$D_k$} that contains the domain of each single type, $T_{i,}$ *attribute type*;
(4) The relationship between a ***GDT*** *generic document type* hierarchy and its constituents document types belonging to a ***DTH*** *document type hierarchy* can be described by hyperarcs representing *is-a* relationships; the hierarchy is a mapping of supertype–subtype relationships between document types. The relationships can be deduced from the variables, their attributes, and the types of attributes.
(5) The relationship between a document $doc_i$ and a *document type* ***DT*** can be described by a hyperarc representing the *instance-of* relationship.

The concept of generic document type offers possibilities for the derivation of new document types from other document types that can be regarded as templates. The derivation rules can be formalized as logical statements that assist to create a slightly different document type through modifying the structure of an existing document type; and when an instance of a free document of the document type is created then the derivation rules may control the life cycle of the instance of the document type. A document type belonging to a document type hierarchy having at a top level a generic document type may contain business rules in the form of predicates, data retrieving and calculation rules.

As an illustration of the above-mentioned facts, we can investigate the case management and processes of legal procedures in public administration. A generic document type can be the 'Legal Case', i.e. a comprehensive document that incorporates the concept of issues that might be initiated by a citizen (Figure 2). There is a generic document type – 'Legal Case' – that contain document types for i*dentification of citizens* involved in the *case* and checking the eligibility, i.e. his/her/their rights, to handle and manage the specific legal issue. The example suggests that the usual personal data and credentials can be used to identify the actual person(s) attached to the *case*. Even a specific legal case may involve several different document types through the networking of legal procedural rules. These document types referring to *sub-cases* should be within the hierarchy and scope of the actual generic document type. During the document evolution stages, the generic document type and document types that contained in the document hierarchy instantiated into documents with free variables; then variables are valuated till the documents reach the finalized status; however, the finalized state does not mean an end-state in the practice, and in public administration. The finalized document should be modified according to specific requirements till it achieves the end-state that may be called *ground-document*, when all the free variables are valuated and no change can be executed on the document, i.e. it is in a frozen status. Any modification can be carried out only on a new instance of the document type, starting or continuing the development of the document at a certain point of its life cycle.

A 'legal case' can be defined as a collection of tasks, actions, processes, and content in support of a specific public administration objective. *Cases* can involve multiple flows, tasks, and content, and sometimes have sub-cases within them. An integral part of a *case* is the related documents that evolve along with the strongly coupled tasks within processes and move within the data flows and go through transitions (Van der Aalst, Weske, & Grünbauer, 2005).

**Definition 4.2:** A *case* comprises of tasks, procedures, processes, and content that is embodied in the form of documents to achieve certain goals.

A *case* can be represented by a generalized hyperedge that contains:

- The nodes for the relevant document types and the hyperarcs that represent the transformation of documents:
    - $h \in \boldsymbol{E}_{Configuration\_Document}$, $h \subset V_{Doc}$, where $V_{Doc} \supseteq \{OGDT\}$, where *OGDT* designates the *overarching generic document*, that is the supertype of all other document types and their instances. The set of hyperedges *Configuration_Document* embodies the arrangement and structure of documents that can be described by hyperedges that contain vertices that represent the relevant parts of documents, a single hyperedge represent a specific arrangement of constituents for documents.
- $V_{Configuration} = \cup\, h_i$, where $h_i \in \boldsymbol{E}_C$, and $\cap\, h_i = \emptyset$, where $h_i \in \boldsymbol{E}_C$. $\boldsymbol{E}_C$ consists of the *configuration hyperedges*. Each $h_i \in \boldsymbol{E}_C$ is a simple hyperedge, i.e. containing only vertices, it does not contain complex structures and other hyperedges. All $h_i \in \boldsymbol{E}_C$ can be labeled unambiguously.

- $V_{Process} \supseteq \{BPM, \{external\_evironment\}\}$, where *BPM* designates the overall *Business Process Model* consisting of process models, the *external_environment* refers to the outside world that is typically the source of *stimulus* that is generated by either humans or any other systems;
  - The set of arcs (directed edges of graphs) **A** is partitioned into subsets $\boldsymbol{A}_{Doc\_Target}$, $\boldsymbol{A}_{Process\_Target}$, $\boldsymbol{HA}_{Interaction}$, where $\boldsymbol{A}_{Doc\_Target} \subseteq V_{Configuration} \times V_{Doc}$, $\boldsymbol{A}_{Process\_Target} \subseteq V_{Configuration} \times V_{Process}$. The *Doc_Target* and *Process_Target* designate those documents and processes that are on the 'output side', on the *head* of hyperedges ($\overrightarrow{e_i^-}$); meanwhile, the *Interaction* denotes the information exchange between processes and documents, or any other entities.
  - $\boldsymbol{HA}_{Interaction} \subseteq V_{Model} \times V_{Doc}$, $\boldsymbol{HA}_{Interaction} \subseteq \boldsymbol{E}_D$ (*directed hyperedges*); the interaction between certain *process models* and *specific documents* can be expressed by *hyperedge* $h \in \boldsymbol{HA}_{Interaction}$.

An arc, a directed hyperedge ($e_{doc\_target_i} \in A_{Doc\_Target}$) can describe that a set of documents or multiset of documents can be an input and another set of documents or multiset can be the output. The transforming tasks and processes can be contained in another hyperedge ($e_{process\_target_j} \in A_{\Pr ocess\_Target}$), the input, the output, and the relevant tasks of processes can be represented through a directed hyperedge $e_{interaction_k} \in HA_{Interaction}$.

$\boldsymbol{E}_D$ is a set of *hyperarcs*, i.e. *directed hyperedges*; the hyperarc $\vec{e_i} \in \boldsymbol{E}_D$ can be as follows (see Definition 3.3):

- $\vec{e_i} = \langle v_j, \vec{h} \rangle = (\overrightarrow{e_i^+} = (e_i^+ = v_j, i); \overrightarrow{e_i^-} = (i,\ e_i^- = h))$, where $v_j \in V$ set of *vertices*, $j \in J$ index set, and $\vec{h} \in \boldsymbol{E}_G$; $\boldsymbol{E}_G$ is made up of *grouping hyperedges* that symbolize various structuring principles on components, e.g. *views*, and *perspectives* in architecture describing approaches (Zachman, 1987); they denote interrelationships between certain models and pieces or parts of documents, 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;
- $\vec{e_i} = \langle v_j, \vec{h} \rangle = (\overrightarrow{e_i^+} = (e_i^+ = v_j, i); \overrightarrow{e_i^-} = (i,\ e_i^- = h))$, where $v_j \in V$, $j \in J$ index set, and $\vec{h} \in \boldsymbol{E}_C$;
- $\vec{e_i} = \langle v_j, \vec{h} \rangle = (\overrightarrow{e_i^+} = (e_i^+ = v_j, i); \overrightarrow{e_i^-} = (i,\ e_i^- = h))$, where $v_j \in V$, $j \in J$ index set, and $\vec{h} \in \boldsymbol{E}_E$ (extensional hyperedges);
- there does not exist two hyperarcs $\vec{e_i} = \langle v_j, \vec{h} \rangle$ and $\vec{e_k} = \langle v_l, \overrightarrow{h'} \rangle$ that either $\vec{h}, \overrightarrow{h'} \in E_C$ or $\vec{h}, \overrightarrow{h'} \in E_E$, 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 in the following way: a vertex may belong to a configuration structure (either document or process model) or it may belong to an extension that represents the instantiation of either a document or a process model.

In modelling perspective, a *case* can be perceived as a collaborative process that includes several actors and roles out of public administration (in the case study) and citizens. The participating public officers can belong to various branches and sectors of public administration; in the case of commercial companies, the actors may be associated with multiple departments, groups within an organization. The *case* can be considered as content-intensive, i.e. a *case* contains lots of document in different states, at various points in their life cycles. In the process modelling aspect, a *case* is semi-structured or unstructured, i.e. a *case* may be enhanced by ad hoc tasks dynamically during its life cycle either being in the analysis, design, or operational phase. The processes of a *case* include several tasks and actions that are ordered in the hierarchy or network. The *document type hierarchy* and the structure of a *case* should be correlated, considering their structures, namely their graph representation within a hypergraph. The dynamism is the immanent characteristics of a *case* in the sense that each single case differs from each other as the case handling depends on the specific circumstances; these circumstances cause changes in *cases* so that each 'season' or phase of the *case* – either generic type level or at a point in the life cycle of the actual instance – requires modifications in the activities, documents, etc.

As it can be concluded from the above-described formal definition, a *case* typically consists of the following components: documents, variables (data fields), data items, content of documents (structured, semi-structured, unstructured), processes and their constituents as tasks and actions. A *case* that has – as a major objective – to accomplish a goal of the stakeholders (in the example both citizens and public administration) possesses other accessories, e.g. the relevant policies, status during evolution of the cases, the history of development of the case, audit trails and other logs for monitoring and tracking, and finally reporting and data analytics mechanisms.

The *case* management and processing within an overarching information system environment are primarily an *intertwined process* and *document* handling problem. One of the inherent attributes of the case management is the dynamic characteristic. During analysis and design time of an information system, the intention is to cover the events, processes, input, and output data/documents.

As we have seen in our case study (Molnár, Béleczki et al., 2016), the *case* – a legal procedure to handle an issue related to a life cycle events of citizens – is broadly defined in the sense of documents and processes linked to the specific issue. The generic documents, the document type hierarchy defines the required documents and data sets; the intensional documents, the entrenched rules, the processes, and tasks linked to a *case* through rules are partially defined in advance (Figure 2). The combined sets of generic document types and the connected process templates provide a framework that gives the opportunity for flexible amendment in all *three seasons* of the life cycles of processes and document types (see Figure 3) (Bell, 2008). The concrete steps of tasks and actions in tandem with documents are determined by a combination of business events, human decisions, and business rules (Kossak et al., 2016). The human interaction has relevance at analysis and operational seasons as well. In public administration, there are laws, legal rules, directives that govern the workflow joined to a *case*. However, there are subtleties, details that necessitate the dynamic amendment on the templates and instances (both documents and processes). There are different legacy systems that are sector specific within public administration; these systems either demand input data from processes or yield some output data to processes and documents involved in a case. Because of the silo-like organization of public administration, the collaboration between different branches is limited to share information and handling a *case*. These facts initiate ad hoc, *dynamic* modifications within workflows, on the involved documents and processes at both analysis and operational seasons. A design problem of workflows and

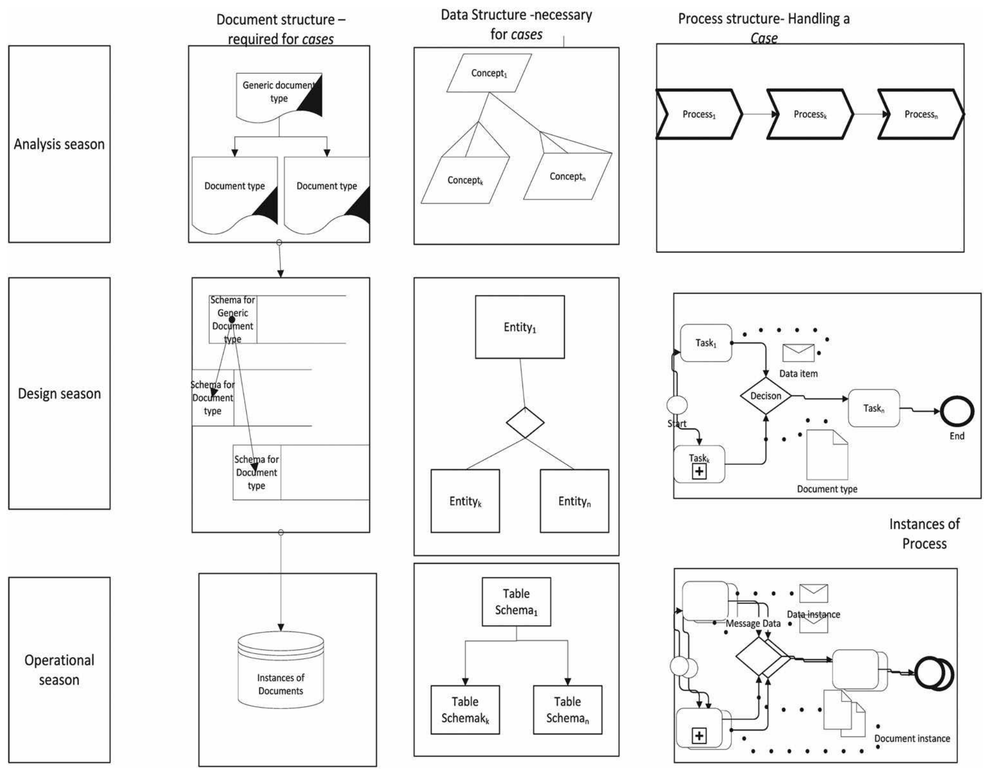


**Figure 3.** Architecture of documents, data, and processes for case management.

business processes is to provide a consistent and timely feedback of about a *case* to the involved citizens. A formal model grounded in graph theory provides the chances to reconcile the various aspects in an integrated and consistent view. The systematic description of the complex relationships makes possible to monitor, track, and manage the *cases*.

A specific requirement is to set up a reporting scheme on *cases* that exploits data analytics and data mining techniques to provide a comprehensive picture on the actual situation, history, and tendencies (Molnár, 2017). Another important obligation is the compliance that is the processing of *cases* should conform several legal rules. These rules should be kept during all the seasons of *case* life cycles, namely, analysis, design, and operation. The compliance means a kind of risks that should be minimized during processing of *cases*.

The formalized model can solve the above-mentioned problems with *case management* that appears in the document-centric information system with special emphasis on business processes.

## 4.1. Description Logics and business processes for cases

One of the most common approaches of formalization is the use of some mathematical-logical language. The description logics belong to the theories of mathematical logics, and their purpose is to create a formal knowledge representation (Baader, 2003; Blokdijk & Blokdijk, 1987). Compared to propositional calculus (or propositional logic), the expressiveness of description logic is higher, and it has a more effective 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 nodes and links are relationships, e.g. the semantic network – can be related to the theory of hypergraphs. In both cases, nodes can be used to define concepts, and links can be used to characterize the relationships among them. On 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 (based on the description languages varieties) 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, such as concept union (U), full existential qualification (E), cardinality restriction (N), or complex concept negation (C).

We can exploit the description logic for formulating logical rules and axioms that bear on documents and processes. The advantage of the use of a generalized hypergraph is that we can associate the more static description logic axioms to hyperedges and hyperarcs, and intensional documents; moreover, the more dynamic inference rules can be attached as well to them. Therefore, formal model for an information system is well prepared for accepting changes resiliently that happen dynamically as it cannot be assumed that all processes can be accurately planned so that the exceptional event that causes dynamic changes in discreet time should be perceived as normal situation; however, the exceptions cannot be forecasted and cannot be designed in advance. The document-centric approach fits to the adaptive case management approach as the focus is on the information content of documents and the focus is not only on the processes themselves. The dynamic aspect of business processes put emphasis on enhancing workflow representation through knowledge management that can emerge in formal modelling as the mapping the knowledge into formal representation, e.g. into description logics. The concept of dynamic case management permits the formal representation of the collaboration work either structured, semi-structured, or unstructured; furthermore, it makes possible that the roles played by humans can be considered during interactions between humans and automated tasks (Kossak et al., 2016). The document-centric representation for case management helps organizations to cut across the silo-like arrangement of business units and the related information systems.

As we have seen previously, a case unifies the processes, sub-processes, tasks, documents, and the relevant data elements (Figure 3). The business processes are built up through analysis of the anticipated outputs that are in the form of *ground-documents*, then through the input documents that are originally in the form of *free-documents* populated by free-variables or placeholders. The data items along with their data types are stored in data collections then they are retrieved from a database that contains the data collections in their partitions. The variables, placeholders should fit onto each other in input, output documents, and data collection.

**Definition 4.3:** The *extent* of a *case* consists of the relevant business process, the set of tasks within the process, the flow of the input, output documents, and a transformation *T* that maps the input set of input documents *Inp* onto the output set of documents *Outp*, $T\!: Inp \rightarrow Outp$,, $Inp = \{doc_{Inp_1}, \dots, doc_{Inp_n}\}$, $Outp = \{doc_{Outp_1}, \dots, doc_{Outp_n}\} \subseteq \boldsymbol{DOC} = \{doc_1, \dots, doc_n\}$ that are represented by vertices in the *Document Subhypergraph*.

**Definition 4.4:** *Elucidation* of the *Document Subhypergraph*, i.e. additional properties are stated to have the subhypergraph made subject of Description Logic and business rules.

(1) Each vertex $doc_i$ in ***DOC*** *Document Subhypergraph* represents a document that contains a set of variables and placeholders owning attribute types and referring to data items.
(2) Each hyperedge *e* containing vertices $doc_i$ $i = 1, 2, \dots, n$ describes a relationship between documents.
(3) A generalized hyperedge *ge* may contain hyperedge *e* that represents relationship between certain set or multiset of documents, thereby hyperedge *ge* represents a set of relationships among documents.
(4) Each vertex *v* in a hyperedge *e* has meta-attributes for reasoning, e.g. a declaration that vertex *v* is mandatory, or optional element within the relationship represented by hyperedge *e*.

The hypergraph allows for a multi-dimensional representation of facts and knowledge, syntactic and semantic level information that is codified into the documents. Thus, the graph representation makes allowance for a uniform framework in which unary (collection-like, e.g. containment), binary, and *n*-ary relationships, atomic elements (attributes, data items), and complex relationships can be represented. This approach can underpin the modelling activities of information systems when the life cycle demands requirement analysis, specification, verification, and validation, moreover reasoning on the underlying structure. The hypergraph model can be the subject of graph transformation algorithms. As we have mentioned in Definition 4.2, the hypergraphs can be labelled unambiguously, not only the configuration hyperedges but other hyperedges and arcs as well, thereby this definition of hypergraph can be considered as *directed labeled graph*. The directed labeled graphs can be frequently used for depicting semantic nets, simple statements in first-order predicate logic, Datalog logical statements (Ullman & Widom, 2009), RDF triples, Open Graph (Facebook, 2017), and Knowledge Graph (Google, 2017). This property of the directed labeled hypergraph can be exploited: (1) the hypergraph in default interpretation permits the representation of *n*-ary relationships, the labels of hyperedges express relations among the nodes that are contained in an hyperedge, the labels of hyperarcs can represent, e.g. relational database tuples where the key define the other attributes, or link input and output format of documents that may have several variables (Definition 4.2); (2) the generalized hypergraph makes possible the depiction of nested hyperedges representing nested relationships, expression, terms, and logical formulas; (3) higher order logics, description logics allow for description both the instances and the logical relations between them (A-box, T-box); (4) the generalized hypergraph provides the opportunity that labels of hyperedges and hyperarcs can be used as nodes whereby they can be member of logical statement and inferences.

### 4.2. Description Logic and document-centric approach

To illustrate the use of the Description Logic in a document-centric environment, we give some examples below:

- With ***Parameter*** $\sqsubseteq$ (***Free Variable*** $\sqcup$ ***Bound Variable***) notation, we describe that a document parameter can be free or bound variable.
- ***Parameter*** $\sqsubseteq \exists$ ***is_part_of***. (***Document Fragment***) means that a document fragment consists of parameters, and ***Document Fragment***. $\exists$ ***is_derived.*** (***Free_Document***) means that the document fragments are derived from unprocessed free documents.
- ***State.P*** $\sqsubseteq \exists$ ***has_successor.*** (***Action State.Q***) means that *Q* action-state follows the *P* state.
- The following line describes that an action-state needs free variables to work with: ***Action State*** $\sqsubseteq \exists$ ***has_free_variables.*** (***Document***); ***has_free_variables*** $\equiv \geq 1$ ***is_free_variable*** $\sqcap$ ***is_free_parameter***. ***Parameter.***
- The objective to be achieved by a case can be formulated: ***Goal of Case*** $\sqsubseteq \exists$ ***has_post_condition. Task***
- ***Task*** $\sqsubseteq \exists$ ***Model.Business_Process.*** Defines that a task should belong to an existing business process model.
- ***Intensional_document*** $\sqsubseteq \exists$ ***has_rules. Document_type***. It determines that intensional documents are active documents containing inferences and axioms.
- ***Document*** $\sqsubseteq \exists$ ***Generic_Document_Type*** $\sqcup$ ***Document_Type*** $\sqcup$ ***Instance_of_Document_Type***. The definition of documents that appear as nodes in hyperedges.
- ***Free_Document*** $\sqsubseteq \exists$ ***has_free_variables.*** (***Instance_of_Document_Type)*** $\sqcap \neg$ ***has_bounded_variables.*** (***Instance_of_Document_Type).*** It defines that a document instance at the beginning of its life cycle is a free document containing only free variables, data fields to be filled out.
- ***Variable*** $\equiv\forall$***usedInDocument. (Document)*** $\sqcap\ \forall$***usedInTask. (Task)*** $\sqcap$ ***usedInData_Collection. (Data_Collection)***. If a task within a process uses a variable, then the variable should appear in the related document and partition of the underlying database.
- ***Document*** $\sqsubseteq \exists$ ***has_responsibility. (Role)*** $\sqcap$ ***usedInTask. (Task).*** It declares that a document is manipulated in a task should belong to roles within the organization that have responsibility for manipulation of documents. It is important for checking and controlling the security, safety, and compliance aspects.
- ***Data_Collection*** $\sqsubseteq \exists$ ***Database***
- ***Data_Collection*** $\sqsubseteq \exists$ ***has_data_type. (Data_Item)***. It clarifies the data collection concept, i.e. data collection consists of data items that can be considered as atomic concept and they have data types, thereby data collection constitutes a definite part of the underlying database.
- ***has_variable. (Documents)*** $\sqsubseteq \exists$ ***usedInData_Collection. (Data_Collection)*** $\sqcap$ ***usedInTask. (Task).*** The variables to be filled-in are part of the variables that are manipulated by the specific tasks and the variables can be mapped onto data fields/items that comprise a data collection.

The output of a well-designed formalization with description logics of an information system (represented by a hypergraph) is in a machine-readable format. Thereby it creates the opportunity to use various frameworks and tools to evaluate the model. By this way, it is possible to effectively optimize the information system even in the early model-development phase.

## 5. Information architecture and documents of *cases*

Besides the essential documents, information systems can be described by various models that are ordered into a reasonable structure by the Enterprise Architecture approach. Above, we have already discussed the process models relating to *cases*, although several other models and views exist in an information system environment that should be taken into account in a comprehensive representation.

The models' descriptions appear usually in semi-structured document formats as XML and/or JSON that offers a chance for uniform treatment of documents and models of information systems. As structuring principles for models of information systems, we can use Zachman's ontology and/or TOGAF (Bent et al., 2008; Zachman, 1987). The set of relations among models and the internal structure of models plays an essential role.

The models can be arranged into three meta-groups namely *organization*, *documents,* and *activities/processes* related models. For modelling, the relationships and interactions among these three meta-groups and the underlying collections of data are significant. The models, documents, concepts of information systems, and a node representing the external environment compose a hypergraph that embraces all important parts of the application domain that may be called as *System Hypergraph*.

In the case of documents, a directed hyperedge can express the input and output roles of documents that they may play within activities of business processes. The document may be attached to organization units and actors through a *responsibility hyperedge* (labeled directed hyperedge). The variables of documents may be connected to data vertices of $\boldsymbol{D}$ that is organized into reasonable partitions that are represented by nodes contained in hyperedges that can be mutually mapped to specific data collections. These subhypergraphs may be called *Sub-system Hypergraph*s. Between the models, a refinement relation can be identified within an architectural perspective and represented by a directed hyperarc *is-a-refinement* (Zachman, 1987). The documents and their structures can be described by documents model.

**Definition 5.1.** *Models* of information systems 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 supertype 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 $\cap\, h_i = \emptyset$, where $h_i \in \boldsymbol{E}_C$ *configuration hyperedge*.

The *Architecture Describing Hypergraph* depicts the architecture of information system within an organization through the conceptual framework of the Enterprise Architecture, i.e. the layers of the models, the perspectives and aspects of models, and viewpoints of stakeholders (Zachman, 1987; TOGAF Bent et al., 2008). The hypergraph provides a unified and uniform representation tool for the heterogenous artefacts that are an integral part of an information system, e.g. as an architectural component, as an analysis, design, implementation, and operational model. The hypergraph representation is not only a static description method despite it, the mathematical properties of the hypergraph representation provide opportunities for analysis and assessment of the consistency, integrity, completeness of the functional requirements in a dynamic environment as well.

Thus, the above-defined expressions articulate the fact that the configuration hyperedges represent the structure of artefacts of models and documents in the form of structural constituents as vertices.

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

The directed edges ($\boldsymbol{E}_D$, the arcs) map a complex structure to a vertex within the hypergraph, namely the configuration of elements to a vertex that describes either a document or a model.

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

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

- $\boldsymbol{E}_C$ can be partitioned into two subsets $\boldsymbol{E}_{Configuration\_Document}$ and $\boldsymbol{E}_{Configuration\_Model}$.

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 node either in $V_{Model}$ or in $V_{Doc}$, its value represented by a link to a $d \in \boldsymbol{D}$ 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 node $v_i \in h$ 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 $h' \subseteq h$
      - (a) Notation: $\mathbf{E}_{\mathrm{Super\_doc}} = \mathrm{V}_{\mathrm{Doc}} \setminus \{\mathrm{OGDT}\}) \times \mathbf{E}_{\mathrm{Superclass}} \subset \mathbf{E}_{\mathrm{D}}$;
      - (b) Notation: $\mathbf{E}_{\mathrm{Super\_model}} = ((\mathrm{V}_{\mathrm{Model}} \setminus \{\mathrm{EA}, \{\mathrm{external\_evironment}\}\}) \times \mathbf{E}_{\mathrm{Superclass}} \subset \mathbf{E}_{\mathrm{D}})$;

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 of our modelling approach, we make a distinction between the two top super-classes, namely *OGDT*, the *overarching generic document*, and *EA*, the overall *Enterprise Architecture*. The conditions above specify the transitivity of the *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 hyperedges*);
- 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 nodes) is $h \in \boldsymbol{E}_{Attribute\_Set}$ if $h \subset D$. The following statement is valid as well: $\cup\, h_i = \boldsymbol{D}$, $h_i \in \boldsymbol{E}_{Attribute\_Set}$. The hyperarcs $h \in \boldsymbol{E}_{Attribute\_Set}$ are used to represent the attribute domains and the associated values.
- The hyperarc $h \in \boldsymbol{E}_{Extension}$, if $h \in \boldsymbol{E}_E$ ($h$ set of nodes) 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 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 hyperedges in $\boldsymbol{E}_D$ (hyperarcs) 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 the organization.
  - 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.

One of the main goals is to model the dynamic aspects of the business processes through a set of models to maintain and to enforce the consistency, integrity, and compliance of models. Our approach of formal modelling is tried to react and to handle several dynamic aspects of information systems that originate from changes of documents and these modifications lead amended processing. The description logic along with the hypergraph representation provides an infrastructural background that makes possible to recognize inconsistencies, to regulate the necessary modifications through a controlled environment, to restrict access to the change management as it is proper. The document-centric approach requires the capability that the data processing should evolve with the progressing documents. One of the design principles that could be applied is the *late binding*, i.e. the resources for tasks within business processes can be utilized at the time the work will be carried out. The intensional document type yields an appropriate mechanism for supporting dynamism in data processing through the rules embedded in documents and the rules fire when it is necessary, i.e. dynamically fit the actual environment.

**Definition 5.2.** *Ontology for cases including documents***.** An ontology can be defined as labelled bipartite hypergraph $O = (C, R, E, l)$, where $C$ is a set that represents the concepts in the form of nodes and generalized hyperedges, $R$ is a set that describes the relations in the form of hyperedges, generalized hyperedges, and hyperarcs, $E \subseteq C \times R$, the edges between the bipartite hypergraph depicting the edge relation $L$ between the two sets of vertices that can be coloured by two different colours (bipartite or bi-coloured hypergraph), and $l$ is the labelling of hyperedges, hyperarcs, and edges (Baader, 2003; Bretto, 2013).

**Definition 5.3.** *Characteristics of processes.* The characteristics of processes can be perceived as functional and non-functional attributes, given an $n$-tuple of attributes $\langle C^1, C^2, \ldots, C^n \rangle$, the *values* of the $n$-tuple can be given by $\langle v_1, v_2, \ldots, v_n \rangle$, where $v_i$ can be a numerical parameter on a non-functional property either lower or upper bound depending on the context, or some prerequisite, feature of the functional requirements.

**Definition 5.4.** *Process of a case*. Given an ontology $O = (C, R, E, l)$, a *process* is defined as a triple $p_i = I_{p_i}, O_{p_i}, C_{p_i}$ where:

$I_{p_i} \subseteq O$ represents the set of concepts (documents, variables, and their structuring) that semantically depicts the input of the process.

$O_{p_i} \subseteq O$ represents the set of concepts (documents, variables, and their structuring) that semantically depicts the output of the process.

$C_{p_i} \subseteq O$ is an $n$-tuple of attributes $\langle C_i^1, C_i^2, \ldots, C_i^n \rangle$, where every $C_i^j$ designates the matching *value* of the $n$-tuple of attributes $\langle C^1, C^2, \ldots, C^n \rangle$.

During modelling and analysing of information systems, then design and operational time, verification and validation is required for configuration of *cases* to maintain consistency, integrity, confidentiality, and completeness of the model in all three seasons of life cycles of each process. Documents along with the data collections can be interpreted as semantic input and output entities through the ontology. The matching between input and output entities can prove the compliance of the process to the actual requirements at a certain point in time.

**Definition 5.5.** *The degree of matching of elements of cases* (Bellahsène, Bonifati, & Rahm, 2011; Euzenat & Shvaiko, 2013).

*Exact matching.* An input (document along with its variables and attributes) $i_{p_i} \in I_{p_i}$ of a process $p_i$ within a *case* matches an output $o_{p_j} \in O_{p_j}$ with a degree of exact matching if their representation as concepts in the concept hypergraph can be considered as equivalent ($i_{p_i} \equiv o_{p_j}$).

*Plug-in matching*. An input (document along with its variables and attributes) $i_{p_i} \in I_{p_i}$ of a process $p_i$ within a *case* matches an output $o_{p_j} \in O_{p_j}$ with a degree of *plug-in* matching if $i_{p_i}$ representation as concept in the concept hypergraph is a direct sub-concept of $o_{p_j}$, i.e. $i_{p_i} \sqsubseteq o_{p_j}$.

*Subsumption matching*. An input (document along with its variables and attributes) $i_{p_i} \in I_{p_i}$ of a process $p_i$ within a *case* matches an output $o_{p_j} \in O_{p_j}$ with a degree of *subsumption* matching if $i_{p_i}$ representation as concept in the concept hypergraph is an indirect sub-concept of $o_{p_j}$, i.e. $i_{p_i} o_{p_j}$.

*Subsumed by matching*. An input (document along with its variables and attributes) $i_{p_i} \in I_{p_i}$ of a process $p_i$ within a *case* matches an output $o_{p_j} \in O_{p_j}$ with a degree of *Subsumed by* matching if $i_{p_i}$ representation as concept in the concept hypergraph is a direct super-concept of $o_{p_j}$, i.e. $i_{p_i} \sqsupseteq o_{p_j}$.

*Failure matching*. When none of the aforementioned set of preconditions is fulfilled then the matching attempt is failed and designated by $i_{p_i} \perp o_{p_j}$.

## 6. Model evaluation

The above-outlined model is implemented in a graph database (Béleczki & Molnár, 2017) to create a chance for investigation. As we have mentioned earlier, the original plan for implementation in a hypergraph database (Iordanov, 2010) was not successful in every respect. The main reason is the lack of continuous maintenance and development of the hypergraph database management system. For that reason, other graph database management system was selected, and the hypergraph representation was mapped onto a bipartite graph that provides the chance to exploit the mathematical and logical properties of the hypergraph.

One of the case studies contained a large set of documents and the database that stored them in relation schemas, furthermore processes for retrieving, modifying, maintaining, and creating the documents (Molnár, Béleczki et al., 2016). A well-defined and significant part of the database transformed into the hypergraph representation. The basic hypothesis was that the hypergraph representation provides a tool for analysis of the original design of information systems on the basis of documents.

In the evaluation of the model, we have followed the principles of the Design Science Research (Wilde & Hess, 2007). Our investigation demonstrated that the idea is viable, either of the expected relationships or their lack of can be discerned easily through the graph structures. The seamless integration of the documents and data collection within databases can be realized, the compliance between the input, output documents and their data content with the processes can be enforced, i.e. the discrepancies can be highlighted and then can be corrected.

However, the rule set formalized in description logic and attached to the adequate hyperedges needs extension and refinements. The essential characteristics of the hypergraph as transversal, acyclicity, cliques and the significant properties of information systems should be linked to each other to yield a meticulous analysis of the original design and provide the opportunity to keep the required constraints on the consistency, integrity, and confidentiality. The mutual mapping of the properties in graphs and information systems is an ongoing research.

The original intention of the research was to build up a model for analysis, design, implementation, and operation of information systems that communicate with the external environment through documents. For that reason, the model is *apt to* document-centric, and to document intensive information systems. These information systems and their communication interfaces are typically implemented by web technologies; they communicate with the external world with documents in web interfaces and standardized messages including and transporting documents. During our research, we have concentrated on two case studies – one of them is in public administration that provides assistance for citizens in official procedures through descriptions of *legal cases* related to life events; the other one is the Electronic Learning Management Systems that provides web-based active documents for students, educators, and administrators.

## 7. Conclusion

In this paper, we proposed an Architecture Describing Hypergraph as representation for Enterprise Architectures and related Documents, then we extended the hypergraph representation towards a concept hypergraph. The suggested descriptive method takes advantages of 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 structure of the hypergraph can be interpreted as an ontology, concept hypergraph, thereby it opens the way for semantic methods and reasoning.

The ontology as an extra layer on the Architecture Describing Hypergraph provides a logical tool set for formal verification, namely the exploitation of description logics. The proposed information system modelling through hypergraph includes *cases* besides the outlined document-centric structure. The definition of *case* coalesces the notion of process, documents, input and output data, functional and non-functional properties of models into a unified framework. The focus on the notion of *cases* makes possible the application of the proposed approaches on the one hand in the heterogenous environment, e.g. enterprises, public administration, and on the other hand the handling of the situation in a uniform way.

The outlined approach can also be considered as a formal background to analyse and design information systems. The documents play important roles in information systems during 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 information systems even in a dynamically changing environment.

The novelty of our approach is that it addresses: (1) the functional correctness of the models starting on the documents on the input side with a document type hierarchy; (2) consistency and completeness checking through models of the document input, output, and processes of *cases*; (3) provides the opportunity to take into account the functional and non-functional properties that can be described as *concepts* and 'roles' in the sense of description logics; (4) the alignment between the model elements through the degree of matching can be assessed by the mathematical properties of hypergraphs and the fulfillment of logical statements.

Our ongoing research activities target at formal and mathematics-based computational approach that can be operationalized above an appropriate graph database (hypergraph database, or other ones), then at evaluating the efficiency and effectiveness of the approach that utilizes methods and algorithms of data science to provide a sound empirical investigation.

## Disclosure statement

No potential conflict of interest was reported by the authors.

## Funding

This work was supported by European Commission [grant number EFOP-3.6.3-VEKOP-16].

## Notes on contributors

***B. Molnár*** is Associate Professor (Dr. habil., Ph.D. in Technical Informatics) at Eötvös 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 has published several scientific and professional papers and worked as 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*), *The European Journal of Applied Economics*.

***A. Béleczki*** is Ph.D. student at Eötvös Loránd University of Budapest. His research interest covers Information Systems and Modelling, Big Data, Data Mining. He teaches Database Management Systems. He has published his first paper in December, 2015, about formal modelling of document-centric information systems.

***A. Benczúr*** 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, Modelling 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.

## ORCID

*B. Molnár* http://orcid.org/0000-0001-5015-8883

## References

Ausiello, G., Franciosa, P. G., & Frigioni, D. (2001, October). Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. In *Italian conference on theoretical computer science* (pp. 312–328). Berlin: Springer. doi:10.1007/3-540-45446-2_20

Baader, F. (2003). *The description logic handbook: Theory, implementation and applications*. Cambridge, UK: Cambridge University Press.

Béleczki, A., & Molnár, B. (2017, November 9). *Storing hypergraph-based data models in non-hypergraph data storage*. AIS 2017: 12th international symposium on applied informatics and related areas. Óbuda University Alba Regia Technical Faculty, Székesfehérvár, Hungary.

Bell, M. (2008). *Service-oriented modeling (SOA): Service analysis, design, and architecture*. Hoboken, NJ: Wiley.

Bellahsène, Z., Bonifati, A., & Rahm, E. (Eds.). (2011). *Schema matching and mapping*. Heidelberg: Springer.

Bent, H. V. D., Sante, T. V., Kerssens, D., & Kemmeren, J. (2008). *TOGAF, the open group architecture framework*. Zaltbommel: Van Haren Publishing. Retrieved from http://www.opengroup.org/togaf/

Bernauer, M., & Schrefl, M. (2004). Self-maintaining web pages: From theory to practice. *Data & Knowledge Engineering*, *48*(1), 39–73.

Blokdijk, A., & Blokdijk, P. (1987). *Planning and design of information systems*. London: Academic Press.

Bretto, A. (2013). *Hypergraph theory: An introduction*. Berlin: Springer.

Chiu, C.-M., & Bieber, M. (2001). A dynamically mapped open hypermedia system framework for integrating information systems. *Information and Software Technology*, *43*(2), 75–86.

Christensen, E., Curbera, F., Meredith, G., & Weerawarana, S. (2001). *Web services description language (WSDL) 1.1*. Retrieved from http://www.w3.org/TR/wsdl

Clark, J., Casanave, C., Kanaskie, K., Harvey, B., Smith, N., Yunker, J., & Riemer, K. (2001). *ebXML Business Process Specification Schema Version 1.01*. Technical report, UN/CEFACT and OASIS. Retrieved from http://www.ebxml.org/specs/ebBPSS.pdf

Cohn, D., & Hull, R. (2009). Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Engineering Bulletin*, *32*, 3–9.

Euzenat, J., & Shvaiko, P. (2013). *Ontology matching*. Heidelberg: Springer.

Facebook, Open Graph Stories. Retrieved from https://developers.facebook.com/docs/opengraph

Gallo, G., Longo, G., Pallottino, S., & Nguyen, S. (1993). Directed hypergraphs and applications. *Discrete Applied Mathematics*, *42*(2–3), 177–201.

Google, Knowledge Graph. Retrieved from https://www.google.com/intl/es419/insidesearch/features/search/knowledge.html

Hull, R. (2008). Artifact-centric business process models: Brief survey of research results and challenges. In R. Meersman & Z. Tari (Eds.), *On the move to meaningful internet systems: OTM 2008* (pp. 1152–1163). Berlin: Springer.

Indratmo, I., & Vassileva, J. (2006, April). No more isolated files: Managing files as social artifacts. In G. Olson & R. Jeffries (Eds.), *CHI'06 extended abstracts on human factors in computing systems* (pp. 899–904). Montreal: ACM.

Iordanov, B. (2010, July). HyperGraphDB: A generalized graph database. In *International conference on web-age information management* (pp. 25–36). Berlin: Springer. doi:10.1007/978-3-642-16720-1_3

Isakowitz, T., Bieber, M., & Vitali, F. (1998). Web information systems. *Communications of the ACM*, *41* (7), 78–80.

Kő, A., & Ternai, K. (2011). *A development method for ontology based business processes*. eChallenges e-2011 conference, Florence, Italy.

Köppen, E., & Neumann, G. (1999). *Active hypertext for distributed web applications*. IEEE 8th international workshops on enabling technologies: Infrastructure for collaborative enterprises, 1999 (WET ICE'99), Stanford, CA, USA, pp. 297–302. IEEE. doi:10.1109/ENABL.1999.805216

Kossak, F., Illibauer, C., Geist, V., Natschläger, C., Ziebermayr, T., Freudenthaler, B., … Schewe, K. D. (2016). *Hagenberg business process modelling method*. Springer: International Publishing Switzerland.

Kumaran, S., Nandi, P., Heath, T., Bhaskaran, K., & Das, R. (2003). *ADoc-oriented programming*. Symposium on applications and the internet (SAINT), Orlando, FL, USA, pp. 334–343.

Lukovic, I., Ivancevic, V., Celikovic, M., & Aleksic, S. (2013). DSLs in action with model based approaches to information system development. In *Software design and development: Concepts, methodologies, tools, and applications* (pp. 596–626). IGI Global. doi:10.4018/978-1-4666-2092-6

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R., & Hamilton, B. A. (2006). *Reference model for service oriented architecture 1.0*. OASIS Standard, 12, p. 18.

Molnár, B. (2014). Applications of hypergraphs in informatics: A survey and opportunities for research. *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae: Sectio Mathematica*, *42*, 261–282.

Molnár, B. (2017). *Proposal for application of data science methods in E-government: An investigation about the combination of available techniques for performance measurement*. The 6th international conference on electronic government and the information systems perspective EGOVIS 2017. doi:10.1007/978-3-319-64248-2_11

Molnár, B., Béleczki, A., & Benczúr, A. (2016). Application of legal ontologies based approaches for procedural Side of public administration. In *International conference on electronic government and the information systems perspective* (pp. 135–149). Porto: Springer.

Molnár, B., & Benczúr, A. (2013). Facet of modeling web information systems from a document-centric view. *International Journal of Web Portals (IJWP)*, *5*(4), 57–70.

Molnár, B., Benczúr, A., & Béleczki, A. (2016a). Formal approach to modelling of modern information systems. *International Journal of Information Systems and Project Management*. Retrieved from http://www.sciencesphere.org/ijispm/archive/ijispm-040404.pdf

Molnár, B., Benczúr, A., & Béleczki, A. (2016b). A model for analysis and design of information systems based on a document centric approach. In *Asian conference on intelligent information and database systems* (pp. 290–299). Berlin: Springer.

Molnár, B., & Tarcsi, A. (2011, September). *Architecture and system design issues of contemporary web-based information systems*. 2011 5th international conference on software, knowledge information, industrial management and applications (SKIMA), Benevento, Italy, pp. 1–8. IEEE. doi:10.1109/SKIMA.2011.6089978

Nam, C. K., Jang, G. S., & Bae, J. H. J. (2003). An XML-based active document for intelligent web applications. *Expert Systems with Applications*, *25*(2), 165–176.

Nandi, P., & Kumaran, S. (2005). Adaptive business objects – a new component model for business integration. *Proceedings of international conference on enterprise information systems*, Miami, FL, USA, pp. 179–188. INSTICC.

Rossi, G., Schwabe, D., & Lyardet, F. (1999). Web application models are more than conceptual models. In P. P. Chen, D. W. Embley, J. Kouloumdjian, S. W. Liddle, & J. F. Roddick (Eds.), *Advances in conceptual modeling, LNCS* (vol. 1727, pp. 239–252). Berlin: Springer. doi:0.1007/3-540-48054-4_20

Swenson, K. (2011). *Taming the unpredictable: Real world adaptive case management: Case studies and practical guidance*. Lighthouse Point, FL: Future Strategies.

Ullman, J. D., & Widom, J. (2009). *Database systems: The complete book* (2nd ed.). Upper Saddle River, NJ: Pearson.

Van der Aalst, W. M., Weske, M., & Grünbauer, D. (2005). Case handling: A new paradigm for business process support. *Data & Knowledge Engineering*, *53*(2), 129–162.

Wilde, T., & Hess, T. (2007). Forschungsmethoden der Wirtschaftsinformatik. *Wirtschaftsinformatik*, *49*(4), 280–287.

Zachman, J. A. (1987). A framework for information systems architecture. *IBM Systems Journal*, *26*(3), 276–292.