information

MDPI

Review

Formal Ontologies in Information Systems Development: A Systematic Review

Martina Husáková and Vladimír Bureš *

Faculty of Informatics and Management, Department of Information Technologies, University of Hradec Králové, 500 03 Hradec Králové, Czech Republic; martina.husakova.2@uhk.cz

* Correspondence: vladimir.bures@uhk.cz

Received: 18 December 2019; Accepted: 22 January 2020; Published: 27 January 2020

Abstract: Computational ontologies are machine-processable structures which represent particular domains of interest. They integrate knowledge which can be used by humans or machines for decision making and problem solving. The main aim of this systematic review is to investigate the role of formal ontologies in information systems development, i.e., how these graphs-based structures can be beneficial during the analysis and design of the information systems. Specific online databases were used to identify studies focused on the interconnections between ontologies and systems engineering. One-hundred eighty-seven studies were found during the first phase of the investigation. Twenty-seven studies were examined after the elimination of duplicate and irrelevant documents. Mind mapping was substantially helpful in organising the basic ideas and in identifying five thematic groups that show the main roles of formal ontologies in information systems development. Formal ontologies are mainly used in the interoperability of information systems, human resource management, domain knowledge representation, the involvement of semantics in unified modelling language (UML)-based modelling, and the management of programming code and documentation. We explain the main ideas in the reviewed studies and suggest possible extensions to this research.

Keywords: formal ontology; information system; conceptualisation; software development; software design; semantic web; UML; OWL

1. Introduction

There are many definitions of information systems (IS). According to [1], an information system is perceived as a collection of: "Interrelated components working together to collect, process, store, and disseminate information to support decision making, coordination, control, analysis, and visualization in an organization." An information system involves output that tries to solve specific problem(s) occurring in an organisation. Problems often arise when an organisation feels that its productivity is not as expected, or when pursuing competitive advantage. Software development is focused on building software products effectively and of the desired quality, where a customer and a software company are satisfied with a final product. Generally speaking, decision making and problem solving is an inherent part of the methodologies used for the systematic development of software products, including IS. These activities are often related to categorisation and conceptualisation [2]. The categorisation of "things" into groups reflects recognition of the most important features of these "things" and the methods used to deal with them [3]. Conceptual models of these "things" are automatically created in our minds. They consist of series of concepts representing abstract or concrete entities with relationships between them. They can also represent our beliefs, desires, or intentions, which are transformed into concrete actions [4]. Conceptualisation and categorisation are related. Categorisation is realised during the development of conceptual models. The ability to solve specific problems

autonomously, proactively, and efficiently is integrated into artificial intelligent systems to assist people in their everyday activities. Computational ontologies are used to encode information or knowledge into these systems. These formal structures model very general or very specific concepts which reflect the knowledge in particular application domains. Generic (upper-level, top-level) and domain-specific ontologies are thus distinguished. Generally, they can be used for [5]:

- information or knowledge sharing between people,
- information or knowledge sharing between machines (intelligent agents),
- information or knowledge sharing between people and machines,
- reusing general or domain-specific pieces of knowledge,
- analysis of the application domain,
- development of knowledge bases for knowledge-based systems.

Ontology originated in philosophy, where it tries to answer on the most fundamental questions about human being via metaphysics. American computer scientist J. McCarthy first introduced ontologies into computer science as an approach to expressing common sense knowledge: routine (often inaccurate) knowledge which we receive during our everyday activities [6]. Knowledge-based systems should not fail due to the ignorance of common sense knowledge. There are various definitions of ontologies in computer science. T. Gruber and W. Borst provide two of the most commonly known definitions of ontology and the attributes required in ontological engineering. T. Gruber notes [7]: "Ontology is the explicit specification of the conceptualisation." Explicitness means that ontologies should be expressed in a way that they are not hidden only in human minds but available to others for use. Conceptualisation has already been defined above. It identifies the most important concepts to integrate into an ontology with respect to the purpose of the model. W. Borst extends this definition as [8]: "Ontology is the formal explicit specification of shared conceptualization." This definition is closer to the ideas of the semantic web initiative introduced by T. J. Berners-Lee in [9], where he declares: "The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users. The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation." Ontologies should provide this well-defined meaning which should be formally expressed in the languages where Resource Description Framework (RDF) [10], Resource Description Framework Schema (RDFS) [11,12] or Ontology Web Language (OWL) [13,14] is the most known and used. The formal nature of ontologies helps intelligent machines to interpret the meaning of concepts. An ontology is the result of consensus between interested parties and provides a vocabulary of concepts which can be shared for mutual communication and understanding.

The main question behind our systematic review, involves how beneficial formal ontologies are in information systems development. Information systems development covers a wide spectrum of activities. The activities (phases) taken into account during development depend on a particular methodology. As shown in [15], there are many and various software development methodologies. The main aim of this review is not to present or compare all these methodologies, but to focus on the application potential of formal ontologies mainly during the analysis and design of software systems (information systems), together with coding and documentation preparation. These phases correspond closely to phases in the majority of methodologies used in software development [16–19]. Several reviews or documents describe the role of ontologies in information systems. Gonzalez-Perez presents ontological thinking from the perspective of software engineering. Their chapter is mainly focused on the relationships between ontologies, models and metamodels [20]. The differences between metamodels and ontologies are also presented in [21] where the author tries to identify strategies to ensure compatibility between them. Guan, Levitan, and Kuhn investigate the role of ontological structures in the development of accounting information systems, but they also provide an overview of where ontologies can be used in computer science and information systems [22]. Beydoun et al.

advocate the usefulness of ontologies for information systems and provide an approach to selecting the most suitable ontology for realising a software project [23]. These authors, and the authors in Section 3, do not provide a systematic review which would provide a deeper analysis of ontologies and their role in information systems development.

This paper is structured as follows: Section 2 provides an overview of the methods used during the systematic review process, which is partially based on the mind mapping technique. Section 3 is the core of the paper where specific studies are examined and used to explain the role of formal ontologies in IS development. Section 4 summarises the added value that the ontologies bring to information systems development. The end of this section introduces future directions, suggesting where the ontologies could be used in education about IS development. Section 5 concludes the paper.

2. Methods

2.1. Search Strategy

V. Larivière, S. Haustein, and P. Mongeon published a research paper where they analysed almost 45 million documents of various types, which were indexed in the Web of Science over the period 1973–2013 [24]. They identified the top five publishers with the highest number of scientific documents focused on the natural and medical sciences and social science and humanities. We used their conclusion to identify publishers whose studies were then used in the systematic review. We examined studies from the following publishers: Elsevier (ScienceDirect and Scopus), Wiley-Blackwell, Springer (SpringerLink), Taylor and Francis, and Sage. Figure 1 provides a more detailed description of the search process. It corresponds to the PRISMA 2009 flow diagram.

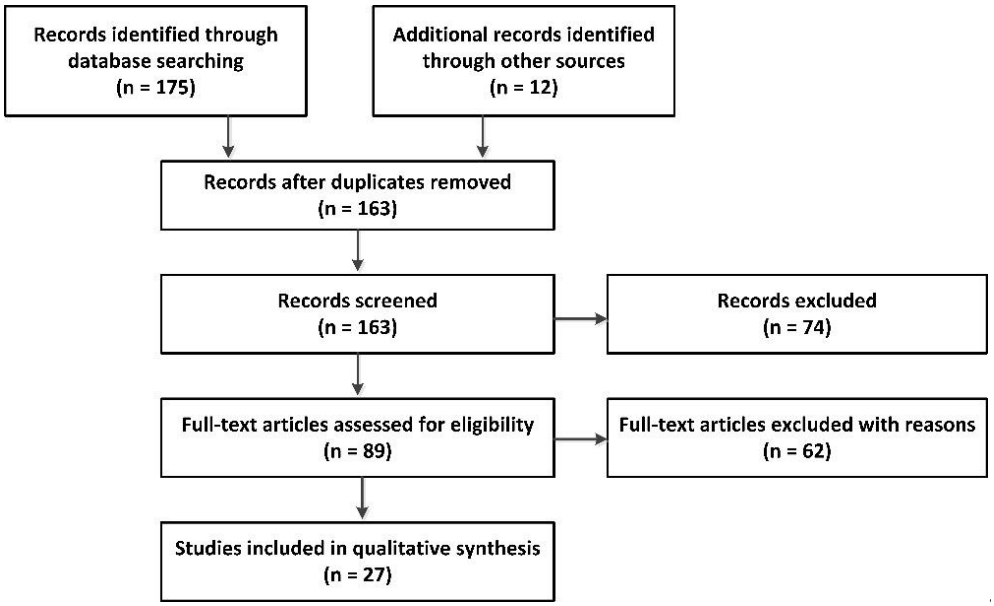

Figure 1. Search strategy used for the selection of particular publishers in the systematic review.

2.2. Selection Criteria

Information is a scientific peer-reviewed open access journal, which has celebrated 10 year of publication, and so we analysed documents published between years 2009 and 2019. We selected original documents of all types which were published in English. We extracted documents that offered full access to content. The following keywords and noun phrases were used when searching the online databases: ontologies in information systems, ontology-driven information systems, ontologies in design or analysis of information systems, ontologies and unified modelling language (UML), and ontologies and information systems in case studies. An advanced search using the

default settings of the online database was applied when searching all online databases. Keywords and noun phrases were searched anywhere in the sources; no restrictions were specified about their positioning. All 187 documents were included in predefined working categories for the fast identification of their content. The complete list of categories, together with a first author, year of publication, title of the document and publisher are included in the electronic supplemental file (see SystematicReview-Filtration-Overview.xslx). Information about duplicate documents is also given in this working file.

2.3. Information Extraction

After the elimination of duplicate documents, they were identified as either relevant or non-relevant based on reading the abstract and conclusion, although in some cases additional parts of a document also had to be examined when the information in an abstract or in a conclusion was not sufficient. A file labelled as SystematicReview-Filtration-Overview.xlsx was extended on the relevancy of the documents. Two additional working documents were created: the most important facts in the selected studies, and the potentially new ideas of the authors of the systematic review, which may extend such research in the future.

Formal Ontologies in Information Systems Development describes a wide spectrum of application domains which are represented by ontological structures or which are embedded in the information system. Many solutions were identified, suggesting how ontologies can help in information systems analysis or design. This is why the mind map was developed. The mind map is "a strategy" to non-formally visualise ideas for solving a specific problem [25]. This creative technique was firstly introduced by T. Buzan in the 1960s and is actively used during brainstorming [25–27]. The mind map was created for an overview of the interconnections between ontologies and information systems. The *Ontologies for information system development* concept is at the center of the mind map and additional relationships are attached to this central concept, i.e.,

- an application domain (for which purpose the ontology is used),
- the first author of the document with year of publication,
- the category of an information system (if it is mentioned),
- brief description of a document.

Generally, the readability of a mind map decreases as more concepts are included in the map. Two mind maps were developed. The first included 51 documents which were categorised into groups where some thematically overlapped: interoperability, UML (unified modelling language), requirements management, agent-based systems, development (not directly related to the UML), domain knowledge representation, information extraction, semantic search, consistency checking of knowledge models and a specific domain category named medicine. These categories were merged during relevant documents filtration. Some of the papers in these categories were reviews themselves, some offered a very general view of the domain examined and some were only partially related to information systems development as such (agent-based systems, information extraction, semantic search). These documents were excluded from the systematic review, but some are mentioned for additional information about ontological engineering. The second mind map reflects this exclusion and the overlap of the above categories. This mind map contains the categories that fully and fundamentally correspond to our research question. Figure 2 depicts the final version of the mind map, which includes the most fundamental categories and interconnections between formal ontologies and information systems. Only relevant and useful documents are included. The first map is not included in this review due to difficulties with readability. It is only a "working mind map".

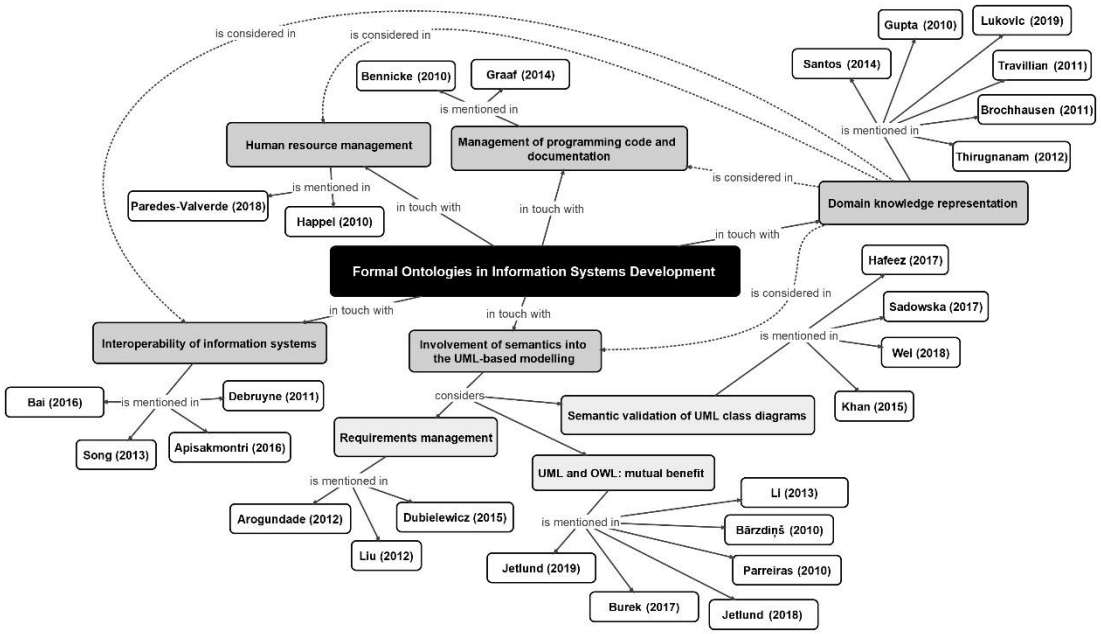

Figure 2. The mind map for systematic review preparation.

2.4. Information Synthesis

Specific clusters reflecting the application potential of ontologies in information systems development arose during formation of the first mind map. The majority of these clusters were merged due to similarities between them. These final clusters were also based on the contents of discovered documents. Only the most thematically similar and relevant documents are taken into account in the systematic review. The most interesting and key studies were identified. These aspects formed the final collection of documents examined in the systematic review. It was not possible to compare all the relevant and interesting studies against each other. They are all related to ontological and software engineering, but relate to different aspects of modelling, and so only studies contained in specific clusters are mutually compared. The following aspects were taken into account in their examination: the type of solved problem, approach used for problem solving, applied technologies and results.

2.5. Number and Type of Studies Included

Of 89 potentially relevant unique studies, 27 were eligible for inclusion into the systematic review (see PRISMA guidelines in Figure 1). Some studies were excluded from the systematic review for these reasons:

- thematically, they were reviews themselves, or
- they were only partially relevant to software engineering (i.e., they were more thematically relevant to agent-based software engineering, natural language processing and semantic searches).

3. Results

We identified five domains where ontologies add significant value in information systems development. Ontologies are applied during the preliminary phases of information systems development, and especially where the user requirements and conceptual UML-based models development are specified. These two uses are closely related to domain knowledge representation, where an analyst has to be familiar with the application domain which is going to be covered by the information system. Domain knowledge representation is a general purpose of domain-specific ontologies. Although it is naturally related to all studies in this systematic review, we feel that

it is necessary to include it as a separate group. Ontologies also contribute to the management of human resources, programming code and documentation. The interoperability of information systems is a wider and also more general domain where the ontologies are often noted. The following subsections explain the role of ontologies in the above domains in more detail. These subsections are ordered according to how we think about information systems from the perspective of the phases of their development.

3.1. Interoperability of Information Systems

Information systems are often connected to heterogeneous data or information sources which are built and managed by different parties. Communication with other information systems is often inevitable and is critical for efficient decision making and problem solving. The problem is that communication between these systems is often based on different conceptual schemas (terminologies) which arise from different firm cultures, experience, or habits. Formal ontologies can provide a solution for problems with interoperability issues.

Web-based information systems are often developed and managed autonomously, which supposes that autonomous negotiations between web services are provided by these systems. Web services are the result of different designers or stakeholders, and this often leads to disunity in the semantics of shared concepts. Debruyne and Meersman [28] use ontologies for the formalisation of the social processes between stakeholders and designers. A formal ontology is used practically for the annotation of components of information systems. Thanks to these annotations, web services (software agents) are able to better recognise these components and efficiently communicate with information systems using these annotations.

Song, Zacharewicz, and Chen [29] tried to solve semantic heterogeneity and interoperability in enterprise information systems (EIS). They introduced an ontology-driven framework using a semantic information layer which plays the role of mediator in the information exchange between heterogeneous EIS. It is possible to query various data sources without with to deal with each data source directly. The techniques of ontological engineering, mainly including ontology extraction from the relational database, ontology enrichment and alignment, were applied.

Financial and accounting analyses provide large and complex information which is valuable for managerial and business decision making. The problem is that various heterogeneous parties are connected to financial analyses which are based on different information sources. Specific XML-based standards, representing financial and business processes, are provided for information exchange between various financial institutions, especially Financial Products Markup Language (FpML), Research Information eXtensible Markup Language (RIXML), e-business XML (ebXML), and Interactive Financial eXchange (NewsML, IFX). eXtensible Business Reporting Language (XBRL) is an XML, XMLS and XLink-based format for simplification of the automated exchange of financial information. Bai, Koveos and Liu [30] present an OWL ontology-based extension of the XBRL format for the provision of more insightful semantics into the expression of financial concepts and for the realisation of financial analysis.

Formal ontologies can also solve problems in emergency information systems. To have information in the right place at the right time is critical during emergency situations. The problem is that these pieces of information are distributed by different parties (volunteers, governmental institutions, paramedics, afflicted persons). Communication is often not unified or is provided in different forms (formats). Information is not complete and changes dynamically. This mix often provides a distorted view of the whole situation and draws conclusions from such inconsistent information is problematic. Apisakmontri et al. [31] apply an ontological approach for the unification of concepts used in disaster and emergency management. They introduce a formal pivot ontology named Humanitarian Aid for Refugees in Emergencies (HARE) to solve interoperability in heterogeneous emergency systems and a methodology for its development. Their ontology is based on the existing Semantic Web for Earth and Environmental Terminology (SWEET) [32], Descriptive Ontology for Linguistic and

Cognitive Engineering (DOLCE) [33], Suggested Upper Merged Ontology (SUMO) [34] ontologies, and the WordNet vocabulary. The methodology is based on purpose identification, the capture of the ontological concepts, coding and integration (see details in [5]).

3.2. Human Resource Management

Human resource management (human capital management) is a continuous process generally focused on the management and development of human resources: employees. It covers a wide spectrum of activities, including the planning, organising, directing, and controlling of human resources. Software projects are as good as the employees who use them. It is not surprising that the quality of human resources strongly affects the quality of software products. Semantics which is integrated in the ontological structures can assist in human resource management.

Software projects can be realised by a firm as a self-contained entity without the assistance of the other companies. Collaborative software development (CSD) is supported by the project activities of teams that are often distributed in different locations, and from different companies and cultures (with different habits). This approach is often applied when a company deals with larger projects, when software is more complex. Research in CSD is focused on how to effectively connect these heterogeneous teams in order to produce the desired software product. Happel, Maalej, and Seedorf [35] advocate the usefulness of ontologies in CSD, especially in coordination activities, knowledge sharing and development. The semantic annotations of test-cases or emails can solve problems with finding the right balance between information overload and the efficiency of awareness measures in software projects. Ontologies can facilitate the integration of heterogeneous information, mainly in a semantic manner. Development teams should also be flexible, as there are often very frequent changes in software products. The authors claim that the ontologies are able to reflect dynamic changes, but it has to be said that ontology is a static structure. An ontology can be extended using procedures (e.g., rules) which can reflect the dynamic nature of software products. Ontologies can be extended using a query mechanism enabling the " ... generation of check lists—a popular tool in agile teams." As the authors also claim, specific information from, for example, release notes, checklists or management reports, can be automatically extracted with the assistance of ontologies, but we note that this is cannot be realised with only the ontology itself, and that natural language algorithms often have to be used for this purpose. Ontologies are knowledge-based structures which can provide a unified vocabulary of terms which is collaboratively developed by stakeholders in order to use "the same language" during software development [35]: "Ontologies can help to explicitly capture contextual information (such as the system configuration when a bug occurred) and give developers a more precise, unambiguous vocabulary to express certain information)." Semantic wikis can also bring added value to the management of software projects, such as in receiving relevant answers to questions related to a concrete software project. The web pages which are included in wikis contain semantically rich information which can assist during the querying of these web pages. More relevant answers with more details can be provided, and if the semantic wiki contains a specific inference mechanism, then additional (often hidden) knowledge can be given in the output.

Paredes-Valverde et al. [36] present a very interesting application focusing on the use of ontologies in human resource management. Human resources play a fundamental role in the success of software products. Human resource managers try to select the most suitable employees for a software project, mainly based on previous experience with software development. The authors of [36] introduce an ontology-based decision support system which selects the best candidate on the basis of their past experience with software project and actual project requirements. A formal ontology models software requirements specifications, and the required competencies of candidates (language skills, technologies). Semantic similarity is computed using the software requirements specification from previous projects and the requirements of the actual project. A semantic indexing approach is used for this calculation.

3.3. Domain Knowledge Representation

There are various definitions of ontologies in computer science. Horrocks explains that the ontology [37]: " ... introduces vocabulary describing various aspects of the domain being modelled and provides an explicit specification of the intended meaning of the vocabulary by describing the relationships between different vocabulary items." Rotondo notes that [38]: " ... ontology is the attempt to formulate an exhaustive and rigorous conceptual schema within a given domain. This is a hierarchical data structure that contains all of the relevant entities, the relationships between them, the rules, axioms, and constraints of the domain." Pinet states that an ontology should primarily contain [39]: "a vocabulary of terms, a set of term definitions that identify concepts and fix the term interpretation, a modelling of the domain of interest to represent relationships between concepts and an agreement of a community of ontology users about term definitions and the domain structure." Roussey et al. propose classifications of ontologies in [40]. An ontology can thus have a non-formal (e.g., mind maps), a semi-formal (e.g., Unified Modelling Language—UML, see 3.4.) or a more formal (e.g., Ontology Web Language—OWL [13]) structure.

The majority of these definitions share one fundamental aspect: an ontology is a conceptual model reflecting the most fundamental concepts under discussion. It models consensual knowledge which is integrated into information systems where it provides the "conceptual backbone" enriched with semantics. This semantic layer assists in the deeper expression of the concepts, which are then used by information systems to provide meaningful feedbacks to users. The previous two subsections indicated that these semantics-based conceptual schemas can be used in business, accountancy, emergency management or software engineering. This list of application domains is far from complete. It would be almost impossible to mention all applications, but medicine has to be included, because the systematic exploration of scientific databases showed that one of the largest groups of documents involves knowledge representation in medicine; medicine-related information systems.

Gupta, Condit, and Qian [41] introduce the BioDB multi-model system, which is able to manage heterogeneous biological data and information where different query processing operations are used for different categories of biological data. The system operates with relational, graph-based and tree-based information models, which are extended by ontological annotations providing meaning for these data. As the authors claim: "An ontology-enhanced system is a system where ad hoc data is imported into the system by a user, annotated by the user to connect the data to an ontology or other data sources, and then all data connected through the ontology can be queried in a federated manner."

Travillian et al. [42] present an interesting application of formal ontologies in the development of an information system used in comparative medicine. Comparative medicine studies particular species through comparison with other species. Their comparative anatomy information system allows users to compare the phenotypes of specific organisms across species at various levels of granularity. The frame-based structure represents the anatomical structures of these species. A frame is a structure modelling " ... a statement that something exists (a declaration) and it collects all relevant properties of the object in one place" [43]. A query mechanism is used to answer questions on the similarities and differences between the compared species.

Advancing Clinico-genomic Trials on Cancer (ACGT) is an EU co-funded project focusing on improving medical knowledge discovery with the assistance of grid technologies and ontological engineering. Brochhausen et al. [44] present ACGT-MasterOntology (ACGT-MO)—an application ontology which supports data integration across different terminologies used by different countries, disciplines, and languages. The ontology is used for the classification of medical documents (e.g., clinical reports, microbiological processes, or findings) where the natural language processing-based extraction of medical terms from domain-specific publications was applied for master ontology validation.

Thirugnanam, Ramaiah, and Sivakumar [45] present a disease information system that provides a diagnosis on the basis of symptoms. A formal OWL ontology models system-wise diseases (circulatory and digestive) and their symptoms. The knowledge-based system uses this ontology and formal rules for provision a possible diagnosis. Formal rules are written in the Semantic Web Rule

Language (SWRL) language [46] which combines facts of the ontology with user input (symptoms) for diagnosis specification.

Clinical family histories are valuable tools in the diagnosis of patients. The majority of health information systems do not provide tools using intelligent or sophisticated methods for manipulation with clinical family histories inside of these systems. Santos et al. [47] introduce a clinical pedigree information system named OntoFam. This system uses a formal ontology Family Heath History Ontology (FHHO) which models knowledge about haemophilia and haemophilic patients. SPARQL Protocol and RDF Query Language (SPARQL) [48,49] is a formal query language ordinarily used for the extraction of knowledge from an ontology. This language is used for the automatic inference of family relations.

Conventional information systems use relational databases for data storage and management. Relational databases require a stable data model. In some cases, knowledge of the system is dynamic and it is necessary to deal with its evolution. Dynamic knowledge is also involved in the diagnosis of adolescent scoliosis—a disease which exhibits as a sideways curvature of the spine. Lukovic at al. [50] present a very interesting study where they introduce the ScolioMedIS information system, which assists in the visualisation, diagnosis and monitoring of the adolescent idiopathic scoliosis. This system uses OBR-Scolio ontology—a knowledge-based structure which models knowledge about scoliosis. OBR-Scolio ontology is based on the OBR (Ontology of Biomedical Reality) OWL ontology which is used for building biomedical ontologies. ScolioMedIS ensures the complete management of this formal ontology, including the creation, deletion and editing of the ontological concepts.

3.4. Involvement of Semantics Into the UML-Based Modelling

Engineering requirements is a subarea of systems engineering, where the scope of a software product is formulated with the assistance of end users. It is an early phase in the software development process, where the attributes and behaviour of a system are defined (functional requirements), together with how these functions will be realised by a system (non-functional requirements). The UML is a standardised modelling language used in the analysis and design of software [51,52]. It provides a collection of diagrams for modelling the static and dynamic aspects of a system. These diagrams are then used for coding. The UML provides a use case diagram consisting of a collection of use cases where each one represents one functionality of a system. A misuse case is the negative form of a use case—a situation which should not happen in a system [53]. We have to be aware of security incidents, such as misuse cases, which can negatively affect a system.

Arogundade et al. [54] point out that the terminology behind use cases and misuse cases is not unified. Many different persons with various perspectives contribute to requirements specification, and semantic inconsistencies can thus be integrated into requirements. The authors present the Use-Misuse Case Ontology (U-MCO) which makes the meaning of use cases/misuse cases more accurate. The SPARQL query language is used for answering questions on use cases and misuse cases.

Requirements are not only specified in the preliminary phases of software development, but often noted during the maintenance of a software. Post-development change requests usually arise a long time after a system is developed. Discrepancies can be found if post-development requests are compared with the existing system. The manual detection of these conflicts is time-consuming when there are huge number of requests. This inconsistency can be detected by formally written rules using a formal ontology. Liu and Yang [55] introduce an ontology-based blog that is able to automatically detect conflicts between post-development requests and existing system.

Business modelling is aimed at understanding the environment in which a software will exist. Analysts and developers are in touch with a terminology which is used by a company for which software is developed. Dubielewicz et al. [56] use the ontology for modelling business classes, instances, taxonomical relationships, domain-specific associations and the hierarchy of these relationships. These ontological elements are translated by an analyst into a UML class diagram, which can be additionally refined by an expert who understands a business domain. This translation can be partly or fully

automated. The authors claim that: "This approach is recommended if the domain knowledge of analysts is not sufficient to properly define a domain model, for example in case of difficulties in communicating with prospective users of the system."

If an analyst has a solid notion about software product requirements, deeper analysis can be achieved with the next UML-based diagrams. Diagrams are not created as independent "containers". They relate to each other. As an example, the instantiation of objects is problematic when UML classes, the modelling structural aspects of a system, contain inconsistencies. UML sequence diagrams model behaviour across one use case. Their methods are taken directly from UML class diagrams. Logical contradictions can arise during UML-based. Their manual detection is time-consuming. Khan and Porres [57] use an ontological approach for the validation of UML-based models, especially UML class diagrams, UML object diagrams and UML state machine diagrams (which model different states of objects across different use cases). These UML diagrams are converted into logical theory using the OWL ontology. This formal ontology is analysed by a logical reasoner which is able to automatically check the consistency and satisfiability of the UML diagrams.

A similar study is presented in [58]. UML class diagrams should reflect a business model which includes the most fundamental domain concepts. The authors check the semantic correctness of the UML class diagrams using an ontological approach. The UML class model is converted into a group of logical axioms. Domain knowledge is modelled using the OWL ontology. A reasoner "compares" these two structures, i.e., checks whether the OWL ontology is still consistent if the logical axioms of the converted UML class model are added one-by-one into the ontology.

Wei, Sun, and Wang [59] focus on a similar problem, but they use the OWL ontology for the formal representation of UML class diagrams, UML sequence diagrams and UML state machine diagrams. They verify whether these diagrams are consistent and complete. As a model example, they integrate the conceptual schemas of the University Information System (UnivSys) into the OWL ontology and use SPARQL to receive feedback about whether these models are consistent and complete.

Hafeez et al. [60] are also interested in UML class models verification, and they propose an ontology-based method (an algorithm) for the verification of the finite satisfiability of the UML class model. The main aim of their study is to reduce the time required for the verification of UML class models and to improve the efficiency of this verification. They claim that: "In this method efficiency of the verification process has been achieved by a reduction of search space."

Jetlund [61,62] is interested in the transformations of UML-based geospatial models into OWL and RDF [10] as these models are presented in the semantic web environment. The majority of geospatial models are available through domain-specific web services (spatial data infrastructures) and it is necessary to find strategies for their intelligent machine processing. The author concludes that the semantics of specific UML concepts cannot be directly transformed into OWL, especially abstract classes, code lists, unions, and aggregations. The author offers reasons for this in [61,62].

The above paragraphs explain that formal ontology can bring added value to a UML, especially in view of adding semantics into UML models. We can also find studies where on the other hand, UML can be beneficial for ontologies. Burek, Loebe, and Herre [63] focus on the improvement of the inner structure ("simplification") of complex ontologies with refactoring. They are interested in gene ontology (GO). GO is a biomedical ontology which formally represents knowledge about molecular functions, cellular components, and biological processes [64]. The authors introduce a new UML profile named FueL (Function Modelling Language). FueL is used to refactor the part of the gene ontology, molecular function ontology (MFO) which represents the molecular activities performed by gene products. Thanks to this refactorisation, the MFO is more understandable for human users.

The third point of view is that the UML and the OWL can enhance each other. Perreiras and Staab [65] introduce the Transforming and Weaving Ontologies and UML in Software Engineering (TwoUse) approach, where UML and OWL can be used together for the specification of integrated models in software development. The TwoUse is a framework using strengths of the UML and the OWL in model-driven software development where it is possible to: "describe classes in UML class

diagrams using OWL class descriptions, semantically search for classes, properties and instances in UML class diagrams, design business rules using the UML profile for SWRL, makes sense of UML class diagrams using inference explanations, graphically model OWL ontologies and OWL safe rules using the OWL graphical editor, graphically model OWL ontologies or OWL safe rules using OMG UML profile for OWL and UML profile for SWRL."

The UML is a modelling language providing a visualisation of the structure and behaviour of an intended system. The rules of visualisation are predefined and strict for UML diagrams. It is obvious which modelling elements should be used in particular diagram. These rules do not exist in the visualisation of ontologies. Many and various software tools exist to examine ontological structure. Some are directly integrated into the comprehensive ontological editors. One of the best known is the Protégé (ver. 4, 5) editor, which provides the OntoGraf plugin or the OWLViz plugin. The TopBraid Composer uses an UML-style visualisation. Others are provided as a simpler visualisation solution (often web-based), such as VOWL [66]. There is no standard notation for the OWL. Bārzdiņš et al. [67] introduce an ontological editor called OWLGrEdd. The semantics of the UML graphical notation are extended to provide a compact visualisation of the OWL structures.

The visualisation of ontologies is also the main topic of authors Li and Zhang [68]. We have to agree that querying very complex ontologies is very difficult if you do not have an idea of what the structure (a schema) looks like. The authors apply an association rule mining algorithm to obtain a RDFS class hierarchy. This hierarchy visualised with the UML-based diagram because it is more known also for novice users. Thanks to this approach, a big picture of the dataset can be obtained.

3.5. Management of Programming Code and Documentation

Systems analysis is focused on the specification and conceptualisation requirements of the application domain which mainly uses UML-based modelling (see Section 3.4). The output of this analysis is used for the design of the software architecture. Implementation follows the design of the software architecture where executable programming code is developed. Whether system architecture reflects system analysis and the programming code reflects systems architecture has to be verified. The consistency of an architecture and programming code is a criterion which has to be checked to ensure the overall quality of a software product. Consistency checking is also realised during the development of ontologies. It has to be proved that there are no logical contradictions in the ontological structure at the level of the class hierarchy and at the level of the individuals ("objects" of these classes). Reasoners (e.g., HermiT [69], Pellet [70] or Racer [71]) are specialised programs able to check the consistency of the ontology.

Bennicke and Lewerentz [43] explain how to interconnect ontological reasoning with architecture-code consistency checking. They apply the reasoner to identification errors in software architecture and a programming code. Concepts used in the software architecture are represented as axioms in a formal ontology. The authors consider Java-based software architecture. Java-based programming code is also formally expressed in the formal ontology. The correctness of the programming code is verified with respect to constraints existing in the architecture.

The documentation of software architecture is also an important step in a software project, especially for new workers who have to become familiar with the software as fast as possible. The problem is that different users (e.g., software designers, software architects, testers, programmers) often use different terminology during software development. It is generally difficult to provide documentation that is comprehensive, unambiguous, and fully comprehensible to all users. Graaf et al. [72] apply ontological engineering for the documenting of software architecture. They develop a formal ontology assisting in the documentation of software architecture. Their main motivation is based on the fact that ontologies are not frequently used for this purpose. They conclude that there is "an empirical evidence that ontology-based software architecture documentation is more effective and efficient for architecture knowledge retrieval than file-based system architecture documentation."

4. Contribution of Ontologies in Information Systems Development and Future Directions

Ontologies are good examples of how philosophy can overlap with computer science; how philosophical (metaphysical) thinking about our reality can be beneficial for software systems development, including IS. Ancient Greek philosopher Aristotle introduced the key term "category" which helps in classifying the "things" of our reality according to similarities and differences [73]. He used a tree structure for the visualisation of these categories and the relationships between them. This is found in the work of the Greek philosopher Porphyry of Tyre, who is the author of the oldest adaptation of a hierarchical structure, named the Tree of Porphyry. It distinguishes material and immaterial substances according to their properties [74]. The Tree of Porphyry is the first classical ontology. This short historical background and examples demonstrate how we think about and sense our surrounding environment. Tree-based and graph-based structures became a basic approach to the informal and semi-formal expression of the entities we talk about. From a historical point of view, philosophical ontologies were used for conceptualisation. This is also true for computational ontologies. Systematic review demonstrates that ontologies provide well-arranged formal structures —a vocabulary of terms which is used by the components of information systems for dealing with the semantics of these terms [54,58] or [63]. The majority of the studies presented in this systematic review use only domain ontologies, except for [31]. The authors developed a formal ontology named HARE, using selected upper-level ontologies Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [33] and Suggested Upper Merged Ontology (SUMO) [34].

If a taxonomy is created, the information system can only distinguish between general and specific terms. This is often not enough for the provision of intelligent feedback to end users. Logical axioms as fundamental modelling concepts of description logic that help in the expression of the semantics of described and defined classes. Information systems can then distinguish between these concepts (classes) and provide meaningful feedback or recommendations [36]. From this point of view, the interoperability between different information systems can be facilitated, because a formal ontology can provide a "unified platform" for communication between these systems [28,29] or [31]. Problems arise when these systems use different terminologies in communication. The algorithms of ontological alignment can detect semantically equivalent and distinct terms [75,76]. The next advantage of formal ontologies is supported by inference machines. A formal ontology expresses information and knowledge explicitly, but specific knowledge can be hidden inside its structure. Hidden knowledge is often the result of a combination of logical statements which are represented in various locations of the ontology. Hidden knowledge can be detected by reasoners, for example in Pellet [70], HermiT [69] or Racer [71]. This functionality can also be beneficial for information systems, such as when inferring family relationships [47] or detecting the inconsistencies in a formal ontology [58].

Difficulties often arise during the conceptualisation of really complex systems where various entities are connected with other various entities. It is often a challenge to "untangle" the great many relationships existing between these entities. This is true especially when building medicine-related information systems. The systematic review identified several applications of formal ontologies in medicine, especially for the diagnoses of specific diseases. An inference mechanism can use an ontology and formal rules for the diagnosis of the system-wise diseases [45].

As was already noted, there are many and various methodologies for the development of information systems. The systems development life cycle (SDLC) is a conceptual and processual model describing the phases that should be applied in IS development. It is believed to be the oldest methodology in IS development [77]. We will use this methodology to specify the phases of IS development in which the ontologies can be beneficial. It was selected due to its generic nature: its phases became a basis for other methodologies, such as the Rational Unified Process (RUP) [16,18], Rapid Application Development (RAD) [17] and Agile software development [19]. The following phases of SDLC are considered [1,77]: analysis, design, programming, testing, deployment, and maintenance. Planning activities are part of all these phases. Figure 3 provides an overview of the SDLC phases to which formal ontologies can contribute.

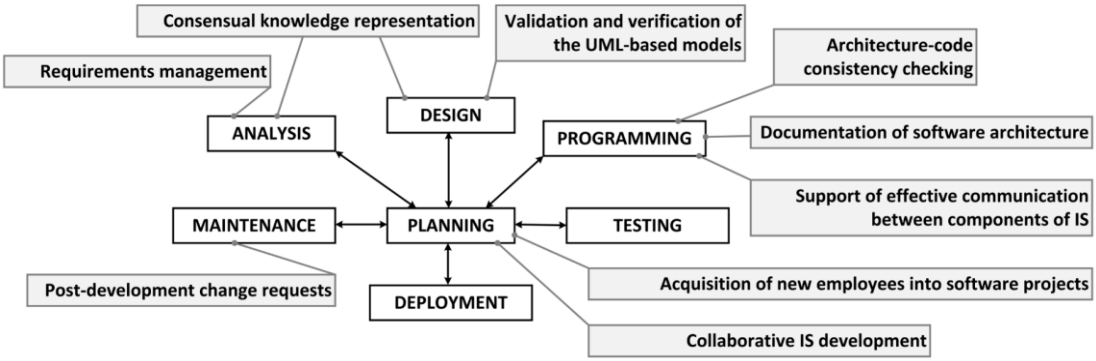

Figure 3. Contribution of formal ontologies to phases of SDLC.

This systematic review indicates that the formal ontology brings substantial added value to the validation of UML-based models [57–60]. UML provides graphical elements for modelling static and dynamic aspects of a system. They do not include a solid formal foundation that could be used for detecting the semantic correctness of the UML-based models. UML CASE tools provide poor support for querying and reasoning. Formal ontologies also model static aspects of a domain, but a reasoner can add something more. It looks like "a brain" which can detect novelties in a knowledge-based structure, i.e., can infer new knowledge of the basis of explicitly described or defined facts. This "brain" can also be used to evaluate the consistency and completeness of a UML-based model. An idea is introduced below regarding ways in which the above ideas can be extended into the area of teaching and learning. *Introduction to object-oriented modelling* is a subject taught at the University of Hradec Králové in the Applied Informatics and Information Management bachelor degree study program. A successful project is one way to acquire credits. The project consists of the analysis and design of an information system. A simplified UML class diagram is modelled during the analysis where the most fundamental classes of application logics with associated relations are detected. This diagram is substantially extended in the design phase where reference attributes, collections, aggregations, compositions, inheritance, utility classes, constructors, getters, setters and other details are added. Students continue by designing a GUI (a graphical user interface) where specific forms are modelled. Dynamic behaviour is modelled using sequence diagrams, and selected designed patterns (Observer, State and Composite) are applied for suitable situation. Around 300 full-time students and 100 part-time students study this subject in each winter semester. It is very time-consuming and exhausting to manually check all the diagrams. The UOMO Validation Service was developed to assist in the evaluation of projects. Firstly, a student's project is exported into an XML file (using XMI (XML Metadata Exchange)) in Enterprise Architect: one of the best known and most complex environments for software design and analysis. This file is loaded into the UOMO Validation Service. Validation is based on a series of conditions which are predefined for each validation task in C#. Validation operates mainly at the syntactic level, not on the semantic level; the. count of modelling elements and their types are checked. The idea is to add a semantic layer to the UML class and the UML sequence diagrams. A validation file is presented in the XML. The OWL formal ontology can also be expressed in XML. Transformational rules should be defined for converting the UML into the OWL. SWRL rules would define conditions which should be fulfilled by the UML diagrams. A rule engine would take the formal ontology and apply the rules for detection if the UML diagram contains contradictions. From an educational point of view, a rule engine would provide guidance about which parts of the UML diagram are not correct and advice about how to fix them. The TwoUse approach could be used for this purpose [65].

5. Conclusions

This systematic review answers the question of how beneficial formal ontologies can be in information systems development. One-hundred eighty-seven records were found through a database search. Twenty-seven studies were investigated after the removal of duplicate documents and the

exclusion of irrelevant and unhelpful papers. The five most fundamental thematic groups were identified, which indicated the most frequent application of formal ontologies in information systems development: the interoperability of information systems, human resource management, domain knowledge representation, integration of semantics into the UML-based models and the management of programming code and documentation. It was found that not only formal ontologies can be beneficial in software (IS) development, especially for semantics-based UML diagrams validation, but that the UML itself can provide added value for formal ontologies. Various applications for the formal ontologies (mainly expressed in the OWL language) were noted in the context of the software development and we provide ideas about how to extend them, which is closely related to teaching and learning. The UOMO Validation Service is actually based on syntactical validation of the UML class and sequence diagrams. It would be beneficial to include the semantics-based validation of these models with the SWRL rules and rule engine, which would detect incorrect statements in the UML diagrams and provide feedback to students about where the problem is and how to solve it.

Supplementary Materials: The following are available online at http://www.mdpi.com/2078-2489/11/2/66/s1, systematic review data and studies processed in single PRISMA phases are available in SystematicReview-Filtration-Overview.xlsx.

Author Contributions: Conceptualisation, M.H. and V.B.; data curation, M.H.; funding acquisition, M.H.; investigation, M.H.; methodology, V.B.; project administration, V.B.; resources, M.H.; supervision, V.B.; writing—original draft, M.H. and V.B. All authors have read and agreed to the published version of the manuscript.

Funding: This systematic review was funded by the Czech Science Foundation GAČR 18-01246s.

Acknowledgments: We would like to thank you our colleague Pavel Čech for providing details about how the UOMO Validation Service works and how it could be improved.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Laudon, K.C.; Laudon, J.P. *Management Information Systems: Managing the Digital Firm*, 15th ed.; Pearson: London, UK, 2017.
2. Merrill, M.D. Knowledge objects and mental models. In Proceedings of the Proceedings International Workshop on Advanced Learning Technologies. IWALT 2000. Advanced Learning Technology: Design and Development Issues, Palmerston North, New Zealand, 4–6 December 2000; pp. 244–246.
3. Cohen, H.; Lefebvre, C. *Handbook of Categorization in Cognitive Science*, 2nd ed.; Elsevier: Amsterdam, The Netherlands, 2017.
4. Lynam, T.; Mathevet, R.; Etienne, M.; Stone-Jovicich, S.; Leitch, A.; Jones, N.; Ross, H.; Toit, D.; Pollard, S.; Biggs, H.; et al. Waypoints on a Journey of Discovery: Mental Models in Human-Environment Interactions. *Ecol. Soc.* **2012**, *13*. [CrossRef]
5. Uschold, M.; Gruninger, M. Ontologies: Principles, methods and applications. *Knowl. Eng. Rev.* **2009**, *11*, 93–136. [CrossRef]
6. McCarthy, J. Circumscription—A form of non-monotonic reasoning. In *Readings in Nonmonotonic Reasoning*; Matthew, L.G., Ed.; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 1987; pp. 145–152.
7. Gruber, T.R. A translation approach to portable ontology specifications. *Knowl. Acquis.* **1993**, *5*, 199–220. [CrossRef]
8. Borst, W.N. Construction of Engineering Ontologies for Knowledge Sharing and Reuse. Ph.D. Thesis, University of Twente, Enschede, The Netherlands, 1997.
9. Berners-Lee, T.; Hendler, J.; Lassila, O. The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities. Available online: http://csis.pace.edu/~{}marchese/CS835/Lec9/112_SemWeb.pdf (accessed on 26 January 2020).
10. Yu, L. The Building Block for the Semantic Web: RDF. In *A Developer's Guide to the Semantic Web*; Yu, L., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 19–86. [CrossRef]

11. Polleres, A.; Hogan, A.; Delbru, R.; Umbrich, J. RDFS and OWL Reasoning for Linked Data. In Proceedings of the Reasoning Web. Semantic Technologies for Intelligent Data Access: 9th International Summer School 2013, Mannheim, Germany, 30 July–2 August 2013; Rudolph, S., Gottlob, G., Horrocks, I., van Harmelen, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 91–149. [CrossRef]
12. Christophides, V. Resource Description Framework (RDF) Schema (RDFS). In *Encyclopedia of Database Systems*; Liu, L., Özsu, M.T., Eds.; Springer: New York, NY, USA, 2018; pp. 3225–3228. [CrossRef]
13. Sengupta, K.; Hitzler, P. Web Ontology Language (OWL). In *Encyclopedia of Social Network Analysis and Mining*; Alhajj, R., Rokne, J., Eds.; Springer: New York, NY, USA, 2014; pp. 2374–2378. [CrossRef]
14. Brockmans, S.; Colomb, R.M.; Haase, P.; Kendall, E.F.; Wallace, E.K.; Welty, C.; Xie, G.T. *A Model Driven Approach for Building OWL DL and OWL Full Ontologies*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 187–200.
15. Isaias, P.; Issa, T. Information System Development Life Cycle Models. In *High Level Models and Methodologies for Information Systems*; Springer: New York, NY, USA, 2015; pp. 21–40. [CrossRef]
16. Hunt, J. An Introduction to the UML and the Unified Process. In *The Unified Process for Practitioners: Object-Oriented Design, UML and Java*; Springer: London, UK, 2000; pp. 19–33. [CrossRef]
17. Kumar, B.; Prashanth, Y. Improving the Rapid Application Development process model. In Proceedings of the 2014 Conference on IT in Business, Industry and Government (CSIBIG), Indore, India, 8–9 March 2014; pp. 1–3. [CrossRef]
18. Kruchten, P. *The Rational Unified Process: An Introduction*, 3rd ed.; Addison-Wesley Professional: Boston, MA, USA, 2003.
19. Stober, T.; Hansmann, U. Overview of Agile Software Development. In *Agile Software Development: Best Practices for Large Software Development Projects*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 35–59. [CrossRef]
20. Gonzalez-Perez, C. How Ontologies Can Help in Software Engineering. In Proceedings of the International Summer School on Generative and Transformational Techniques in Software Engineering, Braga, Portugal, 23–29 August 2017; pp. 26–44. [CrossRef]
21. Henderson-Sellers, B. Bridging metamodels and ontologies in software engineering. *J. Syst. Softw.* **2011**, *84*, 301–313. [CrossRef]
22. Guan, J.; Levitan, A.S.; Kuhn, J.R. How AIS can progress along with ontology research in IS. *Int. J. Account. Inf. Syst.* **2013**, *14*, 21–38. [CrossRef]
23. Beydoun, G.; Low, G.; García-Sánchez, F.; Valencia-García, R.; Martínez-Béjar, R. Identification of ontologies to support information systems development. *Inf. Syst.* **2014**, *46*, 45–60. [CrossRef]
24. Larivière, V.; Haustein, S.; Mongeon, P. The Oligopoly of Academic Publishers in the Digital Era. *PLOS ONE* **2015**, *10*, e0127502. [CrossRef]
25. Adodo, S.O. Effect of Mind-Mapping as a Self-Regulated Learning Strategy on Students' Achievement in Basic Science and Technology. *Mediterr. J. Soc. Sci.* **2013**, *4*, 163–172. [CrossRef]
26. Buzan, T. *Mind Map Handbook: The Ultimate Thinking Tool*; HarperCollins UK: London, UK, 2006; p. 464.
27. Tee, T.; M.N.A, A.; Mohamed, S. Buzan Mind Mapping: An Efficient Technique for Note-Taking. *Int. J. Psychol. Behav. Sci.* **2014**, *8*, 28–31.
28. Debruyne, C.; Meersman, R. Semantic Interoperation of Information Systems by Evolving Ontologies through Formalized Social Processes. In Proceedings of the East European Conference on Advances in Databases and Information Systems, Varna, Bulgaria, 29 September 2011; pp. 444–459.
29. Song, F.; Zacharewicz, G.; Chen, D. An ontology-driven framework towards building enterprise semantic information layer. *Adv. Eng. Inform.* **2013**, *27*, 38–50. [CrossRef]
30. Bai, L.; Koveos, P.; Liu, M. Applying an ontology-augmenting XBRL model to accounting information system for business integration. *Asia-Pacific J. Account. & Econ.* **2018**, *25*, 75–97. [CrossRef]
31. Apisakmontri, P.; Nantajeewarawat, E.; Ikeda, M.; Buranarach, M. An Ontology-based Framework for Semantic Reconciliation in Humanitarian Aid in Emergency Information Systems. *J. Inf. Process.* **2016**, *24*, 73–82. [CrossRef]
32. DiGiuseppe, N.; Pouchard, L.; Noy, N. SWEET ontology coverage for earth system sciences. *Earth Sci. Inform.* **2014**, *7*, 249–264. [CrossRef]
33. Borgo, S.; Masolo, C. Ontological foundations of DOLCE. In *Theory and applications of ontology: Computer applications*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 279–295. [CrossRef]

34. Hnatkowska, B.; Huzar, Z.; Dubielewicz, I.; Tuzinkiewicz, L. Development of Domain Model Based on SUMO Ontology. In Proceedings of the tenth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX, Brunów, Poland, 29 June–3 July 2015; pp. 163–173.
35. Happel, H.-J.; Maalej, W.; Seedorf, S. Applications of Ontologies in Collaborative Software Development. In *Collaborative Software Engineering*; Mistrík, I., Grundy, J., Hoek, A., Whitehead, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 109–129. [CrossRef]
36. Paredes-Valverde, M.A.; Salas-Zárate, M.d.P.; Colomo-Palacios, R.; Gómez-Berbís, J.M.; Valencia-García, R. An ontology-based approach with which to assign human resources to software projects. *Sci. Comput. Program.* **2018**, *156*, 90–103. [CrossRef]
37. Horrocks, I. *What Are Ontologies Good For?* Springer: Berlin/Heidelberg, Germany, 2013; pp. 175–188. [CrossRef]
38. Rotondo, F. *Geographical Information Systems and Ontologies: Two Instruments for Building Spatial Analysis Systems*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 319–328.
39. Pinet, F.; Roussey, C.; Brun, T.; Vigier, F. The Use of UML as a Tool for the Formalisation of Standards and the Design of Ontologies in Agriculture. In *Advances in Modeling Agricultural Systems*; Springer: Boston, MA, USA, 2009; pp. 131–147.
40. Roussey, C.; Pinet, F.; Kang, M.A.; Corcho, O. An Introduction to Ontologies and Ontology Engineering. In *Ontologies in Urban Development Projects*; Springer: London, UK, 2011; pp. 9–38. [CrossRef]
41. Gupta, A.; Condit, C.; Qian, X. BioDB: An ontology-enhanced information system for heterogeneous biological information. *Data Knowl. Eng.* **2010**, *69*, 1084–1102. [CrossRef]
42. Travillian, R.S.; Diatchka, K.; Judge, T.K.; Wilamowska, K.; Shapiro, L.G. An ontology-based comparative anatomy information system. *Artif. Intell. Med.* **2011**, *51*, 1–15. [CrossRef]
43. Bennicke, M.; Lewerentz, C. Towards Managing Software Architectures with Ontologies. In *Graph Transformations and Model-Driven Engineering: Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday*; Engels, G., Lewerentz, C., Schäfer, W., Schürr, A., Westfechtel, B., Eds.; Springe: Berlin/Heidelberg, Germany, 2010; pp. 274–308. [CrossRef]
44. Brochhausen, M.; Spear, A.D.; Cocos, C.; Weiler, G.; Martín, L.; Anguita, A.; Stenzhorn, H.; Daskalaki, E.; Schera, F.; Schwarz, U.; et al. The ACGT Master Ontology and its applications–Towards an ontology-driven cancer research and management system. *J. Biomed. Inform.* **2011**, *44*, 8–25. [CrossRef]
45. Thirugnanam, M.; Ramaiah, M.; Pattabiraman, V.; Sivakumar, R. Ontology Based Disease Information System. *Procedia Eng.* **2012**, *38*, 3235–3241. [CrossRef]
46. O'Connor, M.; Tu, S.; Nyulas, C.; Das, A.; Musen, M. Querying the Semantic Web with SWRL. In Proceedings of the International Workshop on Rules and Rule Markup Languages for the Semantic Web, Brunow, Poland, 29 June–3 July 2015; pp. 155–159.
47. Santos, J.M.; Sousa Santos, B.; Teixeira, L. Using Ontologies and Semantic Web Technology on a Clinical Pedigree Information System. In Proceedings of the 5th International Conference on Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management, Crete, Greece, 22–27 June 2014; pp. 448–459.
48. Yu, L. SPARQL: Querying the Semantic Web. In *A Developer's Guide to the Semantic Web*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 241–290. [CrossRef]
49. Valle, E.D.; Ceri, S. Querying the Semantic Web: SPARQL. In *Handbook of Semantic Web Technologies*; Domingue, J., Fensel, D., Hendler, J.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 299–363. [CrossRef]
50. Luković, V.; Ćuković, S.; Milošević, D.; Devedžić, G. An ontology-based module of the information system ScolioMedIS for 3D digital diagnosis of adolescent scoliosis. *Comput. Methods Programs Biomed.* **2019**, *178*, 247–263. [CrossRef]
51. Bézivin, J.; Muller, P.-A. UML: The Birth and Rise of a Standard Modeling Notation. In Proceedings of the 1998 International Conference on the Unified Modeling Language, Mulhouse, France, 3–4 June 1998; pp. 1–8.
52. UML Superstructure: Language definition and diagrams. In *Real-Time Object Uniform Design Methodology with UML*; Duc, B.M. (Ed.) Springer: Dordrecht, The Netherlands, 2007; pp. 77–190. [CrossRef]

53. Alexander, I. Misuse cases: Use cases with hostile intent. *IEEE Softw.* **2003**, *20*, 58–66. [CrossRef]
54. Arogundade, O.T.; Akinwale, A.T.; Jin, Z.; Yang, X.G. Towards an Ontological Approach to Information System Security and Safety Requirement Modeling and Reuse. *Inf. Sec. J.: A Global Perspective* **2012**, *21*, 137–149. [CrossRef]
55. Liu, C.-L.; Yang, H.-L. Applying ontology-based blog to detect information system post-development change requests conflicts. *Inf. Syst. Front.* **2012**, *14*, 1019–1032. [CrossRef]
56. Dubielewicz, I.; Hnatkowska, B.; Huzar, Z.; Tuzinkiewicz, L. Domain Modeling in the Context of Ontology. *Found. Comput. Decis. Sci.* **2015**, *40*, 3. [CrossRef]
57. Khan, A.H.; Porres, I. Consistency of UML class, object and statechart diagrams using ontology reasoners. *J. Vis. Lang. Comput.* **2015**, *26*, 42–65. [CrossRef]
58. Sadowska, M.; Huzar, Z. Semantic Validation of UML Class Diagrams with the Use of Domain Ontologies Expressed in OWL 2. In *Software Engineering: Challenges and Solutions*; Springer: Berlin/Heidelberg, Germany, 2017.
59. Wei, B.; Sun, J.; Wang, Y. A Knowledge Engineering Approach to UML Modeling. In Proceedings of the The Thirtieth International Conference on Software Engineering and Knowledge Engineering (SEKE 2018), San Francisco, CA, USA, 1–3 July 2018. [CrossRef]
60. Khan, A.H.; Musavi, S.H.A.; Rehman, A.; Shaikh, A. Ontology-Based Finite Satisfiability of UML Class Model. *IEEE Access* **2018**, *6*, 3040–3050. [CrossRef]
61. Jetlund, K. Improvements in automated derivation of owl ontologies from geospatial uml models. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* **2018**, *XLII-4*, 283–290. [CrossRef]
62. Jetlund, K.; Onstein, E.; Huang, L. Adapted Rules for UML Modelling of Geospatial Information for Model-Driven Implementation as OWL Ontologies. *Int. J. Geo-Inf.* **2019**, *8*, 365. [CrossRef]
63. Burek, P.; Loebe, F.; Herre, H. Towards refactoring the Molecular Function Ontology with a UML profile for function modeling. *J. Biomed. Semant.* **2017**, *8*. [CrossRef] [PubMed]
64. Ashburner, M.; Ball, C.A.; Blake, J.A.; Botstein, D.; Butler, H.; Cherry, J.M.; Davis, A.P.; Dolinski, K.; Dwight, S.S.; Eppig, J.T.; et al. Gene ontology: Tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet* **2000**, *25*, 25–29. [CrossRef]
65. Parreiras, F.S.; Staab, S. Using ontologies with UML class-based modeling: The TwoUse approach. *Data & Knowl. Eng.* **2010**, *69*, 1194–1207. [CrossRef]
66. Lohmann, S.; Negru, S.; Haag, F.; Ertl, T. Visualizing ontologies with VOWL. *Semant. Web* **2016**, *7*, 399–419. [CrossRef]
67. Bārzdiņš, J.; Bārzdiņš, G.; Čerāns, K.; Liepiņš, R.; Sproģis, A. UML Style Graphical Notation and Editor for OWL 2. In Proceedings of the International Conference on Business Informatics Research, Rostock, Germany, 29 September–1 October 2010; pp. 102–114.
68. Li, H.; Zhang, X. Visualizing RDF Data Profile with UML Diagram. In *Semantic Web and Web Science*; Springer: New York, NY, USA; pp. 273–285.
69. Glimm, B.; Horrocks, I.; Motik, B.; Stoilos, G.; Wang, Z. Hermit: An Owl 2 Reasoner. *J. Autom. Reason.* **2014**, *53*. [CrossRef]
70. Sirin, E.; Parsia, B.; Grau, B.; Kalyanpur, A.; Katz, Y. Pellet: A Practical OWL-DL Reasoner. *SSRN Electron. J.* **2007**. [CrossRef]
71. Haarslev, V.; Hidde, K.; Möller, R.; Wessel, M. The RacerPro knowledge representation and reasoning system. *Semant. Web* **2012**, *3*. [CrossRef]
72. De Graaf, K.A.; Liang, P.; Tang, A.; van Hage, W.R.; van Vliet, H. An exploratory study on ontology engineering for software architecture documentation. *Comput. Ind.* **2014**, *65*, 1053–1064. [CrossRef]
73. Aristotle; Apostle, H.G. *Aristotle's Categories and Propositions (De Interpretatione)*; Peripatetic Press: Brooklyn, NY, USA, 1980.
74. Strange, S.K. *Porphyry: On Aristotle Categories (Ancient Commentators on Aristotle)*; A&C Black: London, UK, 2014.
75. Ouali, I.; Ghozzi, F.; Taktak, R.; Hadj Sassi, M.S. Ontology Alignment using Stable Matching. *Procedia Comput. Sci.* **2019**, *159*, 746–755. [CrossRef]

76. Essayeh, A.; Abed, M. Towards Ontology Matching Based System Through Terminological, Structural and Semantic Level. *Procedia Comput. Sci.* **2015**, *60*, 403–412. [CrossRef]
77. Hedman, J.; Lind, M. Is There Only One Systems Development Life Cycle? In *Information Systems Development*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 105–116. [CrossRef]