Applying an Ontological Theory of Information Assets to Business Process Reengineering

Patrice Seyed[a,1], John Cabral[b], Daniel Yacob[a], and Michael Corrigan[b]
[a] *Council for Logistics Research, Arlington, VA, USA*
[b] *Synergy BIS, Arlington, VA, USA*

Abstract. In this paper we describe an ontological theory centered on *information assets* as a key invariant for Business Process Reengineering. We also provide examples formulations from its application in a real-world reengineering project. The overall theory includes an ontology whose categories partition into *information asset*, *data element*, and *business object*. Information assets play an important role in our theory because once identified, information assets are 1) decomposed into other information assets and data elements, the latter of which are usually linked to the current-state IT environment systems, 2) linked to business objects in order to construct an ontological model of the information that will be contained and consumed in the future-state IT environment, and 3) linked to both existing and future capabilities and business rules. The first item provides traceability to the existing systems as sources of information, and the latter two items provide a connection to the future design. Within this paper we discuss and formalize these relationships. This approach ultimately provides a detailed specification, which serves as a blueprint for the reengineering of an IT environment.

Keywords. ontology, information asset, systems engineering, business process reengineering

Introduction

In this paper we describe an ontological theory centered on *information assets* as a key invariant for Business Process Reengineering (BPR). Our ontology provides the framework to a formal and rigorous approach to BPR that is IT-implementation neutral. By this we mean the approach abstracts above system-level thinking, which is a typical approach to Enterprise Architecture. In many cases, the capabilities realizable in a future-state IT environment will not in any way resemble the current-state, and so staying within the system-level mindset puts limits on what can be done for the client. Our approach provides a way forward that is independent of a philosophy that is ingrained in a systems view, which we feel is a necessity for communicating with a customer about their overall business needs.

Our specific focus on information assets for reengineering projects forces an enterprise architect (modeler) to consider the needs of the business partners (i.e., people or

[1] Corresponding Author.

organizations that are stakeholders) in a manner independent of how IT systems are configured or how information is stored, which allows for decoupling from specific technologies. As a result of this approach, we are adding an information dimension to traditional BPR, and from this dimension we are deriving the requirements for the future-state IT environment. Typically BPR does not focus on the information dimension, only on process flow, where data objects like forms and reports are linked to process activities. We are aware of some existing work that bridges the gap between business process modeling and service specification, for example in the form of mappings [1], but we found no existing approach that leverages an ontology as the bridge itself, and that comprehends the information dimension. Our ontology enables service specification by determining exactly what information is exchanged and the role it plays for the organizations involved in the exchanges.

The overall theory includes an ontology whose categories partition into *information asset*, *data element*, and *business object*. Information assets play an important role in our methodology because once identified, information assets are 1) decomposed into other information assets and data elements, the latter of which are (usually) linked to the current-state IT environment systems, 2) linked to business objects in order to construct an ontological model of the information that will be contained and consumed in the future-state IT environment, and 3) linked to both existing and future capabilities and business rules. The first item provides traceability to the existing systems as sources of information and the latter two items provide a connection to the future design. In the following section we discuss and formalize these relationships.

An *information asset* is "a body of information, defined, and managed as a single unit so it can be understood, shared, protected, and exploited effectively".[2] An information asset is defined at a level of detail that allows its constituent parts to be managed usefully as a single unit. A necessary (but not sufficient) criterion for an information-bearing object to be an information asset is that it is information exchanged within an activity of a business process, often between internal or external organizations within a business environment. An information-bearing object is an information asset only if it has a specific utility for a person or organization for their business needs. Since an information asset is defined on its utility within certain contexts, its utility is not fully realized without defining its structure and/or formatting. Nevertheless, information assets can possibly exist without IT systems, and be the same across different formats (e.g., a report as a pdf or excel file).

A *data element* is a unit for data of a certain kind, often referred to as an attribute. Data elements correspond to what appears in the schema of database tables, serve as the labels for fields within electronic forms, and serve as inputs for the generation of reports. Data elements describe the data in an information asset, and are often bundled with the aggregation of data that compose an information asset. It is important to identify information-bearing objects that are not data elements, for example, codes. Codes are used for controlled vocabularies to constrain the *values* for some data elements (e.g., NY, NM, CA, for U.S. States).

[2] http://www.nationalarchives.gov.uk

A *business object* is a real world subject of a business activity, although does not necessary refer to real world objects themselves. A business object may be based on a class of entities that have a concrete existence (e.g., a person) or an abstract existence (e.g., a calendar), and are typically those things referred to by nouns. Much like information assets, business objects also have component data elements. Unlike information assets, business objects are not exchanged between activities and among business partners, but play a central role in the mission of the organization in question.

We model business object classes as we would domain ontology classes, as they both represent domain individuals; however, there is a subtle distinction between the two kinds of ontology classes. A domain ontology class typically covers generic knowledge reusable in different applications and environments; on the other hand, a business object is a more constrained representation of reality which is constructed through activities performed within organizations and that occur within relevant business processes. Therefore, a business object is defined more precisely and is subsumed under corresponding, more generic domain ontology classes.

Ultimately, information assets represent business products derived from business processes, and we comprise information assets with data modeled within the business objects. In this sense, the concept of information asset is a "container" within the ontology. In what follows we formally define relations between information assets, data elements, and business objects. To illustrate the formal theory, we provide as examples a portion of an ontology developed for an academic institution's reengineering project in the following sections, and specifically discuss how the ontology informs service design in **Section 3**.

1 Relations Between Information Assets, Data Elements, and Business Objects

1.1 Information Assets and Data Elements

We put forth a primitive relation ***has_component*** as a relation between an information asset and a data element that composes it.[3] For example, a transcript is a report on a student's academic performance which has *AP_score* as a data element. We express this (and all such assertions) in Description Logic,[4] that every transcript has an AP Score data element as a component of it:[5]

$$Transcript \sqsubseteq \exists \boldsymbol{has\_component}.\{AP\_Score\}$$

In this case *AP_Score* is an individual (i.e., a nominal) because it represents a single data element, whereas *Transcript* represents a class that is multiply instantiated by transcripts generated for different students.

[3] We provide the domain and ranges restrictions of our relations informally, for simplicity.

[4] What we refer to more intuitively as relations are roles in DL and properties in OWL-DL.

[5] [5] We apply the '{}' notation to indicate an individual which is in a relation with all individuals of a class. The existential quantifier is misleading but correct syntax for this expression.

We put forth a primitive relation ***has_part*** as a relation between an information asset and another information asset that composes it. For example, a transcript is structured such that the grades that correspond to a specific semester can be easily observed. Duly, semester grades for a student are available in a report at the midway and end points of a semester. We consider the grades for a specific semester *for a transcript* to be an information asset that a transcript information asset has as a part of it:

$SemesterGradesForTranscript \sqsubseteq SemesterGrades$
$Transcript \sqsubseteq \exists \textbf{\textit{has\_part}}.SemesterGradesForTranscript$

Take as another example, a program summary, which for a student indicates courses taken and grades in past semesters, and the courses he plans on taking in future semesters. We consider the grades for a specific semester *for a program summary* to be an information asset that a program summary asset has as a part of it:

$SemesterGradesForProgramSummary \sqsubseteq SemesterGrades$
$ProgramSummary \sqsubseteq \exists \textbf{\textit{has\_part}}.SemesterGradesForProgramSummary$

Both classes, *SemesterGradesForTranscript* and *SemesterGradesForProgramSummary* have further decomposition into data elements, some of which that are shared and some of which are not:

$SemesterGradesForTranscript \sqsubseteq$
$\exists \textbf{\textit{has\_component}}.\{GPA\} \sqcap \textbf{\textit{has\_component}}.\{TotalSemesterHours\} \sqcap$
$\exists \textbf{\textit{has\_component}}.\{CumulativeGPA\}$

$SemesterGradesForProgramSummary \sqsubseteq$
$\exists \textbf{\textit{has\_component}}.\{GPA\} \sqcap \exists \textbf{\textit{has\_component}}.\{TotalSemesterHours\} \sqcap$
$\exists \textbf{\textit{has\_component}}.\{SemesterGradePointTotal\}$

Via inference, the data elements of these information assets are also data elements for the more generic information asset class *SemesterGrades*. The significance is apparent in **Section 2**, where formulations involving capabilities are introduced.

As given, our formal theory provides a relation for partonomic modeling between information assets; it also provides a relation for partonomic modeling between data elements. A data element is typically atomic in nature, that is, refers to a single thing. This allows for greater reusability of the data. There are however exceptions. For example the simple aggregation of the data elements *First_Name*, *Middle_Name*, and *Last_Name* really are parts the *Person_Name* data element. There are other examples, like *Address*. Note that *Person_Name* does not represent a class of information assets, simply because it has further decomposition into data elements. We formalize the relationship between data elements and those that are part of it with the ***has_part*** relation, also:

has_part(*Person_Name, First_Name*)
has_part(*Person_Name, Middle_Name*)
has_part(*Person_Name, Last_Name*)

1.2 Information Assets and Business Objects

We put forth a primitive relation ***about*** as a relation between an information asset and a business object that it is primarily "about". For our example, every transcript is about some student:

$Transcript \sqsubseteq \exists \textbf{\textit{about}}.Student$

We provide several relations between business objects, for example, a student is a person, student *takes* a course, and a course *has* a course schedule:

$Student \sqsubseteq Person$
$Student \sqsubseteq \exists \textbf{\textit{takes}}.Course$
$Course \sqsubseteq \exists \textbf{\textit{has}}.Course\_Schedule$

1.3 Data Element and Business Object

We represent a primitive relation ***has_attribute*** as a relation between a business object and a data element that is crucial for the information asset's modeling within a business environment. For example, every person has a person name:

$Person \sqsubseteq \exists \textbf{\textit{has\_attribute}}.\{PersonName\}$

We infer that *Student* has the *PersonName* attribute, which also holds for any other subclass of *Person* (e.g., *Faculty*, *Staff*).

Unique identifiers are important to any IT system, as the corresponding values are used for tracking and uniquely referencing various things, for example students and courses. We formalize the relation between a business object class and a data element that uniquely identifies it as ***identified_by.*** For example, *StudentIdentifierNumber* is a data element used for identifying every student, and also, *CourseNumber* is a data element used for identifying every course:

$Student \sqsubseteq \exists \textbf{\textit{identified\_by}}.\{StudentIdentifierNumber\}$
$Course \sqsubseteq \exists \textbf{\textit{identified\_by}}.\{CourseNumber\}$

We illustrate these relationships in **Figure 1**.[6] It is important to note that for each data element, depending on its significance in the business environment, it may have a

[6] We omit the graphing of existential restrictions and conjunctions, where they hold, for simplicity of illustration.

corresponding business object. For example, a *Student* business object is in the 'takes' relationship with the *Course* business object,[3] where courses listed for a student in their transcript are described as data elements (e.g., *CourseName* or *CourseNumber*).

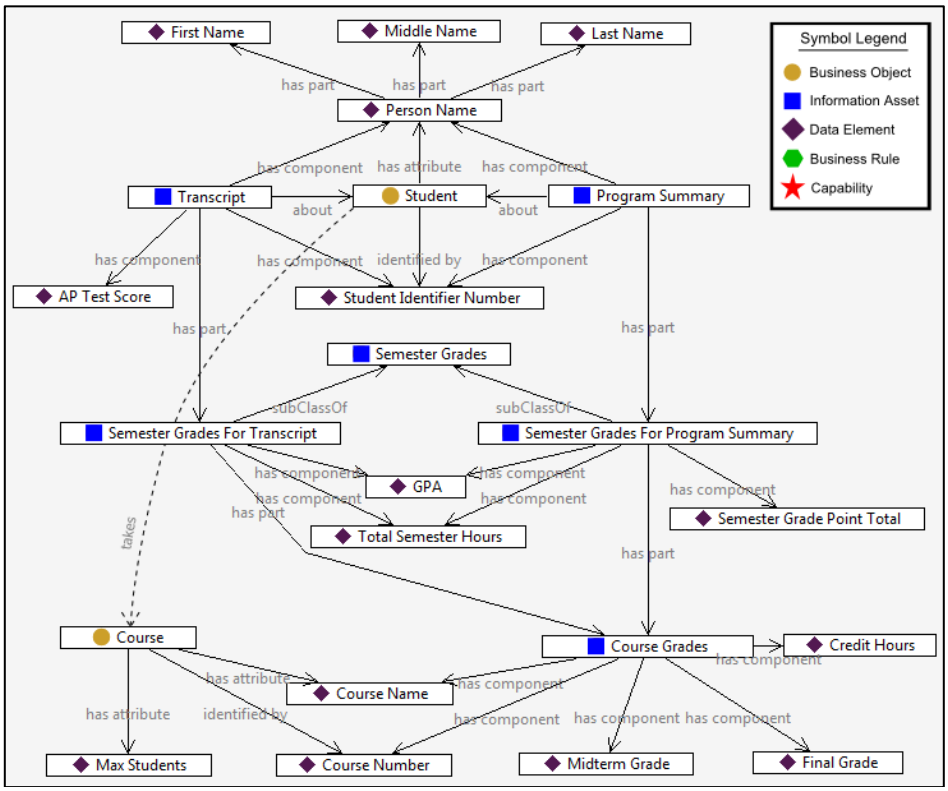

Figure 1. Example Relationships Among Information Assets, Data Elements, and Business Objects

We also note that the design pattern under which we define business object classes requires some additional linkages to mirror how it is normally represented and understood in a domain ontology. For example, the assertion '*Person* ⊑ ∃***has_attribute***.{*HomeAddress*}' provides the relationship between a class of persons and an individual data element, *HomeAddress*. *HomeAddress* here refers a single data element used to describe where a person lives. In formally linking this to a typical formulation, the *HomeAddress* individual holds in a certain relationship with a (distinct) class, *HomeAddressClass*,[7] that has as instances actual locations in the real world where people live. Given this linkage, the assertion '*Person* ⊑ ∃***resides_at***.*HomeAddressClass*' completes the example, which expresses that "every person resides at some location that is his or her home address", a fact we generalize about every person.

[7] We append 'Class' to 'HomeAddress' only to create a label that differentiates it from the corresponding data element label.

2 Linking Information Assets to Capabilities and Business Rules

A *business rule* is simply a statement that defines or constrains some aspect of a business environment. A capability is the ability to achieve a desired effect under specified (performance) standards and conditions through combinations of ways and means to perform a set of activities.[8] Capabilities are often tied to an IT function within an organization. A capability that involves exchanges between organizations, where one organization is the source of the information exchange, is a prime candidate for the development of web services. From a requirements gathering standpoint, capabilities correspond to what functionalities the future-state IT environment enables.

We introduce a primitive relation ***depends_on*** as a relation between capability and an information asset that it realization depends on. By including this relation be have a link between a capability and the data elements on which it relies on. For example, the capability *FlexibleReportGeneration* depends on the information assets *Transcript* and *Personnel Summary*:

depends_on(*FlexibleReportGeneration, Transcript* ⊓ *ProgramSummary*)
We introduce a primitive relation ***produces*** as a relation between a capability and the information assets that it produces. Also, we introduce a primitive relation ***constrained_by*** as a relation between a capability and a business rule that constrains it. For example, a reporting capability on applicant data for a given year for analytic purposes is constrained by the business rule of removing personally identifiable information (PII):

constrained_by(*ReportOnApplicantDataForAnalytics,*
PIIInformationMustBeRemovedForExternalReports)

Therefore, this reporting capability depends on applicant information and produces applicant information without PII:

depends_on(*ReportOnApplicantDataForAnalytics, ApplicantInformation*)
produces (*ReportOnApplicantDataForAnalytics, ApplicantInformationWithoutPII*)

We illustrate these relationships in **Figure 2 and 3**. For our first example capability, *FlexibleReportGeneration*, there is a transitive link between the capability and the data elements that are required for its implementation, through relationships with the information asset classes *Transcript* and *ProgramSummary* (as well as others not listed here). This linkage is crucial, as it serves as structured documentation of the components necessary to implement the given capability in the future-state IT environment.

[8] http://dodcio.defense.gov/dodaf20/dodaf20_conceptual.aspx

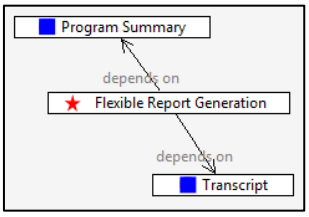

Figure 2. Example Relationships of a Capability with Information Assets

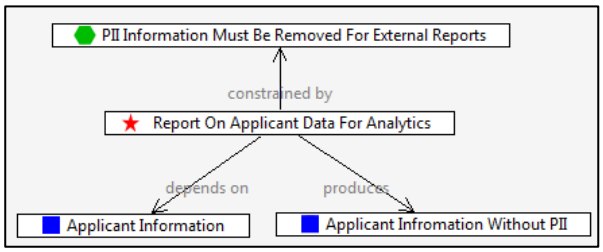

Figure 3. Example Relationships of a Capability with Information Assets and a Business Rule

3 In Practice: Ontology as a Link from Business Activities to Service Design

In what follows we provide a brief sketch of how we practically apply our ontological theory within our current project example. In our case previous work had been performed to construct business process models (i.e., "maps") of the existing environment in Microsoft's Visio. We reviewed the process models with the project's subject matter experts to identify the flow of information between business activities within the models, and depicted it visually. These depictions of the information consumed and produced by business activities are primarily of information assets, but may also be data elements or business objects. Information assets were further decomposed into constituent data elements and linked to the business object to which they pertain. By this approach, we identified shared data elements, for example *GPA* as a data element of both *Transcript* and *Academic Program Summary*, as previously discussed. Both information asset classes are in the *about* relationship with the business object *Performance Assessment* (not depicted visually in this paper).

Some approaches for using an ontology to drive development of an IT solution suggest a direct connection between an OWL ontology and the IT solution, for example where OWL classes are used to generate "code stubs'' such as Java classes [2]. We prescribe a more indirect approach. Our ontology is used to describe services that enable capabilities, and can consist of web service code, COTS/GOTS, traditional system implementation or any combination of these three, leveraging reusable components as much as feasible. The level of service description serves as a bridge between requirements gathering and IT solution implementation.

The ontology of information assets, business objects, and their decomposition contain the detailed information requirements for the IT solution. However, the ontology must be linked to the functions or services that the reengineered IT solution must provide to

the customer organization in order to be practically useful. This includes services that are internal or external to the organization, the latter for which a shared conceptual representation in the form of an ontology is ever more critical.

By our methodology the description of services is accompanied with the ontology portion for the information that will be input into the IT solution, and will be generated by the IT solution. One such service is 'Manage Student Performance'. Beneath each service are one or more operations. In this case we have the operation 'Generate Student Performance Assessment Report'. For this particular service, the data elements of both *Transcript* and *ProgramSummary* are fixed for their type of report, and additional templates can be constructed for new report types using any of the data elements of either. The approach presented here guarantees traceability between business requirements and service specification and assures semantic continuity from the business process models and subject matter expert input to service specification. This reduces the service development effort and later rework by avoiding interpretation problems that stem from ambiguities that typically creep into the final service specification.

Once an ontology based on our theory is used to specify services, there are additional ways in which it can directly facilitate the goals of the enterprise. First, the ontology can be used as the basis for the semantic interpretation of the XML output from a web service. This is accomplished using the SAWSDL[9] standard to map the XSD for the web services to the ontology via an XSLT. Second, we can use the ontology to tag our services with metadata so users can discover them. Here we leverage a metadata population service to automatically generate discovery metadata, using the ontology to identify the relevant subject metadata and other forms of metadata from the XSD and WSDL. The metadata is then stored in a repository to support discovery of information. Finally, we can use the ontology to support query expansion during discovery, enabling users to expand and/or refine their queries based on the problem spaces defined within the ontologies. In each case since we are using the same ontologies for exposing our information, tagging it for discovery, and query expansion during discovery, we anticipate highly accurate query results. We do not yet have a sample size large enough to verify that we are achieving the anticipated accuracy, but we are testing this as the capabilities are deployed operationally.

4 Conclusions and Future Work

Decomposing information assets into their component data elements allows one to clearly determine which data elements are shared among information assets to identify reports with redundant information. This enables future capabilities design, like *FlexibleReportGeneration*. Formally describing information assets that capture the precise information consumed and produced by a business activity allows us to constrain and bind the ontology modeling effort to specific business problems. This allows us to mitigate the risk of entering into unbounded domain modeling.

[9] http://www.w3.org/2002/ws/sawsdl/

In summary, our ontology provides a framework for specifying the linkage of information assets, data elements, and business objects to capabilities and business rules. When applied, the resulting specification serves as a blueprint for the future-state IT environment, and provides the appropriate level of business and information concept modeling needed to specify and implement and data modeling and web services required for that environment.

In future work we will provide an analysis of other information-bearing objects, such as codes, and their relations with information assets, business objects, and data elements. We will also formally define the relationship between capabilities and services and their operations in our ontology. Furthermore, we will present our overall methodology, including how we used various COTS tools together in a workflow for ontological engineering, in a manner that ensures synchronization and avoids rework. Also, given our desire to re-use standards, in the future we will try to frame our ontology as an extension to another architecture framework. One architecture we will evaluate for this work is DoDAF Meta Model 2[10] and its representation of information and data.

References

[1] Stephan Buchwald, Thomas Bauer, and Manfred Reichert. Bridging the Gap between Business Process Models and Service Composition Specifications. In Service Life Cycle Tools and Technologies: Methods, Trends, and Advances, p. 124-153, 2012.

[2] Knublauch, Holger. Ontology-Driven Software Development in the Context of the Semantic Web: An Example Scenario with Protege/OWL. Editors: Frankel, David S. and Kendall, Elisa F. and McGuinness, Deborah L. 1st International Workshop on the Model-Driven Semantic Web, Monterey, California, USA, 2004.

[10] http://dodcio.defense.gov/dodaf20/dodaf20_dm2.aspx