Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2009 Proceedings

Americas Conference on Information Systems (AMCIS)

2009

A Conceptual Model and Typology for Information Systems Controls

Lior Limonad
University of British Columbia, lior.limonad@sauder.ubc.ca

Yair Wand
University of British Columbia, yair.wand@ubc.ca

A Conceptual Model and Typology for Information Systems Controls

Lior Limonad
Sauder School of Business
University of British Columbia
lior.limonad@sauder.ubc.ca

Yair Wand
Sauder School of Business
University of British Columbia
yair.wand@ubc.ca

ABSTRACT

Controls are widely used in business and are often related to information technology (IT) because IT systems are used to implement business controls and because the introduction of IT entails additional control concerns. Thus, control aspects should be part of information systems analysis and design. Furthermore, information systems need to be examined for completeness and correctness of their controls.

However, despite the importance of IT controls, no general, well formalized framework is available to guide the analysis of controls requirements, the design of controls in systems, and the audit of existing systems.

This paper presents a conceptual framework of controls based on an ontological foundation and an extended typology of IT controls. The framework can be used to analyze IT control issues and manage IT control assets. An initial evaluation of the typology using a published control framework and an example indicates its potential usefulness.

Keywords

Information systems controls, conceptual modeling, information systems analysis, ontology.

INTRODUCTION

The purpose of controls is to assure that the organization operates in compliance with external regulations and internal policies. The importance of compliance has increased with the introduction of laws and regulations enacted in response to security threats and business misconduct. Examples are the Patriotic Act (Doyle et al. 2002) and the Sarbanes-Oxley act (Lander 2004).

To assure compliance, organizational activities are subject to auditing inspections (Weber 1998). Auditors focus on whether administrative controls have been well designed and whether they are operating effectively. Considering the complexity of IS and the interdependencies between policy artifacts, it is essential for organizations to substantiate a chain of traceability between controls and policies and regulations (Ueli et al. 2006). Auditing needs now reflect demand for timely and broader information from decision makers and various stakeholders such as potential investors, creditors, customers and suppliers (AICPA 1999; Kogan et al. 1999).

In addition to acting in response to external motivators, organizations install controls to address managerial concerns. Being able to assess the effectiveness of control activities is the key for an organization's ability to assess risks, enforce coverage of concerns, and being able to systematically address them.

Since IT is used to support business activities, control needs impact IT applications. IT applications both need to reflect control considerations, and are used to help implement controls. As well, the introduction of IT brings about additional control concerns. In fact, frameworks such as COSO[1], COBIT[2] and ITIL[3] are strongly related to control aspects of IT. More

[1] http://www.coso.org

[2] http://www.isaca.org

specifically, control mechanisms such as 'input validation', 'exception handling' and 'user's authentication' are already widely embedded in application systems. It follows that control considerations are an important part of IT requirements, and should be included as part of systems analysis and design.

To be able to analyze control needs and to assess existing controls, it is necessary to have well-defined notion of controls, and a "map" of control concerns. No such generally accepted theoretically-based framework exists yet. In this paper we propose a formal conceptualization of controls based on ontological foundations. We use the conceptualization to generate a typology, of controls. We suggest that this typology can be used to both analyze control needs and manage control assets.

In the following, we present the ontological foundation, followed by a conceptual model of controls. The following sections present the typology, demonstrate its evaluation and provide a short example for its usage.

A GENERAL CONCEPTUALIZATION OF CONTROL

Ontological Basis

In this work, we use an ontological formalization to model controls. In particular we use Bunge's ontological view (Bunge 1977, 1979) and its adaptation to IS modeling (Wand and Weber 1989, 1990, 1995). According to the ontological view, a domain can be described as a composition of *things* that possess *properties*. Properties can be intrinsic to things or mutual to several things. Humans perceive properties as *attributes* whose values may change over time. The set of attribute values at a given time is the *state* of the thing. Each attribute function (over time) is termed a *state variable*. *Interaction* among things is defined in terms of their effects on each other's states.

When a thing changes its properties, human perceive this change as a change of state, termed an *event*. Events are *internal* - caused by internal transformations in the thing, or *external* – caused by interactions with other things. Not all states are possible, and not all state changes can occur. The rules governing possible states and state changes are termed *state law*s and *transition law*s, respectively.

We model organizational activities using an ontology-based business process model (Soffer and Wand 2004, 2007). A *domain* is a composite thing – comprising a set of things and their interactions. A domain can be described in terms of its state variables. A *sub-domain* is a part of the domain, modeled by a subset of the state variables of the domain. Formally, a (sub)domain D is modeled in terms of a set of state variables $X^D=<x_1 \ldots x_n>$ where $x_k=f_k(t)$ are attribute functions. The set of possible states is denoted: $S^D=\{<x_1 \ldots x_n>\}$.

Changes in a (sub)domain can occur due to internal transformations, determined by the domain transition laws, and due to interactions with things outside the (sub) domain. The latter are termed *external events*. When a state can be changed by an internal transformation it is termed unstable. If a state can only be changed by interactions it is termed stable. The behavior of a (sub)domain is modeled as follows (Wand and Weber 1988):

> **Definition 1:** The behavior of a domain D is $B^D=<S^D,E^D,L^D>$, where S^D is a set of possible domain states (lawful states), E^D is a set of relevant external events, and L^D is a set of laws determining internal domain transformations.

We make certain premises regarding the domain behavior. First, based on an ontological assumption that every thing changes:

> **Assumption 1:** Every domain state changes with respect to time.

Additional premises will be described in the following sections.

A Formal Conceptual Model of Control

The development of the proposed model of controls is accompanied by a simple example of a fire door and a self-closing spring. The purpose of the spring is to implement the policy that the door has to be closed, which in turn reflects the need to comply with Occupational Safety and Health Administration (OSHA) regulations[4].

We distinguish between the *target domain* (*T*) and the *implementation component domain* (*IC*). The *target domain* is a part of the organization which is required to comply with certain rules (e.g. policies, regulations, and business rules). The

---

[3] http://www.itil.org

[4] http://www.osha.gov/comp-links.html

implementation component is the organizational mechanism (any combination of people and technology) put in place to assure compliance. An example for a target domain in a company is payroll operations. In this case the implementation component may include the software which is used to calculate and pay salaries. In our simplistic example, the target domain is the door and the implementation component is the spring. In our analysis we will be interested in information systems as implementation components.

We denote the behaviors of the target domain and the implementation component by B^T and B^{IC} respectively. We distinguish between *expected behavior* - indicating how the domain is expected to behave, and the *actual behavior*. In the example, the expected behavior is that the door must be shut unless forced to open. This rule should be enforced by the spring, once the door and the spring are installed and attached properly. The actual behavior of the door indicates what really happens once it is in place. For example, it might not be shut completely because the spring is poorly attached. The various behaviors are summarized in Table 1.

Domain / Behavior	T - Target (examples: Fire Door in place, computerized accounting system)	IC - Implementation Component (example: returning spring, the accounting software)
Expected	$B^T(E)=<S^T(E),E^T(E),L^T(E)>$	$B^{IC}(E)=<S^{IC}(E),E^{IC}(E),L^{IC}(E)>$
Actual	$B^T(A)=<S^T(A),E^T(A),L^T(A)>$	$B^{IC}(A)=<S^{IC}(A),E^{IC}(A),L^{IC}(A)>$

Table 1 - Behaviors related to controls

We now formalize the concepts further.

> **Definition 2:** The *expected behavior* of the target domain is $B^T(E)=<S^T(E),E^T(E),L^T(E)>$ where $S^T(E)$ are the acceptable states, $E^T(E)$ is the set of expected external events, and $L^T(E)$ is the set of transformations that can happen.

Both sets $E^T(E)$ and $L^T(E)$ are in the form of $\{<s \rightarrow s'> \mid s,s' \in S^T(E)\}$. In $E^T(E)$ s is stable, and the change is forced by interaction with something outside the domain. In $L^T(E)$ s is unstable and a change happens due to actions in the domain.

Although $S^T(E)$ includes only acceptable states, it is also possible to describe required states in the definition of $B^T(E)$. A required state s_r is either a direct result of an expected external event (i.e. $\exists s:<s \rightarrow s_r> \in E^T(E)$) or an indirect outcome of an external event reached through a sequence of internal transitions included in $L^T(E)$.

In the context of the fire door example, $S^T(E)$ can include all positions and angles the door can be at as depicted in the door blueprint. $E^T(E)$ includes all direct reactions of the fire door to external stimuli as described in the blue print (e.g. the door being pushed from the inside). $L^T(E)$ includes all indirect reactions that will eventually change the state of the door to being closed as required by the OSHA policy.

> **Definition 3:** The *actual behavior* of the target domain is $B^T(A)=<S^T(A),E^T(A),L^T(A)>$, such that: $S^T(A)$ is the set of all actual states that can be observed, $E^T(A)$ is the set of all actual external events that can be observed, and $L^T(A)$ is the set of all internal transformations that actually occur.

We assume that everything that can happen over time will be eventually observed.

The formalization of actual behavior is based on the assumption of *deterministic behavior* as follows.

> **Assumption 2:** The actual behavior $B^T(A)$ is deterministic. That is, given a specific initial state, if the same external stimuli occur, the system will always reach the same state.

The expected behavior is captured by the allowed states $S^T(E)$ and transformation laws $L^T(E)$, and reflects both laws of nature and organizational requirements reflecting regulations, business policies, and norms. Except for laws of nature, all other laws cannot be guaranteed to hold in the actual behavior of the target domain. To enforce these laws, additional mechanisms are often used. We term such a mechanism an *implementation* component. In the fire door example, the implementation mechanism is the returning spring intended to shut the door after it has been opened.

The implementation component comprises a separate domain (IC) then the target domain. However, the two domains are connected. This means that they can interact and (ontologically) this means they have some mutual state variables. For example, the spring is attached to the door. We define the implementation component:

> **Definition 4:** An implementation component is an artifact that is included in target domain so that the combined behavior will satisfy: $S^T(A)=S^T(E)$ and $L^T(E)\subseteq L^T(A)$.

Note, the definition allows for additional transformation laws (specifically, those related to the behavior of IC).

In our example, the spring is one possible realization for an implementation component. Other realizations, not necessarily technological, are also possible. For example, a policy that forces employees to assure the door is shut.

Similar to the target domain, the implementation component has two assigned behaviors - expected and actual:

> **Definition 5:** The *expected behavior* of the implementation component is $B^{IC}(E)=<S^{IC}(E),E^{IC}(E),L^{IC}(E)>$.
>
> **Definition 6:** The *actual behavior* of the implementation component is $B^{IC}(A)=<S^{IC}(A),E^{IC}(A),L^{IC}(A)>$.
>
> The interpretation of each element in these definitions is the same as for the target domain.

The IC is intended to assure the correct behavior of the target domain T. Hence, its expected behavior should include a part that reflects this behavior. We term this the *derived behavior* (denoted $B^{IC}(E)_{/T}$). For example, whenever the door is open the spring is expected to push it back. As well, the IC has an inherent behavior ($B^{IC}(E)$-$B^{IC}(E)_{/T}$). An example would be the internal working of the spring. Accordingly, we distinguish between:

- $E^{IC}(E)_{/T}$ and $L^{IC}(E)_{/T}$ – external events and internal transitions of IC that represent the expected behavior of T.
- $E^{IC}(E)$-$E^{IC}(E)_{/T}$ and $L^{IC}(E)$-$L^{IC}(E)_{/T}$ – all external events and internal transitions of IC not derived from the expected behavior of T (i.e. inherent to IC).

The insertion of the implementation component adds new external stimuli to the behavior of the target domain. These add events to $E^T(A)$ which are the result of the interaction between the target and implementation domains, hence:

> **Assumption 3:** the insertion of an implementation component does not add any new event to the target domain T that has no corresponding event in the implementation component domain IC.

Being an artifact, the actual behavior of the implementation component might be different than expected for several reasons:

- Planning errors: not all states and events (internal or external) of the target domain T were considered (i.e. incomplete coverage or incorrect design).
- Implementation errors: the actual behavior of the implementation component is inconsistent with its expected behavior as a result of some flaw in its construction.
- Operation errors: the usage and operation of the new implementation component is not as expected (specifically, it is subject to external events not planned for).

Due to such errors, the implementation component might fail to assure the target system behaves as expected. The purpose of a *control system* is to overcome this problem.

THE CONTROL SYSTEM

We describe the essence of control mechanisms in organizational settings according to the dynamic view and the concepts that were presented in the previous section. We adapt the basic definition of the control from (Wand and Weber 1989; Weber 1998). This definition relies on the same ontological foundations and therefore fits well with our dynamic view of the organization. According to this view, a control system is defined as follows.

> **Definition 7:** A control System is any system that either prevents, detects, or corrects unexpected transitions (i.e. events) or unexpected states of another system.

In organizational settings, the target of the control system is the implementation component domain IC (e.g. the spring). Similar to other components in the organizational environment, the control system resides in its own domain – the *control domain* (C). Like all other domains, the domain C is associated with two behaviors: expected and actual. The expected behavior of the control system has a dual responsibility:

- Functional – assuring that the implementation component (i.e. the spring) achieves its objective as defined in definition 4, by checking that the actual behavior of the target domain $B^T(A)$ meets all acceptable states and expected transitions in $B^T(E)$.

- Non-functional – assuring that the inherent operation of the implementation component is as expected, namely, checking that $B^{IC}(A)$-$B^{IC}(A)_{/T}$ meets all its inherent expected behavior $B^{IC}(E)$-$B^{IC}(E)_{/T}$ (i.e. all aspects not related directly to the target domain behavior).

According to the above, the functional responsibility is achieved by monitoring the target domain (e.g. the fire-door) behavior and not of the implementation component (e.g. the spring) behavior. The behavior of the implementation component can, in principle, serve as a proxy for that of the target domain (e.g. the position of the spring's arm represents whether the door is closed or not). However, its actual behavior cannot be guaranteed to be flawless and hence cannot be monitored instead of that of the target system. Monitoring the implementation component only will contradict the original motivation for control – identifying errors in its operation. On the other hand, monitoring the target domain when the implementation component is in place will also test if the latter behaves correctly (i.e. $B^{IC}(E)_{/T}$). Thus, in a general control system, the functional focus is on the target domain and not on the implementation component. This assumption can be relaxed only in cases where the implementation component is a reliable representation of the target domain (see next sub-section about IS Controls). This general view also reflects the principle of control dependability in safety critical systems (Storey 1996).

In our example, a sensor that sounds an alarm when the door is open for longer than a certain time can make a valid functional control system. Such a sensor is independent of the self-closing spring and it operates only based on the state of the door (i.e. the target domain).

Next, we add an assumption to eliminate the possibility of failures due to lack of information:

Assumption 4: All domain behaviors in our model are subject to the closed world assumption.

This assumption amounts to the closed world assumption used in formal logic. Based on this assumption it can be presumed that all states and transitions that are not stated in $B^T(E)$ can be considered as failures in the system. Otherwise, functionality of the control system cannot be specified.

A third aspect of control is implied in Definition 7. A control can be preventive, corrective or detective.

Finally, a control mechanism itself is a system which resides in its own domain. As for other domains, this domain has *expected* and *actual* behaviors. Since the control system is an artifact, it is subject to the same types of errors as the implementation component. This brings about the possibility of inconsistency between the two behaviors – expected and actual - of the control. In organizational settings, this is the purpose of audit, which is beyond the scope of this paper.

IS Controls as a Variation of the General Conceptualization

In this section we specialize the general notion of the control to information systems by adopting the *deep structure view* of an IS (Wand and Weber 1995). According to this view, an IS is an artifact intended to represent some perceived aspects of a represented domain. This representation view entails the *state tracking* requirement that the states and behavior of the IS faithfully represent the states and behavior of the represented domain. The represented domain can be either concrete or a conceived system. Therefore, this view encompasses systems that monitor the present state of affairs in the organization (e.g. Transaction Processing IS), systems that aggregate information about past states (i.e. Management Reporting Systems), and an IS that represents future states (i.e. Decision Support Systems).

In the context of our work, we consider the IS as the implementation component. An example would be an organizational Payroll System. The Payroll System is an implementation of the organizational activity of paying employees and is subject to rules (such as government regulations and employment contracts). A non-computerized, alternative implementation could be the employment of a *payor* - a person employed by the organization who has the responsibility to calculate salaries and deposit cash to the accounts of all employees every month. Based on our earlier conceptualization of controls the Payroll System is an implementation component intended to guarantee that the target domain – payroll activities – operate as required and employees are indeed being paid according to the rules (i.e. $B^T(A)$ is as specified in $B^T(E)$ which reflects regulations and employment contracts).

As an IS, the Payroll System should be able to represent, at any given time, data about past payments, the payable activities of each employee, salary calculation details for each employee, and all other details to be considered by the organization in order to process salary payments. A good Payroll System will account for the relevant expected behavior of the organization. That is, the system should assure paying all salaries as required by regulations and contracts. However, as mentioned above, the Payroll System is an artifact. As such, it may be subject to the types of errors described in the previous section (i.e. planning, implementation, and operation). Therefore, a control system should be put in place. The responsibility of such a control system will be twofold:

- Functional responsibility – verify that payroll operates as required (salary calculations are correct and salaries are paid).

- Non-functional responsibility – verify all inherent aspects in the Payroll System that are not derived directly from the payroll requirements (e.g. safety and security of information, and correct system usage).

Based on our proposed model, the first responsibility should be enforced by comparing the actual behavior $B^T(A)$ (e.g. statements issued by the bank that confirm payments) and the expected behavior $B^T(E)$ (e.g. employment contracts). Such verification should be independent of the implemented Payroll System. This requirement for independence was made as there is no way to assure that the state of the Payroll System is consistent with the state of the target domain (which includes, in particular, the employee's account). However, since the notion of representation is an inherent characteristic of the Payroll System (i.e. it is an IS), the validity of such representation is enforced by the control system as part of its non-functional responsibility. Therefore, in the general context of IS as an implementation component, the functional responsibility of the control system can be enforced through the IS itself. That is, assuming that the Payroll System is a faithful representation (which is enforced by the non-functional responsibility of the control system), verification of payments can be validated by comparing information about payments in the Payroll System with employee's contracts.

Based on the above discussion, we specialize the general concept of control described earlier to the case of an *IS Control System* as follows:

> **Assumption 5:** when the implementation component is an information system, the actual behavior of the implementation component $B^{IC}(A)$ is expected to be a faithful representation of the actual behavior of the target domain $B^T(A)$. Therefore, controls can be based on the IS.

Therefore, the modified responsibilities of an IS Control System are:

- Functional – assuring that the IS achieves its objective as specified in definition 4, by checking that the actual behavior of the IS $B^{IC}(A)$ meets all acceptable states and expected transitions in $B^T(E)$.
- Non-functional - assuring that the inherent operation of the IS is as expected:
  - Assuring that the IS acts as a faithful representation of the target domain.
  - Checking that the inherent behavior of the IS (i.e. $B^{IC}(A)$-$B^{IC}(A)_{/T}$) meets all of its inherent expected behavior in addition to being a representation. This aspect includes, in particular, all rules that are not part of the tracking requirement.

CONTROL TYPOLOGY

Based on our conceptual model, we propose a typology of controls that can be used to manage control activities. A typology is defined as a *multidimensional* and *conceptual* classification. Successful classification depends on the identification of its dimensions. Each *type* in the typology is designated by a combination of specific levels (or values) for each dimension. A typology can be evaluated by populating it with empirical cases from a set believed to be exhaustive. Thus, our approach comprises two phases: first, a top-down generation of conceptual classes (e.g. IT Control types) based on the conceptual model; second, validation by a bottom-up matching of empirical cases from a practical domain (e.g. industry practices).

Typology Dimensions

Overall we propose six dimensions for a typology of controls. The dimensions are listed in Table 2. As demonstrated in the next section, the typology can be used to classify common control mechanisms. Such classification can be used to identify coverage, redundancy and similarity of different controls. Thus, we propose it as an analysis tool for control activities.

Dimension Name	Definition	Possible Values	Specification
Control Focus	The monitored domain Can be T or IC. For IS Controls it is IC	Controlled aspects of T (i.e. specific people, devices).	T
		Controlled aspects of the IC (i.e. specific people, devices).	IC
Functional / Non-Functional (Non-functional is always related to IC).	Control system responsibility: The expected behavior of T which should be realized by IC / other aspects of IC	Functional: reflects the purpose of IC.	For general controls: $B^T(A)$ vs. $B^T(E)$ For IS controls: $B^{IC}(A)$ vs. $B^{IC}(E)_{/T}$
		Non-Functional: aspects of IC	$B^{IC}(A)$-$B^{IC}(A)_{/T}$ vs. $B^{IC}(E)$-$B^{IC}(E)_{/T}$

		not directly related to its purpose (e.g. correct usage).	
Control object	Assuring only lawful states or Assuring transitions (internal / external) occur only as expected	Unexpected states.	identify: $s \in S(A) \wedge s \notin S(E)$
		Unexpected events.	Identify Undesired internal transitions: $e \notin \{<s;s'>\|s'=L(s), L \in L(E)\}$ Unexpected external events: $e \notin E(E)$
Hard / Soft Goal	Whether the control system intends to enforce hard goals (necessary objectives) or soft goals (improved performance measures) (Soffer and Wand 2005).	Hard Goal: a subset of all acceptable states.	Identify states from which there is no sequence of transitions to the goal.
		Soft Goal – an order relation on all goal states.	Compares the value of a measure function $m(s)$ over states to possible values.
Risk Functionality	Does the control addresses: *vulnerability* – possibility for an unsatisfactory outcome, *hazard* – potential damage, *risk* – a combination *Risk = Hazard×Vulnerability*. Related to Risk Management Theory (Boehm 1991; Stoneburner et al. 2002)	Prevention – reducing vulnerabilities or mitigating the hazards.	Prevents the target domain from reaching an unexpected state or from traversing an unexpected transition.
		Detection – monitoring and identifying vulnerabilities and hazards.	Detects when the target domain reaches an unexpected state or traverses an unexpected transition.
		Correction – following detection, removing or suppressing all hazard effects.	Alters the system into an acceptable state if an unexpected state or transition is detected.
Control Structure (irrelevant for detective controls).	Related to Systems and Control theory (Heij et al. 2006)	Open loop – no overlap between input (e.g. sensors) and output (e.g. actuators).	no mutual state variables between control input and controlled domain output
		Closed loop – some overlap between input and output.	Some mutual state variables between controlled domain and control related to domain output

Table 2 - Control Typology ("T" – Target Domain, "IC" - Implementation Component)

Typology Evaluation and Use for Analysis

As an initial test of the correctness and completeness of the proposed typology we evaluated it using an existing IT control framework published by *The Institute of Internal Auditors (IIA)* (www.theiia.org). This framework, illustrated in Figure 1, establishes a hierarchy for different levels of concerns that should be enforced by IT controls. In order to represent all concerns as determined by the IIA framework, we have instantiated a possible set of 16 IT controls using a Payroll system example (Table 3). The typology has enabled classifying each of the given IT control types according to all dimensions and was found to cover the overall set of IT control types. Although the concrete classification of each control type reflects our subjective judgment, the typology was useful in generating relevant classification questions (e.g. identifying if the responsibility of a control is functional or non-functional). Furthermore, all levels of all dimensions were instantiated by at least one control type. Therefore, the typology was found consistent with the IIA framework.

After classifying each control by its typological levels, the effectiveness of the overall set can be analyzed for possibly missing controls or for redundant controls. Based only on the example (Table 3), it is apparent that accountability for the correctness of payments is weak (i.e. only one control exists to verify it. As well, there are no controls to account for an efficient operation of the Payroll System, as there is only one 'Soft-Goal' control that addresses the development process (i.e. design methodology).

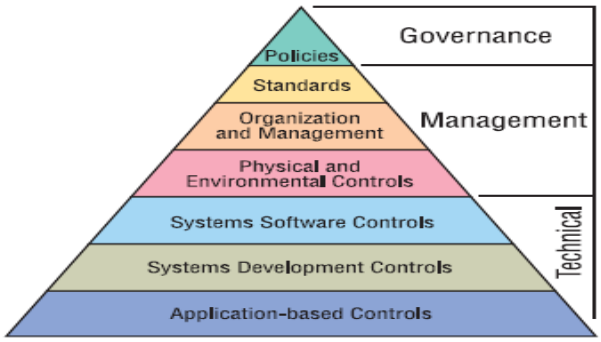

Figure 1 - IIA Control Concerns (adapted from IIA)

Control System (Payroll System)	**IIA Layer**	**Focus – Target domain (T) / Implementation component (IC)**	**Functional (F) / Non-Functional (NF)**	**Object – States (S) / events (E)**	**Hard Goals (HG) / Soft Goals (SG)**	**Risk Functionality – Detect (D) / Correct (C) / Prevent (P)**	**Internal Structure – Open (OL) / closed loop (CL)**
Uninterruptible Power Supply	Physical and Environmental	IC	NF (without it there can still be salary payments)	S (system should be operational)	HG (system should work)	D,P: Detection and Prevention of voltage spikes and power failure	CL (switches power off based on state of system)
A transaction locking mechanism	Financial Control	IC	F (preserves correctness of funds transfer)	E (prevents withdrawal in certain cases)	HG (i.e. ensures correctness of data)	P (i.e. prevents wrong transactions)	CL (i.e. can roll back)
User Authentication and Authorization	Separation of Duties	IC	NF (not required to assure payroll functions)	E (prevents unauthorized changes)	HG (assures system only operates as expected)	P (prevents access)	OL (validation is independent of system output)
Design Methodology *IC: The development team*	System Development	T (the development environment – team, tools, etc.)	NF (not necessary for development)	S (project failure)	SG (a good methodology should lead to better outcome)	P (intended to prevent project failures)	CL (contemporary methods use process based on feedback)
Data backup	Application-based	IC	NF (improve system reliability)	S (the state of data loss)	HG (to assure data are available)	C (recover in case of data loss)	OL (fixed, based on pre-defined backup plan)

Table 3 - An example for the classification of a Payroll System controls (IC is the payroll system unless otherwise indicated)

CONCLUSIONS

We have proposed a model and typology of controls. We used an example (Payroll) to demonstrate that controls that might appear quite different can be framed using one set of dimensions. Our evaluation is only qualitative and is not supported by statistical reliability measures. As well, it is based on one framework (IIA) which is not guaranteed to be comprehensive. This limitation refers to the exhaustiveness of the typology. However, it does not necessarily reflect on the validity of its control classification criteria. We plan to conduct a more comprehensive empirical study in a realistic environment. Such assessment can also increase confidence in the robustness of the proposed typology (i.e. ability to classify new control types that were not part of this framework). In particular, a larger data set might reveal dependencies between dimensions. For example, in the proposed typology the focus of non-functional controls is always on the implementation component. Identification of more complex patterns and interdependencies will further contribute to our ability to analyze control needs and evaluate existing controls.

REFERENCES

1. American Institute of Certified Public Accountants (AICPA) (1999) Continuous Auditing - Research Report.
2. Boehm, W., Barry (1991) Software Risk Management: Principles and Practices, Software, IEEE (8:1), pp. 32-41.
3. Bunge, M. (1977) Treatise on Basic Philosophy, Ontology I: The Furniture of the World, Boston: Reidel, Vol. 3.
4. Bunge, M. (1979) A World of Systems, Treatise on Basic Philosophy, Vol. 4.
5. Doyle, C. (2002) Library of Congress, and Congressional Research Service. The USA Patriot Act a Sketch, Congressional Research Service, Library of Congress.
6. Heij, C., Ran, André C. M., and F. v. Schagen (2006) Introduction to Mathematical Systems Theory, A Birkhäuser book.
7. Kogan, A., E. F. Sudit, and M. A. Vasarhelyi (1999) Continuous Online Auditing: A Program of Research, Journal of Information Systems (13:2), pp. 87-103.
8. Lander, G. P. (2004) What is Sarbanes-Oxley?, McGraw Hill.
9. Soffer, P., and Y. Wand (2004) Goal-Driven Analysis of Process Model Validity, Lecture Notes in Computer Science, Springer, pp. 521-535.
10. Soffer, P., and Y. Wand (2005) On the Notion of Soft-Goals in Business Process Modeling, Business Process Management Journal (11:6), pp. 663-679.
11. Soffer, P., and Y. Wand (2007), Goal-Driven Multi-Process Analysis, Journal of the Association of Information Systems, (8:3), pp. 175-202.
12. Stoneburner, G., A. Goguen, and A. Feringa (2002) Risk Management Guide for Information Technology Systems, NIST Special Publication, pp. 800-830.
13. Storey, N. R. (1996) Safety Critical Computer Systems, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.
14. Ueli, W., I. Majid, M. Matthew, N. Ana, P. Celio, and S. Jason (2006) Rational Business Driven Development for Compliance, IBM.Com/Redbooks.
15. Wand, Y., and R. Weber (1988) An Ontological Analysis of Some Fundamental Information Systems Concepts, Proceedings of the Ninth Annual International Conference on Information Systems, Minneapolis, Minnesota, pp. 213-225.
16. Wand, Y., and R. Weber (1989) A Model of Control and Audit Procedure Change in Evolving Data Processing Systems, The Accounting Review (64:1), pp. 87-107.
17. Wand, Y., and R. Weber (1990) An Ontological Model of an Information System, IEEE Transactions on Software Engineering (16:11), pp. 1282-1292.
18. Wand, Y., and R. Weber (1995) On the Deep Structure of Information Systems, Information Systems Journal (5:3), pp. 203-223.
19. Weber, R. A. (1998) Information Systems Control and Audit, Pearson Education.