Applying categorical logic on a holistic security architecture across distributed systems

Dimitrios Sisiaridis
1st EPAL of Eleftheroupoli
Vocational and Technical Lyceum
Eleftheroupoli, Greece
dimsis@sch.gr

Nick Rossiter
School of Computing, Engineering and Information Studies
Northumbria University
Newcastle Upon Tyne, UK
nick.rossiter1@btinternet.com

***Abstract*—Current security approaches on handling security across distributed information systems are characterized by their locality. We propose the development of a new architecture for organizing in a holistic way the protection measures needed to be taken in a distributed system based on categorical logic.**

holistic security; process calculi; categorical logic; distributed systems

I. INTRODUCTION

Distributed information systems allow users to use common resources and data transparently. For security purposes, it is especially important to know the state of each process as well as the state of communication channels used for exchanging messages between processes.

Most of current approaches are partial solutions as they deal with either the needed controls (addressing security management on a top-down basis) or security measures (in that case, ad-hoc locally applied security mechanisms usually as a software solution, following a bottom-up approach). All high level security aspects, such as policies, services and mechanisms, should be dealt with under the umbrella of a complete layered holistic architecture that is based on higher-order logic, in order to ensure secure transparent distributed computations and to enhance the availability of system's services. Complete, in order to include all the state-of-the-art security services and extensible to include the future ones, holistic in order to meet the new trends and to ensure that everything is in balance, and finally higher-order logic to provide a global solution for handling interoperability and data integrity, issues that are strongly related to security. Thus, processes and channels will be secured and resources will be protected in order to achieve data sharing transparently between system components.

A comprehensive analysis of the literature shows that security for distributed systems is not a local feature but has to be treated globally. It is based on higher-order activities which are related to issues such as data integrity and interoperability among complex heterogeneous systems that share data coherently and transparently.

Holistic security involves both network engineering and application development. Perimeter network defense based primarily on the use of internal and external firewalls cannot guarantee an adequate protection from potential security breaches. The latest attempts on a holistic basis are either platform-dependent [5], cover a specific restricted environment [16] [17] or not fully developed as Butler's holistic security framework [2].

A holistic security architecture, at least, should include baseline assessment, risk analysis, specific policy development, security measure implementation as well as monitoring and reporting action. Assuring optimal security of an information system is not a trivial task, as it requires a wide variety of expertise from technological to organizational. Research has shown that there is not a concrete method to ensure that all possible attacks and loopholes are excluded while a secure system is designed as the final result will be based on the best state of the art available standards. A good practice while designing or evaluating a security framework is to assume for the worst taking into account all the processes which take place in high-level or low-level security services.

II. PROCESS CALCULI

A. Process Management

Good practice in software engineering for process management, in distributed systems, has been based on a variety of mathematical formal interpretations of processes, known as *process calculi*. Process calculi provide a tool for the high-level description of interactions, communication (through message-passing), and synchronization between a collection of independent agents or processes. They have been used for modeling transactions, particularly for representing concurrency. CCS (Calculus of Communicating Systems) [11], CSP (Communicating Sequential Processes) [4] and ACP (the Algebra of Communicating Processes) [1] constitute the three major branches of the process calculi family. The use of channels for communication is one of the features distinguishing the process calculi from other models of concurrency, such as *Petri nets*.

B. λ-Calculus and π-Calculus

Milner has shown [12] that computation and communication in distributed systems and particularly in mobile distributed systems, can be modeled using the notion of the *process*. λ-calculus has been used for handling single-threaded computation. In typed λ-calculus higher-order functions are generally those with types containing more than one arrow. But as Milner observed, processes consist of many elementary parallel, interacting, communicating *threads* which exchange messages with each other.

In π-calculus, all participants in a process such as objects, workflows or any other computational entities or services, can be considered to be just different forms (i.e. types) of an abstract *process* data type. *Names* represent channels that can act both as the actual communication channels (e.g. input and output channels) or as the actual data (the contents encapsulated in messages). Pairs of processes interact with each other by sending and receiving *named messages* in a synchronized way, allowing a dynamic change in the communication topology among processes.

The relationship between π-calculus and λ-calculus as well as a number of applications of π-calculus to object-oriented paradigm has been examined by Sangiorgi and Walker [13]. The relationship of π-calculus with *Business Process Management* (a holistic management approach for good systems design that promotes business effectiveness and efficiency by attempting to improve and optimize processes continuously) along with an implementation for workflow systems was presented by Smith and Fingar [15], where new values are created from processes in an analogous way with an RDBMS that creates new values from data.

III. SECURITY IN DISTRIBUTED INFORMATION SYSTEMS

The issue of security in distributed systems mainly derives by the need of sharing resources via communication channels (e.g. secure channels as SSL) used in the network level by processes in order to exchange messages. These channels should be secured in order to deliver secure distributed computations by enhancing confidentiality and integrity. In the processing level, this requirement mainly is achieved by the use of secure distributed transactions. Actually, security has to do with the applied level of security in each layer of communication between the different layers of a distributed system, expressed in terms of the provided security services as core processes of the system.

A distributed service (e.g. a security service) can be provided by one or more server processes, interacting with each other and with other client processes, in order to maintain consistency on service's resources. Processes (e.g. server processes or peer processes) encapsulate resources (e.g. objects), in the exchanged messages, and allow clients to access them through interfaces. Interfaces are represented as components in the application level. Principals (users or other processes as participants) are authorized to operate on resources. Resources must be protected against unauthorized access.

IV. THE PROPOSED ARCHITECTURE

The current research is towards an *anticipatory* framework as it is unlikely that an organization will have a clear understanding of the probability that an event will occur. It is an attempt to deal with issues as process communication (processes, channels, resources), security mechanisms (in application, network and host level), security services (as complete solutions, not as patches or piecemeal countermeasures), security policies (independent of the implemented security mechanisms and based on security requirements defined in standards). It is also an attempt to define a global security framework for interoperability, with natural closure on the top level, based on categorical logic (i.e. the study of logic with the help of categorical means).

The current work is towards of a non set-theoretic-based approach. It uses applied category theory based on higher-order logic to express complex security activities. It aims to address issues such as semantic and organizational interoperability and data integrity problems which arise when there is a need for communication and computation between local environments of a distributed system, implementing different paradigms, by following industry standards. Furthermore, it attempts to provides the ability to revise, redefine, reorganize and evaluate the current security measures taken in the system under consideration, using state-of-the-art security techniques, measurements of applied security measures efficiency, balancing cost of applied security measures based on risk management as well as taking into account the knowledge of the experts very seriously as in some cases it can reduce time and cost significantly. Security services and activities are explained in terms of core processes of the distributed system. An example of a security service in the case of access control is shown in Fig. 1.

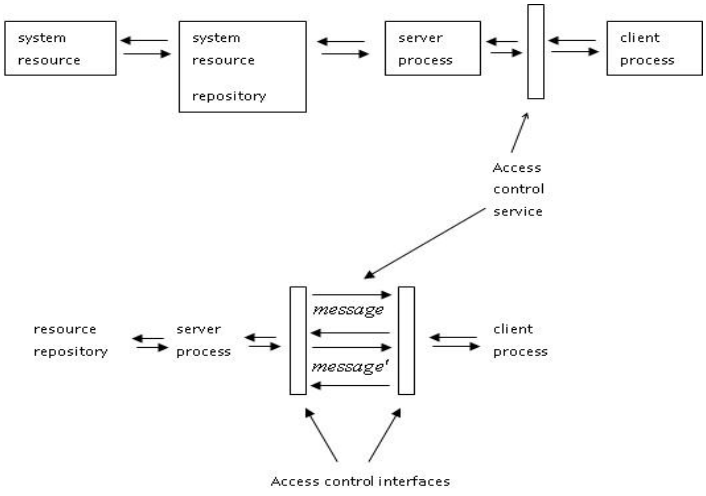

Figure 1. An example of a security service (access control)

Following the ideas of λ-calculus and π-calculus, processes, channels, participants, services, activities and the actual data transmitted in process interaction or those referred to local system state changes, in the architecture, are treated as instances of an abstract type *process*.

A system can evolve in different ways depending on the interaction among processes. The execution of a distributed system is characterized as a series of transitions between global states. As the distributed information system represents *reality* using good software engineering practice, it is "live". That means that when something is happening, we should check for the next *instance* (i.e. view) of the system. In software engineering terms, such an activity is represented by a linearization, which describes the transition from a consistent global state of the system to a new one, also consistent. System evolves (even when it seems that nothing is happening), thus we need to describe this ongoing process. In other words, we need to know what exists between two linearizations. This ongoing process is attempted to be expressed in terms of higher- order categorical structures.

V. CATEGORICAL LOGIC IN THE PROPOSED ARCHITECTURE

A category consists of objects (data) and arrows (functions) between them. Higher order functions, called functors (arrows between categories), describe notions as natural transformations (arrows between functors) and adjunctions, as 1-cells, 2-cells and 3-cells [10].

The four levels in the proposed architecture are named as *Concepts* (**CPT**), *Constructs* (**CST**), *Schema* (**SCH**), and *Data* (**DAT**). **CPT**, **CST**, **SCH** and **DAT** are just labels, not new categories. They represent general categories with internal structure and connectivity. The relationships between the levels are expressed as categorical adjunctions in terms of *functors* which ensure that certain appropriate restrictions are satisfied in the mapping between the source and target categories. Each type level taken with its adjacent type level, acts as a level pair so that there are three level pairs across the four levels. This means that each point at each level is directly related to a point at the other level in the level pair. Between each level the mappings are strictly defined by their starting and terminating points in the respective level types. Thus, each functor in the downward direction (decreasing abstraction) or the upward direction (increasing abstraction) acts as a level pair (Fig. 2).

The proposed architecture [14] is based on applied category theory. We make use of commutative diagrams which are as formal as algebraic equations. Natural transformations are used for process evaluation. Adjointness is used to describe system behavior, something that is important for anticipatory systems [8]. Contravariant logic is fundamental for systems and type theory. Composition of arrows, based on Godement calculus [3], is used to describe process interaction for computation and communication purposes e.g. in distributed transactions for achieving concurrency. Pullbacks provide a tool to illustrate top- and low-level relations and associations between the elements of a system. The exponentiation feature is adequate for binding system activities in a framework. Cartesian closed categories provide a natural setting for λ-calculus. Certain Cartesian closed categories, the *topoi*, have been proposed as a general setting for mathematics, instead of a traditional set theory. Product categories, product functors and bifunctors are used to explain the intensionality and extensionality of a system in the context of comma categories, expressed as pullbacks. Monads and comonads are used to describe step-by-step the internal processing of closed operations in categories.

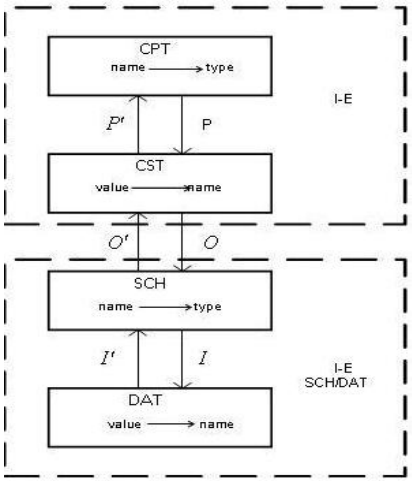

Figure 2. The four levels – intension/extension pairs

Higher-order logic in categorical terms describes notions as *n-categories* and *multicategories* [7] [9]. 2-categories are needed in order to integrate local activities of system components to a global view. 3-cells (i.e. modifications), particularly, can be used to explain system state changes. Bicategories have already been used to represent information flow and access control in security frameworks [6]. Pullback functors can explain the relations in typed systems in terms of Locally Cartesian Closed Categories (LCCC) and categories with families in general. We have shown that LCCC correspond precisely to the industry-strength standard of 3NF for data design, therefore justifying the choice of LCCC as the underlying structures in the architecture. The full interaction and connectivity is given by the functor categories between comma categories involved in the distributed system evolution, focused on those processes that deal with security issues. The internal processing in distributed transactions via secure communication channels is explained using monads/comonads constructions. The step-by-step computation and communication processing is described using the Cube and the Lattice of Cubes categorical constructions that we have developed. Monads can be implemented as *closures*, in programming languages that support closures, based on the endofunctor *T* defined with free variables for closed operations on the components (monads constructions have already been implemented as *wrappers* in functional programming languages).

In Fig. 2, there are two functor categories between each level-pair (top-down / bottom-up direction). A higher-order activity (e.g. a security service) is represented as two adjoint pairs of functors, between each level-pair (for each pair, we have the actual service and its impact - two pairs of adjoint functors for evaluation of ongoing processes in a holistic way). Therefore, if levels **DAT** and **SCH** are needed for defining schemas and explain process interaction in low-level aspects, then levels **CST** and **CPT** are indeed meta- and meta-meta levels of the categories in the architecture (including objects, morphisms, functors, natural transformations etc.). What connects the levels, is the pairs of functor categories between them, not only for defining structures (LCCC) but also for defining and describing all the activities that take place in the system (in the local computational environments/extensions).

The pair of functor categories $\mathbf{DAT}^{\mathbf{CPT}}$ and $\mathbf{CPT}^{\mathbf{DAT}}$ provide ultimate closure in the top level, even if we proceed to higher-dimensional categories as *n*-categories. For example, composition of two 2-natural transformations in a 3-cell (i.e. a modification) takes place naturally inside the boundaries of a system expressed in the proposed four-level architecture. As associativity is not a strict requirement in exponentiation, composition in Cartesian closed categories can be associative up to natural isomorphisms [10]. In weak *n*-categories, as they are treated in the current research with **0**-cells being Cartesian closed comma categories, associativity is not strict (not given by equalities) but it is satisfied up to a natural isomorphism (of the next level).

Thus, using adjointness, for example between categories **SCH** and **DAT** (Fig. 3), arrows f in **SCH** are compared with arrows g in **DAT.** Thus, a **SCH**-object a is compared with the result $GF\alpha$ after applying functors *F* and *G*. This comparison is a natural transformation η involving type changing $a \to Fa \to GFa$, called the unit of adjunction. The comparison is made in the context of the corresponding object Gb which maps b in **DAT** to **SCH**. Similarly, a **DAT**-object b is compared with the result FGb , after applying functors *G* and *F*. This comparison is a natural transformation ε called the counit of the adjunction, involving type changing $b \to Gb \to FGb$. Thus, based on equation solving, there is a functorial way to relate any arrow $g : Fa \to b$ to an arrow $f : a \to Gb$ in such a way that f solves the equation $g = \varepsilon_b \circ Fy$ and that the solution is unique for either some arrow or object y in category **DAT** (as any object y can be regarded as an arrow $1 \xrightarrow{y} \mathbf{DAT}$).

Let us have e.g. a user *Alice* as a participant in a client process. *Alice* is an object or a record, in level **DAT**. It is actually a *user_instance* (a user_object_instance or a user_record_instance, in levels **DAT** and **SCH**) of a type *user* (object_type or entity_type, in level **SCH**). *User* is an instance of the *abstract_object_type* or of a *table* (in level **CST**) as an example of encapsulation or aggregation (in the **CPT** level), in the context of an object-oriented system design or an RDBMS, respectively.

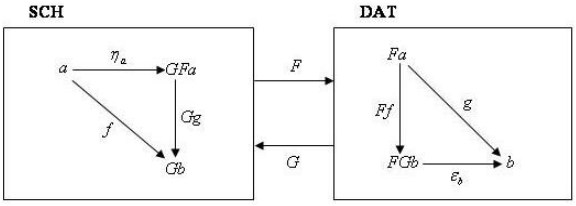

Figure 3. Adjointness between categories **SCH** and **DAT**

The vertical categories (i.e. bifunctors) in Fig. 4 for the three level-pairs are $T(\mathbf{CPT}, \mathbf{CST})$, $T(\mathbf{CST}, \mathbf{SCH})$ and $T(\mathbf{SCH}, \mathbf{DAT})$. Horizontal composition between the second and third level-pairs is given by the bifunctor $K_{\mathbf{CST,SCH,DAT}} : T(\mathbf{SCH}, \mathbf{DAT}) \times T(\mathbf{CST}, \mathbf{SCH}) \to T(\mathbf{CST}, \mathbf{DAT})$ while between the first level-pair and the composite of the other two level-pairs is given by the bifunctor $K_{\mathbf{CPT,CST,DAT}} : T(\mathbf{CST}, \mathbf{DAT}) \times T(\mathbf{CPT}, \mathbf{CST}) \to T(\mathbf{CPT}, \mathbf{DAT})$.

In terms of the corresponding functor categories, horizontal composition of the second and the third level-pairs is given by the bifunctor $\mathbf{DAT}^{\mathbf{SCH}} \times \mathbf{SCH}^{\mathbf{CST}} \to \mathbf{DAT}^{\mathbf{CST}}$, while horizontal composition of the first level-pair and the composite of the other two level-pairs is given by the bifunctor $\mathbf{DAT}^{\mathbf{CST}} \times \mathbf{CST}^{\mathbf{CPT}} \to \mathbf{DAT}^{\mathbf{CPT}}$ The correspondence between the implicated vertical categories and the functor categories is represented in Fig. 4. The functor category $\mathbf{DAT}^{\mathbf{CPT}}$, as the exponential object, provides the ultimate closure in the four-level architecture.

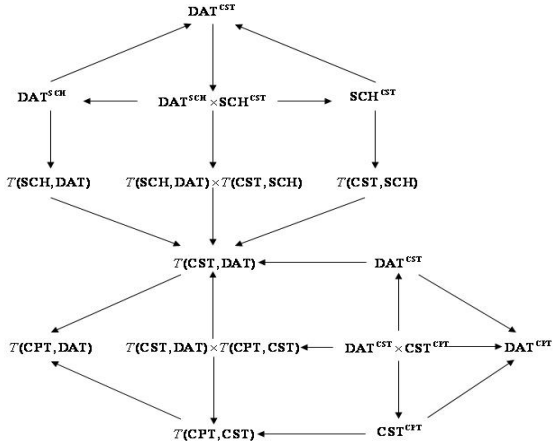

Figure 4. Semantic and organizational interoperability – the functor category $\mathbf{DAT}^{\mathbf{CPT}}$ provides closure in the top level

The four-level architecture in terms of the implicated comma categories is presented in Fig. 5. There, isomorphic categories, for example $P \downarrow \mathbf{CST}$ and $\mathbf{CPT} \downarrow P'$ and equivalent elements in the comma category can be projected onto the same elements

of the product category $\mathbf{CPT}\times\mathbf{CST}$. P, O, I are the free functors who select a target type at a lower level; they preserve colimits. On the other hand, P', O', I' are the underlying (or forgetful) functors, responsible for the prescription of rules; they preserve limits.

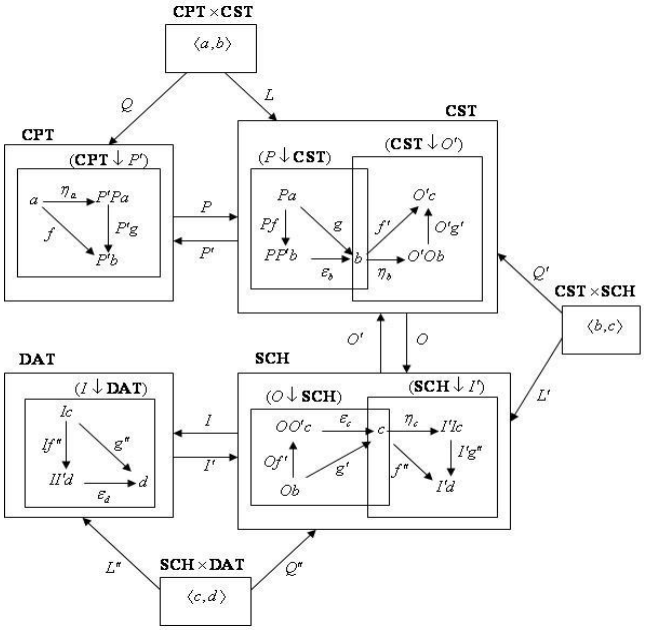

Figure 5. The architecture in terms of comma categories–top/down level

Security requirements and security standards must be satisfied in every level of the architecture Security policies are satisfied in the top level (e.g. security policies defining communication between client, server and peer processes) as well as in the bottom levels (e.g. access control policies governing access on system resources). Security mechanisms are represented as a dynamic synthesis (i.e. composition) of comma categories.

VI. CONCLUSIONS AND FUTURE WORK

The proposed architecture, as it was developed and deployed in the current research in the context of system resources management and system security management in particular, provides the means to organize in a holistic way the protection measures needed to be taken in a distributed system in order to ensure secure transparent distributed computations and to enhance the availability of system's services in complex and interoperable environments. Processes and channels should be secured and resources should be protected in order to achieve data sharing transparently between system components.

A demonstration of the architecture with a real example would be very useful but would be a very substantial endeavour. In its current form, the architecture describes security activities deal with distributed transactions, event-ordering for non-repudiation and auditing control, access control, as well as threat assessment, risk evaluation and control. In practical terms, the current research can be the basis for a future development and implementation of a software graphical tool with the abilities to visualize and evaluate the proposed applied categorical logic in terms of holistic security in distributed information systems. This tool, based on mathematical principles and logic, could be the basis for a future standard way to represent interoperability issues governed by a global security framework in distributed information systems expressed in the proposed four-level architecture.

REFERENCES

[1] J.A. Bergstra and J.W. Klop. "Algebra of communicating processes with abstraction", Theoretical Computer Science, vol. 37, pp. 77-121, 1985.

[2] C. W. Butler. Holistic Security - References on Terrorism, Homeland Security, Threat Assessment and Preparedness, 3rd ed., Butler Research LLC, 2007

[3] R, Godement. Theorie des faisceaux, Hermann, 1958

[4] C. A, Hoare. Communicating Sequential Processes, Prentice Hall, London, 1985

[5] IBM. Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions, RedBooks, 2006 http://www.redbooks.ibm.com/redbooks/pdfs/sg246394.pdf

[6] S, Kasangian, G. M, Kelly and V, Vighi. "A bicategorical approach to information flow and security", in Categorical studies in Italy, Perugia, vol. 2, pp. 99-122,1997

[7] J, Lambek and P.J, Scott. "Introduction to higher order categorical logic", in Cambridge Studies in Advanced Mathematics, Cambridge University Press, vol. 7, 1986

[8] F. W, Lawvere. "Adjointness in Foundations", vol. 23 pp.281-296, 1969 (Reprinted with commentary in Reprints in Theory and Applications of Category Theory, vol. 16, pp. 1-16, http://tac.mta.ca/tac/reprints/articles/16/tr16abs.html, 2006)

[9] T, Leinster. "Higher operads, higher categories", London Mathematical Society Lecture Notes, Cambridge University Press, 2004. (Reprinted as "Operads in higher-dimensional category theory" in Reprints in Theory and Applications of Category Theory, vol. 12, pp. 73-194, http://arxiv.org/abs/math.CT/0305049, http://tac.mta.ca/tac/volumes/12/3/12-03abs.html, 2004)

[10] S, Mac Lane. Categories for the working mathematician, Graduate Texts in Mathematics, Axler, S., Gehring, W. F. & Ribet, A. K. Eds., 2nd ed., Springer-Verlang, New York, 1998.

[11] R. A, Milner. Communicating and concurrency, Prentice Hall, New York, 1989

[12] R. A, Milner. Communicating and Mobile Systems: the Pi-Calculus, Cambridge University Press, 1999

[13] D, Sangiorgi and D, Walker. The π-Calculus – A Theory of Mobile Processes, Cambridge University Press, 2001j

[14] D, Sisiaridis, N, Rossiter. and M, Heather. "A Holistic Security Architecture for Distributed Information Systems - A Categorical Approach", in Proceedings of EMCSR-2008: European Meeting on Cybernetics and Systems Research, Symposium Mathematical Methods in Cybernetics and Systems Theory, University Vienna, 25-28 March, vol. I, pp. 52-57, 2008

[15] H, Smith and P, Fingar. Business Process Management: The Third Wave, Journal of Information Systems, 2004

[16] N, Winn. "Holistic Security in a Fragmented Governance Structure", http://www.esrcsocietytoday.ac.uk/ESRCInfoCentre/ index11.aspx

[17] A, Zuccato. Holistic security management framework applied in electronic commerce., Computers & Security, vol. 26, pp.256-265, 2007