Assessing Distributed Application Vulnerabilities Using MERICS

Cătălin Alexandru TĂNASIE
Bucharest University of Economic Studies, Doctoral School
catalin_tanasie@yahoo.com, catalin.tanasie@hotmail.com

The paper starts by presenting vulnerabilities arising from user interactions as part of distributed application usage. Tokens and user roles are analyzed and interactions formalized based on factors including the location of communicating actors. Authentication issues are detailed, followed by the identification of technical difficulties and administrative incidents, as well as security threats within the distributed ICT applications – DIA, framework. The support for models addressing the evaluation of user and component behavior is done through the MERICS software application. A quantifier regarding inter-component incident impact and interactions is defined and results are shown as obtained using MERICS.
Keywords: *Distributed Applications, Risks, Models, Assessment, Software*

1 Introduction

Distribute applications reunite sets of software modules distinguished by function, location, development technology or security. Users and processes interact with components over varied environments, requiring the usage of encryption tools and the development of flexible protocols for authentication and authorization. These features enhance the frequency of incidents, either by the unavailability of sections of the application, the tampering of data or its incomplete formatting and dissemination. The synchronous or asynchronous communication and processing of tasks introduces additional vulnerabilities, of a qualitative nature and harder to prevent or discover. Delays in the serving of requests, deficiencies in the analysis of input, auditing deficiencies have a negative effect on the efficiency, reliability and performance of the whole system. The study of risk in distributed applications has led to the development of assessment models and their implementation using MERICS, a software application envisioned as a testing and refining mechanism for operational and analytical processes.

2 User interaction vulnerabilities

MERICS, Modele de Estimare a Riscului în Contextul Securizării, Romanian for *Security-Aware Context Risk Assessment Models*, is designed as a **d**istributed **IT** **a**pplication – **DIA**, offering functional context and analytical support for risk factor identification and assessment model evaluation and refining [7]. It implements both operational and analytical software constructs in order to provide the optimum support in threat detection, impact evaluation and performance assessment.
Distributed application users trigger incidents by means of their role and activities as relating to the operational and analytical contexts of the distributed application. Improving the user-DIA interactions through the management of access rights requires the following steps:

- assigning individual security tokens, enabling the creation of unique associations between users and sets of credentials – passwords, digital certificates, biometric information, auditory or visual parameters;
- assigning roles, allowing for the filtering and separation of functional areas in the application as required by security and operational specifications; the users are prevented from accessing irrelevant or sensitive information, as long as it is not required by the specifics of their activities as managed by the distributed system.
- assigning users to groups based on their location or interest area; roles and groups allow for an increased flexibility in mapping business requirements to security practices, as well as decreasing the complexity and manageability of the administrative processes;

- observing and optimizing the user interactions, either through the adding or subtraction of roles; in complex DIAs, where access to supplementary functionalities translates into increasing complexity in the user interface-managed activities, these actions improve training efficiency and increase productivity.

Figure 1 details on the filtering of access through successive architectural layers of distributed applications, as user access is delimited using groups and user roles.

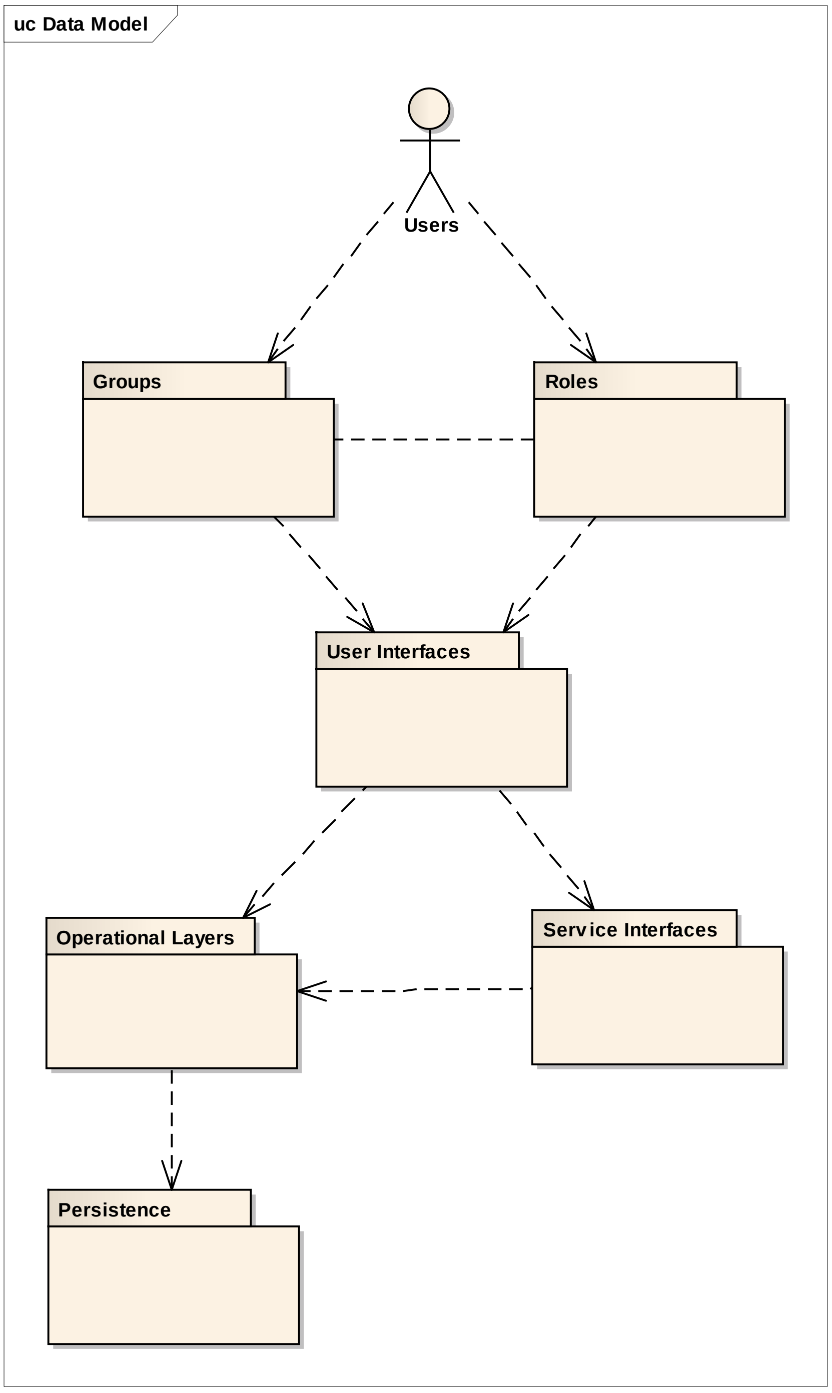

Fig. 1. User – DIA interactions

Timing and access history constitute additional factors that contribute to access management functions. User activity is surveyed and the information gathered is used in the discovery of patterns and limits, which in turn trigger defense mechanisms in case a suspect behavior is observed. In a financial application targeting branch operations involving customers, the detection of activities outside the normal workday interval may constitute a security breach.

3 Deficiencies in administering DIAs

The processes relating to DIA administering and maintenance include a set of activities that influence the security of the system – credential management, process surveillance, tasks relating to the surveillance and repair of automatic components, message flows or services availability. The deployment platform constitutes another factor in determining the risk susceptibility for the distributed application, alongside the following:

- *administrators*, specific users whose activities relate to the ensuring of the functioning of the application; their relation to the context of DIA usage – internal or external – influences the measures taken in improving information security;
- *maintenance and administration jobs* – their specifics are a factor in determining the assignment of users and protection of information; optimizing operational databases often requires access to sensitive data, while interfering with the deployment platform does not imply access rights to information stored or communicated by the distributed application's components; it does, however, imply an increase in availability-related risks;
- *interactions* within the maintenance tasks – the solving of incidents and restoring of functionality to damaged areas of the application requires the collaboration of multiple actors across the context of usage – users, database and functional administrators, testing teams, possibly designers and developers, in case additional safety measures are to be implemented; the management of their activity and communication requires provisioning for both the scheduling of interactions for each party and the hiding of sensitive information.

Ensuring the optimization of administrative tasks requires that management actors ensure that parties involved communicate and act in a manner that allows for parallel operations on the distributed application, without triggering delays or additional issues by concurrent access to the same area. Figure 2 details on envisioned administrative use cases, presenting three of the roles often interacting in solving DIA operational issues. System administrators form the category furthest from the context of DIA usage, often being part of the organization managing the deployment platform or external communication channels. They may not be aware of the specifics of the individual DIA components interacting using their platforms, making them susceptible to causing damage. Database administrators constitute the next level in maintenance-related operations, being trained in the specifics of persistence-related components – or the structuring of information within – and less in the functioning of the system. Their access to sensitive information is an additional risk factor. The last category, functional administrators, is tasked with jobs relating to the operational context of the distributed application – user management, service and information flow process triggering, surveillance.

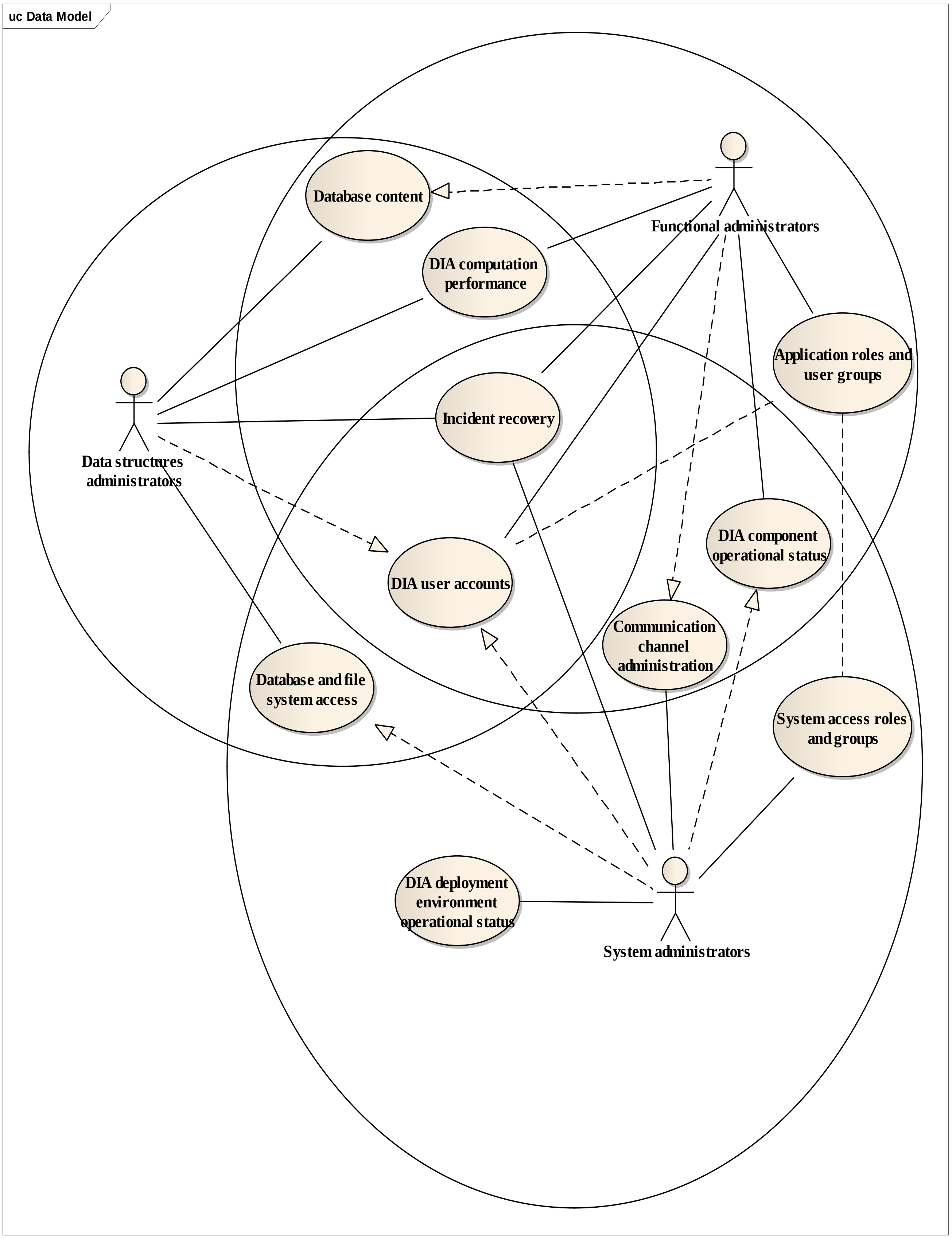
uc Data Model
Database content
DIA computation performance
Functional administrators
Application roles and user groups
Data structures administrators
Incident recovery
DIA component operational status
DIA user accounts
Communication channel administration
Database and file system access
System access roles and groups
DIA deployment environment operational status
System administrators

Fig. 2. DIA administration use cases

MERICS, the model testing and refining application used in the course of the current research effort, requires provisioning for administration-related damage, by building and managing information on issues arising in different components. Table 1 presents some of the incidents relating to correctly identifying credentials, as well as on the behavior and solutions used in removing damage or triggered vulnerabilities. The component type, as relating to a generic architectural approach, is also present.

Table 1. Credential administration issues

Issue	Component	Behavior	Solutions
External factors and processes	All	Users and processes operating outside the DIA context, yet interacting with some of its components, as in sharing communication channels	Cross-platform credential management
Identifying users	Presentation	The party triggering the execution of improperly used or malicious processes is not directly visible	Appending of user credentials to messages
Group-role associations	Presentation, service	Improper assignment of roles or lack of flexibility in creating user groups, leading to insufficient fragmentation of functional access	Reducing user access to sensitive information not relevant to its immediate task, either by the creation of new roles or the deletion of role access for the specific part of the application
Managing contextual parameters	All	Insufficient information or lack of usage of operational context indicators in determining access rights	Including timing and pattern of usage information in security controls

DIA component security requires both passive and active approaches, with backup and recovery tools, as well as constant auditing for uses operations supplemented by generic instruments that act on the detection of a suspect activity, denying the user further rights to the functions of the system, while ensuring that information is stored on their identity, location and actions in order to prevent future incidents.
The deployment environment constitutes and additional risk factor, if users relating to it are allowed access to the informational content of the application. An effective measure is the encryption of data stored using specific repositories. However, the implicit access to communication channels and components leads to incidents relating to the availability of the distributed application. Building an explicit degree of redundancy into the system, as well as maintaining separate location for components that are able to replace existing functionalities or operate in a parallel fashion, helps lower the risk of affecting the global functionality.
Backing up information, as well as storing data relating to the operations performed on an item, preferably through usage of universal, external tools that ensure there is no functional dependency on the status of the analyzed system, is an additional safety measure, as is storing data in separate repositories, with limited or no relevancy assigned to individual records, relying on the access to the whole in order to decipher its meaning. Keeping customer identity in separate repositories is one such step.

4 DIA communication risks

The interaction between users or external processes and the components of distributed applications, as well as their mutual interactions, is managed using a series of instruments that form part, as shown in Figure 3, of increasingly generic areas, starting with the application and ending in the generic context of usage.

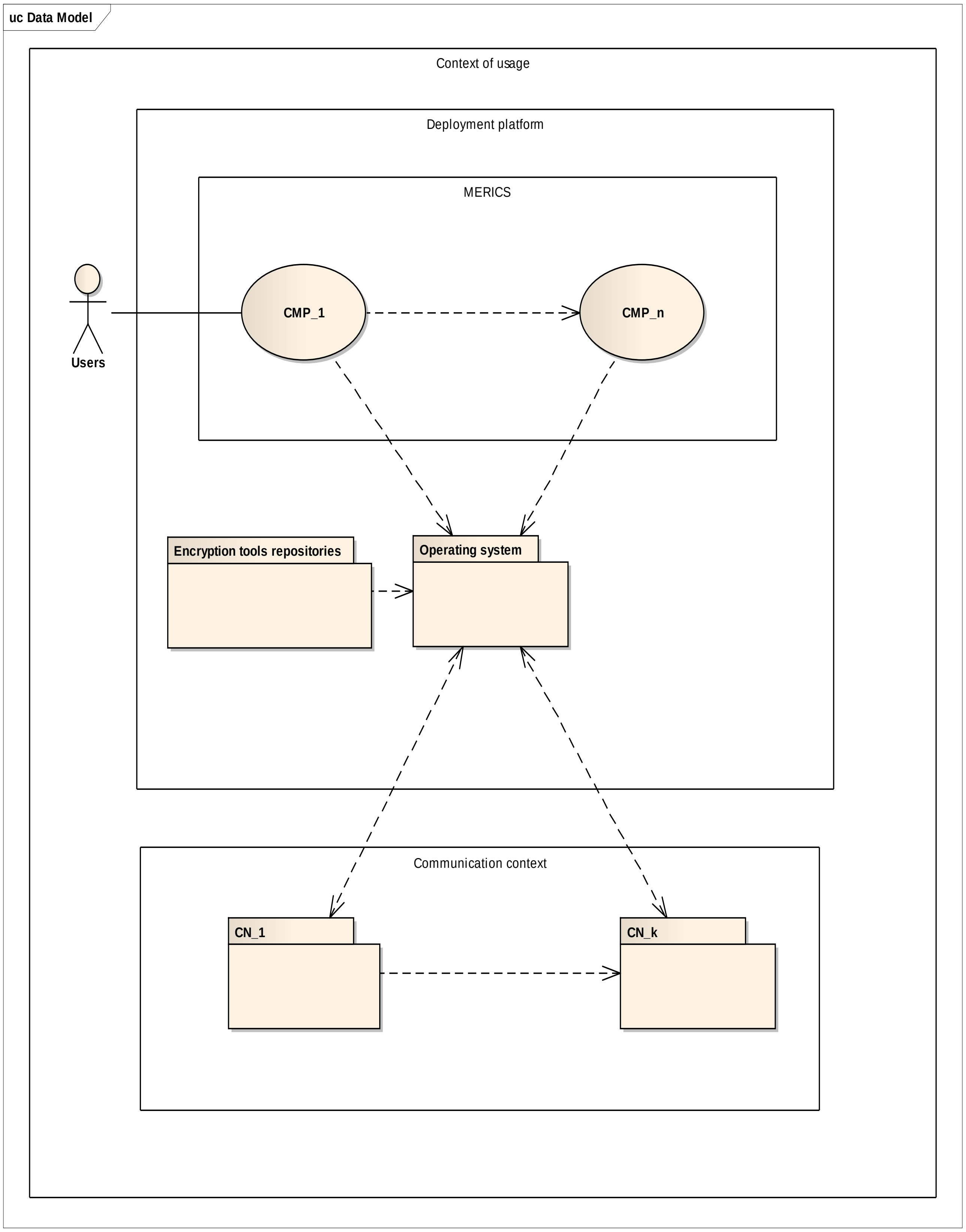

Fig. 3. Communication between MERICS components

The users interact with the application by accessing its interfaces with the help of communication and deployment platforms.

Let set $CMP = \{CMP_1, CMP_2, \dots, CMP_n\}$ define n components of the MERICS distributed application interacting in the solving of a task, and set $CN = \{CN_1, CN_2, \dots, CN_k\}$ of communication nodes involved in distributing information.

In order to properly send and encrypt the data using specialized instruments such as digital certificates, the operating system is used, as well as deployment-platform specific repositories.

Considering an action that requires sending a message between CMP_1 and CMP_n, and that they are deployed in different physical or logical locations, a defining characteristic for distributed systems, the risks associated to this process are envisioned as the sum of the individual entities involved in the processes, the *road* that the message takes from CMP_1 to CMP_n when viewing the entire context as a graph.

Let R_A define the risk associated to the before mentioned action, marked as A. Consequently, it is defined as the sum of its components:

$$R_A = R_{CMP_1} + R_{OS} + R_{CN_1} + R_{CN_k} + R_{OS} + R_{CMP_n},$$

where:

R_{CMP_1}, R_{CMP_n}	–	risks associated to components CMP_1 and CMP_n respectively;
R_{OS}	–	risks associated to the operating system;
R_{CN_1}, R_{CN_k}	–	risks associated to communication nodes 1 and k.

The individual values of risks associated to component communication are seldom quantifiable or directly measurable. Therefore, a system of weights is associated to each of the entities involved, directly related to the number of incidents observed or identified through available information on the behavior of the specific constructs over a predetermined period. Let set $W = \{w_1, w_1, \dots, w_u\}$ define weights associated to u malfunctioning components, as counted over a period of t days. Individual risk is therefore describable as product of measurable losses $-loss$, multiplied by associated weights, or

$$R = w * loss,$$

leading to the redefinition of global risk as a sum of y individual components:

$$R = \sum_{i=1}^{y} w_i * loss_i$$

where:

y	–	number of components associated to measured risk;
w_i	–	weight associated to component i;
$loss_i$	–	loss caused by the occurrence of an incident, specific to component i.

Applying the generic model to the beforementioned risk value R_A, and considering a constant value of external losses over the t period, or one that is not directly traceable to one of the external components, $loss_{ext}$ we obtain the following, as viewed from the MERICS context of usage:

$$R_A = \left(w_{CN_1} + w_{CN_k} + 2 * w_{OS}\right) * loss_{ext} + w_{CMP_1} * loss_{CMP_1} + w_{CMP_k} * loss_{CMP_k}.$$

The model allows for fine-grading the factors involved in risk assessment, as well as eliminating them, to the extreme of considering a constant value of losses, 1, over the measured period, both for external and internal communication processes when viewed from the MERICS scope.

$$R_A = w_{CN_1} + w_{CN_k} + 2 * w_{OS} + w_{CMP_1} * loss_{CMP_1} + w_{CMP_k} * loss_{CMP_k}.$$

This translates to risk being measured as the sum of occurrences for incidents during the usage of the distributed application.

5 Improving distributed applications – test stage analysis

In the lifecycle of software applications, testing relates to the efforts that developers, users, customers, as well as specifically trained personnel undertake in order to provide for a functional, efficient, optimized system. The degree of involvement varies, as well as the moment of their actions.

The first testing-related activities occur during the development of the software application, with the debugging and refining of code as well as the redesign of underperforming or overextended components. Documentation serves as an important support in determining the overall expectancy that the end-user has in relation to the functional areas of a distributed application. A constant occurrence is the lack of detailed description for the procedures and requirements envisioned, as well as the loss of information through successive transformations, as areas of functional expertise translate into technical requirements.

Figure 4 details on the successive interactions between development and test teams in ensuring the proper format and function of a distributed application. Incidents are identified and communicated to the developers, who solve them and resubmit the improved version of the component for revision. Risks arise from the improper communication between parties, leading to improper or incomplete correction of errors, as well as delays in finalizing tasks.

Another issue is the frequency or ability to replicate the observed incident. Due to an improper detailing of the problem, or problems in communicating in a manner understandable to the development teams, less trained in the functional specifications or terminology, specific behavior may be hard to replicate. This deficiency is diminished by the inclusion of explicit logging and error tracing information, allowing for the pinpointing of the failing or misbehaving code fragment or process and its subsequent correction. Test case technology, available with an increasing number of integrated development environments, serves to isolate and replicate the communication endpoints for a specific section of code based on the input and output required by its functions and entities.

The delegation of tasks between users and testing teams, as well as lack of complete understanding of the business requirements with actors involved in the testing of distributed applications, often assemblies spanning multiple areas of operation, leads to the misinterpretation of behavior as erroneous. A long delay caused by a slow, external service interaction may trigger the categorizing of the delay as an error on part of the client endpoint, which incorrectly assigns the incident. The usage of specialized software in managing the interactions between testing and development teams, as well as constantly improving documentation, helps decreasing the frequency of such issues.

The discrepancy in resources allocated to the development and testing as relating to production environments, in terms of processing power, communication channels, availability of users and differences in components stress constitute an additional problem within the testing state of distributed applications. In the course of extending already existing DIAs, the increased strain on resources brought in by the new functionalities requires the redesign of older parts of the system. Risks relate to the improper resource allocation, as the effects of the new modules are hard to assess in preliminary steps and may only appear during the usage stage.

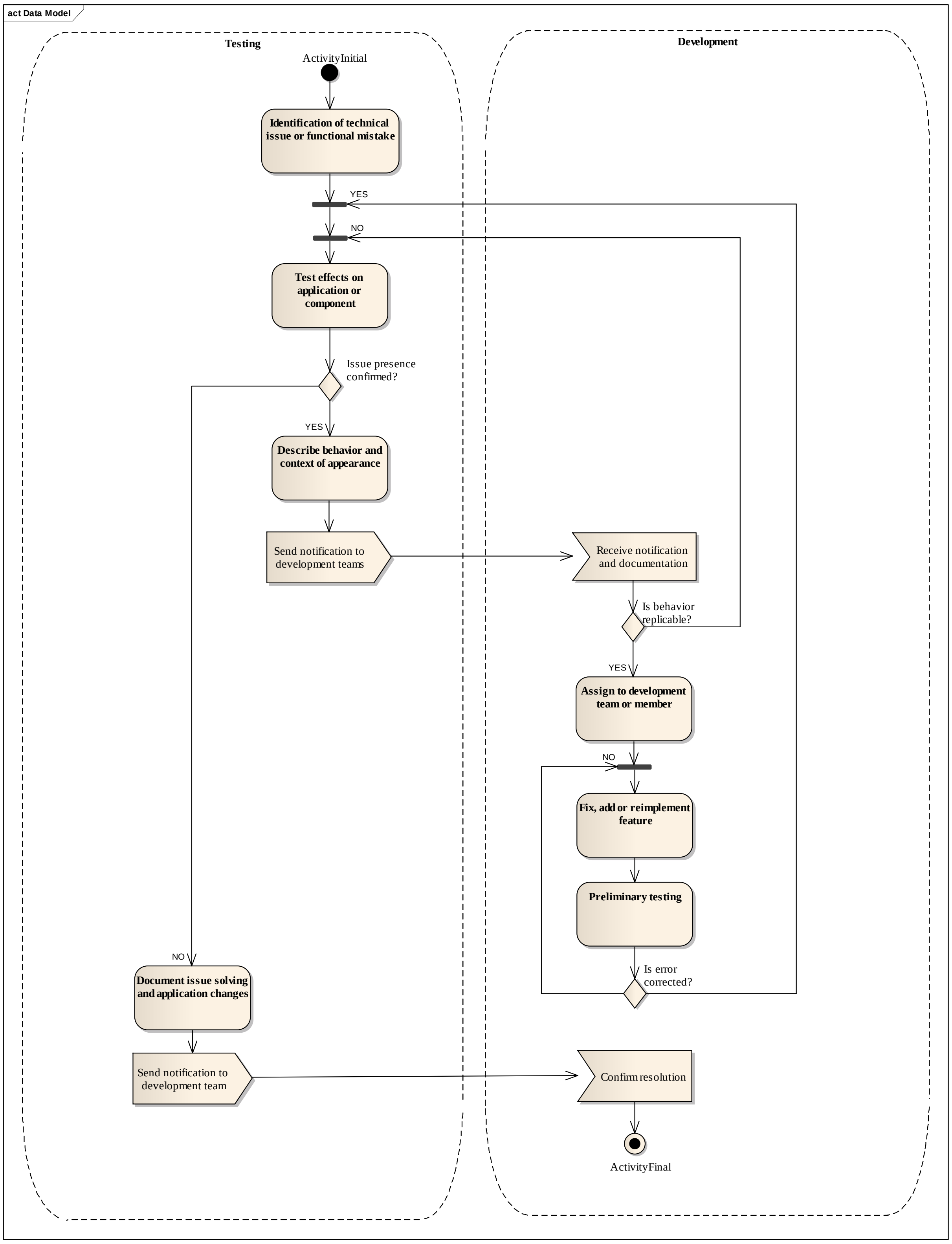
act Data Model
Testing
Development
ActivityInitial
Identification of technical issue or functional mistake
YES
NO
Test effects on application or component
Issue presence confirmed?
YES
Describe behavior and context of appearance
Send notification to development teams
Receive notification and documentation
Is behavior replicable?
YES
Assign to development team or member
NO
Fix, add or reimplement feature
Preliminary testing
Is error corrected?
NO
Document issue solving and application changes
Send notification to development team
Confirm resolution
ActivityFinal

Fig. 4. DIA testing

Due to the impact of communication performance in the overall performance and security of distributed systems, testing the efficiency of cross-component interactions and messaging queues is essential to reducing risk and lowering costs. Impacting the quality and content of the information, communication raises a number of difficulties related to the testing stage:

- *replicating the live usage parameters* – encryption, message frequency, external environment; consequently, errors caused by delays due to the sharing of the communication channels in production environments are not traceable at earlier stages due to the absence of the capacity to replicate the multitude of external factors; MERICS includes analytical components designed to process and store information relating to communication strain, as well as to delegate and delay non-essential tasks;
- *ensuring the simulating of synchronous processes*, in which delays in the processing and storage of information propagate and aggregate through the extent of the components used, impacting unrelated tasks; the reliance on scheduled and asynchronous processes, when applicable, ensures the minimization of risks originating in the described behavior;
- *running tests on complete use case scenarios*, in which multiple interactions are required; while isolating and testing specific components improves on the time needed for the preliminary survey of application functionalities, the inclusion of all relevant processes, users and parameters in testing scenarios ensures the discovery of complex incidents, targeted by the sum of interactions rather than each of them taken individually.

Information quality is tested trough the thorough replication of valid data structures, as well as enforcing the usage of *pre-live environments*, in the sense that they replicate as much as possible the content and extent of data in production repositories. The replicas should, however, be strictly build using publically available or fictive information, as confidentiality may be breached by the replication of customer data.

6 Conclusions

The detection of vulnerabilities in distributed applications requires the structuring and identification of actors and processes that interact during its lifecycle. Diffusing access to users by their aggregation using groups and assignment to functionalities using roles ensures the building of flexible, configurable production environments, with the minimization of information security breaches.

Administrative processes are to be filtered by the properties of their interacting actors as relating the context of the distributed application. Communication is improved by the assessment of individual risk factors and the construction of flexible, weight based indicators used in targeting vulnerable sections for improvement or assessing and provisioning for losses.

The identification of requirements and their correct and complete description influences testing efficiency. Constructing realistic test scenarios and incrementally deploying and improving the application helps the owners or developers of the system minimize risks before the incidents that define them are triggered in the production environment.

References

[1] M. Marian, “A PKI Study within the Educational Environment”, Proceedings of the IWSSIP 18th International Conference on Systems, Signals and Image Processing, Sarajevo, 16-18 June 2011, ISBN 978-9958-9966-1-0

[2] I. Ivan, M. Doinea, D. Palaghiţă, “Aspects Concerning the Optimization of Authentication Process for Distributed Applications”, Theoretical and Applied Economics, Vol. 15, No. 6, 2008, pg. 39-56, ISSN: 1844-0029

[3] D. Šmite, “What Happens When Software Product Development Companies Go Global”, Global Software Engineering (ICGSE), 2010 5th IEEE International Conference, 23-26 August 2010, pg. 319, ISBN 978-1-4244-7619-0

[4] C. Badica, G. Mangioni, V. Carchiolo and D. Burdescu, *“Intelligent Distributed Computing, Systems and Applications”*, Proceedings of the 2nd International Symposium on Intelligent Distributed Computing – IDC 2008, Catania, Italy, 2008, pg. 305, e-ISBN 978-3-540-85257-5

[5] G. Radha Mani and G.S.V.Radha Krishna Rao, "Web services security and e-Business", Idea Group Publishing, Londra, Marea Britanie, 2007, 410 pg, ISBN 1-59904-170-7

[6] A. Belapurkar and A. Chakrabarti, "Distributed Systems Security - Issues, Processes and Solutions", John Wiley & Sons Ltd., Chichester, West Sussex, Marea Britanie, 2009, 307 pg , ISBN: 978-0-470-51988-2

[7] C. A. Tănasie, "Post-release distributed software applications risk management", Proceedings of the Eleventh International Conference on Informatics in Economy IE 2012, ISSN 2284-7472

[8] J.A. Ibrahim, M. Majid, A.H. Hashim and R.M. Tahar, "Risk Quantification in Coal Procurement for Power Generation: The Development of Supply Shortage Impact Matrix", Second International Conference on Computational Intelligence, Modelling and Simulation (CIMSiM), 28-30 September 2010, pp. 401 - 406, ISBN 978-1-4244-8652-6.

Catalin Alexandru TĂNASIE, born at 18.08.1984 in Pitesti, Arges, is a graduate of the I. C. Bratianu National College and of the Faculty of Cybernetics, Statistics and Economic Informatics within the Bucharest University of Economic Studies, the Economic Informatics specialization, 2007 promotion. Starting 2007 he attended the Informatics Security Master in the same institution, and is currently a PhD student at the Doctoral School within the Bucharest University of Economic Studies. He has concerns in the field of distributed applications programming, evolutionary algorithms development, part of the field of artificial intelligence - neural and genetic programming. Currently he works as an application designer in a financial institution. He is involved in creating commercial applications using development platforms belonging to leaders in the field, companies including Microsoft, Oracle and IBM.