Increasing Distributed IT&C Application Security

Ion IVAN, Catalin Alexandru TĂNASIE
The Bucharest University of Economic Studies, Bucharest, Romania
ionivan@ase.ro, catalin.tanasie@hotmail.com

The development of distributed IT & C applications – DIA is presented alongside their main characteristics and the actors involved in activities through-out their lifecycle are identified in the before-mentioned scope. Aspects pertaining security risks, as well as methods of enhancing security, are detailed by DIA architectural features. The analysis includes risk elements, vulnerabilities, means of enhancing the behavior of the system, as well as a hierarchical feature dependency model based on a qualitative assessment of DIA security features, obtained through an inquiry in the common means of protection used by Romanian professionals, as well as their prioritization in the context of limited resources. A graph-based model of feature interactions is built. The last section deals with the ways of improving risk detection methods, as derived from the answers and features presented.
Keywords: *Distributed IT&C Applications, Security, Development, Questionnaire, Assessment*

1 DIA Development

Today's developed informational society involves the implementing of distributed applications, software characterized by components that are either shared by multiple operators – common databases, or part of a larger, enveloping system.

Distributed **I**T **A**pplications – DIAs – define *collections of software modules separated based on function or location and interacting in a structured manner in order to provide an optimized solution to a request*. DIAs are characterized by the existence of components with varying degrees of interdependency, differentiated by role, software technology and geographical location, communicating in a synchronous or asynchronous manner in solving a task. One particularity of such systems is the existence of multiple processing nodes.

DIA components are characterized by:

- *diversity* , in the sense that module development technologies vary widely; the layered architecture of the systems increases the array of tools for implementing the required components, as database management systems, integrated development environments, logging & system auditing are used together in providing better solutions;
- *functional orientation*, each component having a pre-determined, well defined role in the system; communication, computing the input, auditing, authorization, authentication, encryption, maintenance form the array of tasks that characterize distributed application subassemblies;
- *technical autonomy*, the property of a component to function even if other components it interacts with are not accessible; data access and operation security is increased by the usage of duplicate modules, of independently operating computing units and load balancers that besides splitting the workload act towards redirecting jobs in case an error occurs or a component becomes unavailable;
- *logical autonomy*, derived as a notion from the technical one, but encompassing a module's ability to perform its individual tasks at maximum efficiency and without waiting for an external process-based input; when the latter is required, the interaction occurs asynchronously; parallel processing software, acting in the solving of complex problems requiring large amounts of computing power, solve the issues by separating tasks based on the

DOI: 10.12948/issn14531305/17.2.2013.13

degree of interdependency in obtaining the results, analogous to an algorithm for solving a system of equations, in which the order of the initial computations is irrelevant as long as they are all applied;

- *redundancy*, the presence of common functions in more than one component, used to improve system-wide performance or as backup in case an incident occurs; the sensitive nature of financial and security systems, as well as public services, leads to architectures using components performing identical tasks and auditing of the same data set information at multiple points in the system.

Distributed systems are identified with respect to the manner in which resources are accessed, as *parallel* or *concurrent*. An application's components behave in a client or server manner in exchanging information, although the 2 categories are not well delimited. Geographically dispersed users access the application through complex networks in their effort to connect to an often unique pool of resources.

Distributed computing allows for creating roles with regard to the interactions between components, as each module interacts with at least one other in requesting input and providing results. Additionally, roles are not required to be mutually exclusive, although they are built with a predefined scope and share resources accordingly.

The software application lifecycle starts with the decision of meeting an organization, group or individual's demands through the development of task-specialized tools that improve on productivity by the direct or indirect management of economic, social, educational or cultural interactions over uncertain output generation [1], [2]. The procedure of determining the *functional needs* that turn into specifications in the development of software applications includes the following steps:

- *observation*, *awareness* of deficiencies in operations or interactions which occur in the processes that characterize human activities; delays in processing a collaborator's request, unavailability of communication, need of computation power in reducing resource spending, lack of the tools by which to disseminate and exchange ideas or capability to homogenously acquire data, satisfaction of an individual or group's social aspirations, extending an already functioning process or improving its results or consequences, all constitute factors to be addressed as part of today's informational society; the source of the observation is usually the beneficiary of the final product, yet software development companies meet demand by trying to discover and describe patterns and tendencies in target consumer area [3]; this approach leads to the building of extensive information warehouses, market studies and also the appearance of discrepancies between predicted and observed results which translate to risks;
- *executive decisions* which lead to the initiation of procedures regarding the construction or acquisition of tools designed to achieve the desired behavior and containing the required set of functions in mediating operational activities; this role belongs to either the party experiencing the lack of resources or to individuals or organizations that act as service providers; the conclusion of this step involves the assignment of analysis, development and testing resources, either external or internal as relating to the beneficiary of the software and clarification of roles each party involved plays in the completion of the application building tasks;
- *formalization of user requirements*, by producing documents and schematics which help model the interactions between components of the system, its users and elements outside the context of the application; in the area of collaborations between the teams involved in the development of an application, differences in training and understanding of functional aspects at one end, alongside the technical details

DOI: 10.12948/issn14531305/17.2.2013.13

reliance on the other lead to the gradual abstracting of requirements as intermediate actors translate needs into software development constructs; the specifics of the activity set that the software system will manage may not require direct input or control, yet the end result is always related to the improvement of human activities;

- *defining application limitations* as relating to the proportion in which it mediates or successfully replace inferior productivity factors, action related to the mapping of user requirements and identifying the boundaries that the system will set on the extent in which human activities or outdated processes are transferred to the new functionalities [4]; the understanding of limits helps management plan for future extensions, as well as users and business analysts in the identification of operations and training for interacting with the new solution.

Actors involved in DIA functional analysis are tasked with the mapping of the desired behavior to documents and analytical tools, as well as with passing the translated information to subsequent parties involved in the design and development of the distributed application. Understanding the roles they play leads to evaluating the impact of their actions in the operability and security of the resulting system, with effects in risk assessment and damage control. In order of the functional to technical orientation that activities they are responsible for show:

- *users* or *customers* that will operate directly on the application or supervise composing processes as part of their activity; in fully automated distributed applications, in which all exterior interactions are managed by communicating with automated entities, the user role is played by the DIA operator, as he is competent in describing desired functions; in systems that include human interaction, the users are distinguished by their actions, hierarchical and operational position in the organization, security access and profession, as they are relied upon in identifying the needs that are to be addressed by the development of the DIA;
- *executive* and *management* operatives in organizations and developing companies, as the decisions they take in budgeting and organizing the covering of lacking functionalities impacts on the format and technologies used in the building of the distributed application as well as inducing or eliminating development risks; the organizing of teams and intermediation of their interactions is a factor in the evolution of the application and quality of implementation; project management is responsible with assigning the financial, technical and human resources throughout the analysis, design and development stages, as well as with evaluating and reporting progress;
- *business analysts*, tasked with interfacing between users and design teams and determining the nature, order, impact of user activities, as well as the formatting and validation of requirements for information operated upon; documentation composed at this step serves as future reference in the development of the functional test scenarios, as well as in determining the degree of fidelity between DIA component behavior and initial requirements.

Dealing with the usage of software structures, concepts and technologies that define distributed application entities, activities and interactions, as well as with the preliminary testing of the functionalities and subsequent treatment of technical or logical errors, developers rely on functional specifications and design documentation in building the components as well as defining the structures and formats in communication, security and user management through the distributed application. The roles that development teams play are rarely limited to one type of technological approach. The

DOI: 10.12948/issn14531305/17.2.2013.13

following section describes the areas of activity for this stage:

- user-oriented *presentation layer design*, with user interface control properties, validation, screen positioning and order, as well as the structuring of this content by level of access to the functions of the system, contributing to the performance of the human-distributed application interaction, training efficiency, quality and security of the information; user interface designers are required to translate user specifications into ergonomic controls and structuring of operations in a continuous way, including existing functionalities and design specifications in the analysis of the future formats of the elements; MERICS.WEBAPP, Web-based component of the MERICS system, includes wizards that, upon successful authentication and access rights assessment, guide the user in the choosing of study items, options and parameters of the methods, associated to description paragraphs that inform on the semantics of each possible action; upon computation of the results, they are shown together with information of selected method arguments; if the operation or multimedia content allows it, it is visually represented in both initial form and highlighting the differences; the user is allowed the option of retracing its steps, modifying parameters and resubmitting requests, yet while ensuring that he has specified all the prerequisites; considerations in the design of graphical user interfaces also relate to the level of technical knowledge on the DIA domain of activity that interacting persons possess, as well as their cultural background, with differences in languages used in labels and user messages, mirroring, order of controls and description of operations;
- *service layer programming*, which results in the building of the automated interface components that interact with the exterior through messages containing operational data packages, as well as meta-data on the session and emitter identity; configuration files, records and preprogrammed functions describe the rules of the exchange, techniques for encrypting the operational information and message, as well as the data structures that form the basis in the description of methods and objectives; developer responsibilities are linked to ensuring the optimum usage of communication endpoints, limiting the number of interactions by grouping of data in predetermined structures, caching information that is frequently used in order to reduce processing time and message size, validating the input in format and content, managing the logical layer interactions, programming the tools required in service authentication, cryptographic processes and credential exchange; MERICS implements decoupling in algorithm specifications and content; method parameters or services do not include image or frame content, which is stored in the database and referred through session and file identifiers;
- *logic and control component construction*, done through the description of parameters and implementing of algorithms that manage projected user activities and processes; this development layer contains the highest variance in technologies, frameworks and programming paradigms, which make its components vulnerable to quality-related discrepancies [5] between the projected and implemented operational behavior; in order to address these risks, programming activity is managed by using assemblies and packages that provide structural clarity, diminish redundancy and resources spent, as well as cross-component dependency; in large projects or development teams, there occur instances in which the programming team's activity is limited to one such role

DOI: 10.12948/issn14531305/17.2.2013.13

– building communication components, business objects hierarchies;

- *persistence structures development*, with the building of database objects, stored procedures and functions, triggers, indexes, auditing tools, database communication endpoints, as well as overlapping constructs that manage logical layer interactions and provision for object – relational translations in data structures; MERICS decouples analytical from operational content storage and processing, with database-level exchanges limited to asynchronous processes, which offer the advantage of scheduling in intervals with limited processing strain on the system; developers are required to structure information and implement database schemas partially based on the composition of higher-level entities such as objects and attributes in programming constructs, which relate to tables and columns; they include mapping, communication and archiving functions in assemblies that encapsulate databases in an architectural representation and provide the formatting needed in translating from logical layer to database objects.

The development teams are the first to test the new functionalities, as debugging and scenarios described in the specifications serve the elimination of software errors and the detection of behavior unpredicted or unaccounted for in earlier stages, such as computation delays.

Developers are responsible for acting on the feedback that testing teams and users provide in order to repair malfunctioning components and reduce processing resource usage discrepancies, optimizing the behavior of the distributed application and providing insight on the technical elements that positively or negatively affect the way in which applications respond to requests.

Extensions and modifications on the distributed application, decided at later stages during its life due to moral usage, changes in organization activity, support platforms and technologies or due to requests for the integration of new features [6], are done by retracing the preliminary steps including development.

2 DIA Security

The Oxford English Dictionary defines *risk* [7] as a *situation involving exposure to danger, the possibility that something unpleasant or unwelcome will happen, a person or thing regarded as a threat or likely source of danger, a possibility of harm or damage against which something is insured, a person or thing regarded as likely to turn out well or badly in a particular context, the possibility of financial loss.*

Risks relate to concepts in the Probability Theory by the implicit association to the probability of occurrence for these undesirable events, and by the usage of the latter's concepts and functions in analyzing deterministic or random processes.

The frequency of occurrence, the approximation or precise determination of damage and resulting recovery costs, as well as the context in which the loss-causing events take place are factors included in classifying IT risks.

Risks define probabilities of occurrence for events resulting in losses to the quality of information, operations and budgets for operating parties. DIA analysis is to be included in their assessment, although effects are only observed in later stages or final version of the distributed application.

The evaluation of needs, funding, interacting actors, management decisions play a role in the frequency and relevance of incidents [8]. Due to the absence of specifications and generic approach that parties have in addressing the form of the final product, this stage is prone to the appearance of *qualitative risks*, as defined in the subjectively valued effects they trigger:

- the *lack of inter-department communication* in the company or organization that serves as the DIA beneficiary, due to insufficient knowledge dissemination, differences in areas of activity, order of interaction in

DOI: 10.12948/issn14531305/17.2.2013.13

the functional use cases that form the activity domain for the entire group, lack of ethical codes, limits on their ability to disclose information, resulting in the insufficient detailing of business structures and methods; a complete application domain related mapping of cross-organization responsibilities and procedures are required in order to aggregate functions; in citizen-oriented e-government portals, the specifics of each operation designed are detailed by the public organization that uses them, yet the cross-platform communication and person information specifics are shared;

- the *inability of users to specify the desired functions* that the application should perform in helping with their activity, due to the specialization and lack of general knowledge of impact on the organization; sales and accounting, or human resources are areas with limited overview on processes through the company; on the other hand, executive positions do not know the details of the activities that require modifications or extensions; design in social networking and messaging, publically available to non-technical personnel, is driven by generic user requirements or feedback – an individual wish to communicate, acquire information and share details inside networks, without specifying details on authentication, interface design, communication and encryption or logic and operations control;
- the *unavailability of performance estimators* for the system in its incipient stages [1], with impact on the overall performance and development, as lack of processing speed and memory causes vulnerabilities to denial-of-service attacks later on, or impacts the productivity of other applications accessed through the same deployment platform; overextending functions, database size or structure without the corresponding increase in processing leads to slow performance and errors due to synchronous process timeouts; MERICS implements threading and delegation of tasks in order to minimize impact on individual components.

Today's most popular approaches to software development are *Software as a Service* and *On-Demand*. Differences between the two approaches originate in the perception that the pooling of resources and income generating from the selling of license rights constitutes a superior business approach to the latter, which describes software development as being triggered by the customer.

The budget composition for the design and development of a software application is oriented towards the completion of the following tasks:

- *funding the contractual obligations*, in case the developer and user of the application represent different organizations; as delays in payment lead to operational impediments on the part of the latter, the lack of resources dedicated to implementing the application cause delays and the lowering of quality as software and documentation expenses are not manageable and vary [9]; the availability of resources and scale when compared to the company and distributed application size influence the prioritizing of development tasks and environment structure as well as availability of software and hardware;
- *acquisition of licenses* for using software platforms, requiring the identification of user roles and numbers, as well as the proprietary rights that are needed in order to operate with the distributed application; costs introduced by these reflect on the productivity requirements for the developers and final product – design instruments, database management systems, operating systems, Web servers, integrated development environments, software frameworks; Open-Source approaches to development and publishing reduce costs and improve on flexibility; proprietary software brings benefits such as customer support and backward compatibility;

DOI: 10.12948/issn14531305/17.2.2013.13

- *human resources costs* coverage – employee wages, taxes and insurance, internal and external training; coordinating activity is essential to minimizing the spending, as term differences occur in the completion of tasks and lack of balancing requirements to team size and member availability leads to over or under-assignment, in case multiple applications are developed in parallel; project management deals with the optimization of resource assignment and the prioritization of jobs; distributed applications introduce autonomy in components, which allows for multiple-choice development, as testing and deployment is not limited to one alternative;
- *security budget* – acquisition of digital certificates and information encryption technologies, biometric data handlers, externally-built auditing tools and documentation; a proactive approach to DIA vulnerabilities, risk management allows for the identification of incidents through estimating frequency and compatibility with the application's activity domain, as well as costs; developing risk assessment models allows for the optimization of response and minimization of effects; MERICS includes auditing instruments that record operational context, which serves in identifying patterns and frequency of occurrence for security-related incidents;
- *infrastructure and deployment* – the acquisition of hardware components and software platforms designed for testing and deploying the application in multiple instances as required by the system analysis and design specifications: development tests, training, user acceptance testing, production simulation and stress tests, and finally the live usage platform; software development tools include components designed to emulate application servers, enabling debugging of cross-module features such as Web services; MERICS uses Visual Studio for local tests, as well as integrating components in different building stages.

Determining the development and usage budget of a distributed application impacts the efficiency and quality of the solution, security and operational risk management and software constructs, hardware used for development and testing, as well as the number and technical quality of implementation teams. Maintenance tasks and administration funds affect the incident response times and backup capabilities, as higher amounts allocated to these tasks ensure the acquisition of auditing tools and personnel that interact in order to quickly and safely eliminate the negative effects.

3 Security Processes and Constructs

Defining the operational requirements that constitute the basis in DIA development constitutes the first step in the formalization of data and behavior structures that characterize a software system. Assigned to the interaction between business and system analysts, users and decision factors, this stage results in the completion of documentation that describe the items that need to be constructed in order to facilitate the before-mentioned user activities. As parties pool their resources together, the improper identification of business requirements and logic behind leads to:

- *increased development time and subsequent costs*, as discrepancies between projected and obtained outcomes are tracked to the analysis and reevaluated, with the involvement of each actor in the stages that the DIA has passed through; if an application targeting online messaging fails to provide the users with a message querying function, the feature's translation into data structures, auditing, performance impact, communication strain need to be expressed and evaluated, access rights granted for the user roles to which the operation is relevant and testing for bugs and erroneous behavior conducted;

DOI: 10.12948/issn14531305/17.2.2013.13

- *lowering of result relevancy*, as the improper or incomplete mapping of activities to data structures and functions causes the diminishing of report relevancy, inability to project the evolution of factors, through data mining and business intelligence operations, allowance of unstructured data or invalid formats into the operational database; if a financial application's interface module fail to verify for the presence and correctness of a customer's income and address information, querying and reports on location-income correlations are irrelevant, or underperforming at best;
- *quality losses in the outcome* of the application, due to the improper evaluation and attaining of the potential for activity optimizing through DIA operations; customers and users are often unaware of industry best practices in their field or the extent to which their workload is reduced by the implementation of software constructs; alternatively, they do not know or envision in detail the interactions that take place between the departments in an organization, tending to focus on their own tasks; specialization and separation of resources are advantages in productivity improvement, yet rely on the presence and optimal functioning of coordinating elements and teams; using their experience in previous projects, developers are able to direct the customers towards the proper format; in a bank, if a marketing officer does not know or take interest in the relevancy of primary data compiled into the reports he or she constructs through application features, the developer is able to ensure on the data quality by imposing validation on the activity of the branch operator.

Processes underlying distributed application operations are characterized by the interaction with tasks, exterior applications and users and the order of precedence for the elements that compose them. *Figure 1* plays the role of a test scenario in describing the activity sequence for the enrollment of a financial operation applicant's information. The 4 swim-lanes describe the *Customer*, *Operator*, *DIA* and *exterior system* roles. As observable, the order of the data being served as input, validation performed and outcome decisions, as well as the outcome depend on the interactions and order of the steps. If the exterior system is called ahead of the validation of identification data and it proves to be inaccurate or incorrect, resources are spent on processing and payment for the service without the accountability of the output, as data return will pinpoint the error, inaccurately describe the result or the answer does not reflect the reality. If the address is not correctly copied or stored together with electronic copies of the customer identification documents, the contract signed at the completion of the task is not valid and therefore legally binding for any of the two parties. The previous scenario allows for the identification of vulnerabilities in the definition of processes for distributed applications:

- *improper identification of the order of actions* underlying the cross-application interactions; the users and business analysts are responsible with the specifying and documenting of the process interactions;
- *resource spending* and cost increase due to redundant or inaccurate messaging and computation definition, as well as improperly secured information design that allows for data theft and interaction, leading to the expense-generating unauthorized usage of application functions and decrease of analysis relevancy by the contamination of data repositories;
- *extremes in separation of logic* by excessive reuniting or decomposing of workflows, the first causing limitations in process flexibility and reusability, as well as redundancy in code definition by the reimplementation of features already present in different components, the second unnecessary complexity and

DOI: 10.12948/issn14531305/17.2.2013.13

difficulty in maintaining and updating the excessive number of units.

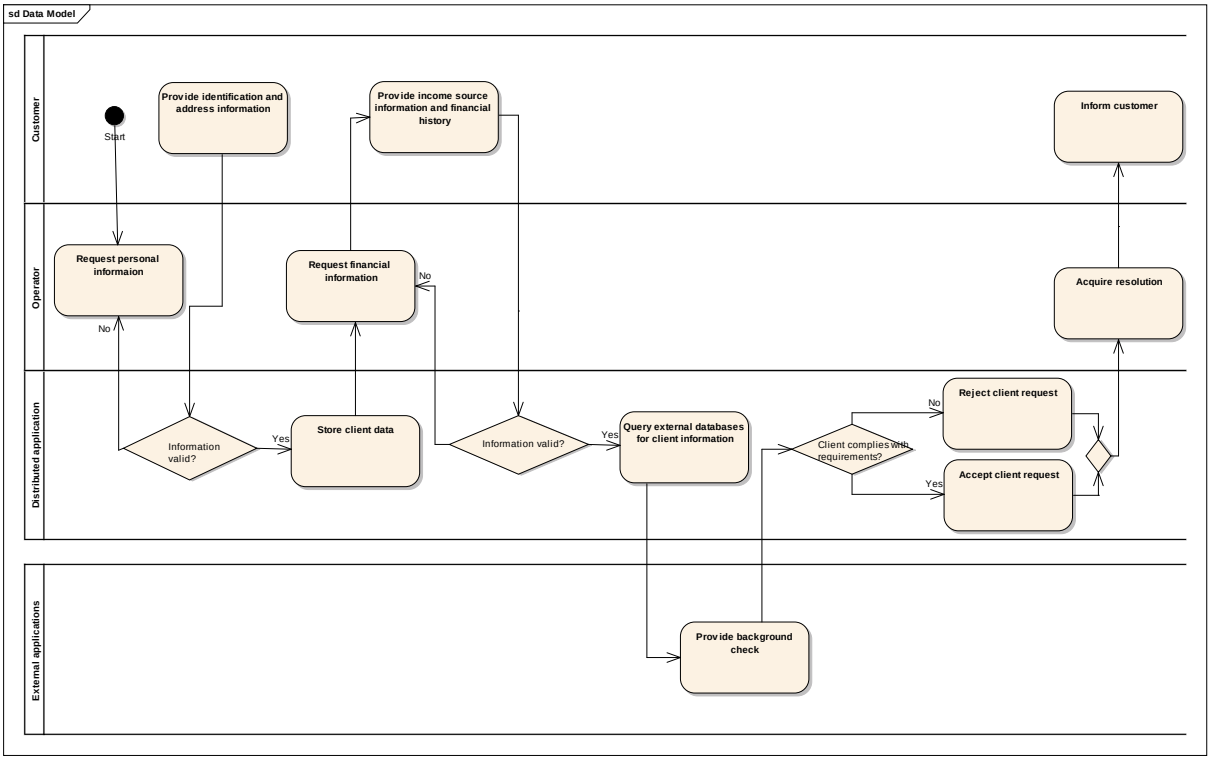

Fig. 1. Customer, operator, DIA and exterior interactions as part of financial transactions

Identifying the roles played by the users in the interactions envisioned to occur in the context of the developing DIA is affected by the amount of information available on their responsibilities and job specifications.

4 Security Factors Influence Hierarchy-Building Methods

DIA security is enhanced through the discovery of risk factors and their ordering based on the severity of the effects or incident number, the model-based description of the appearance frequency, by constructing cost assessment or qualitative impact evaluation methods. The techniques used by an organization in distributed application development impacts on its future performance.

The current study drives at understanding the behavior of Romanian IT organizations and specialists as relating to risk prevention within activities involving the design, development and usage of distributed IT&C applications - DIA.

A questionnaire containing 15 DIA features and practices, as well as contextual mechanisms related to risk assessment, was constructed in order to provide an evaluation of the priorities assigned to various practices within the before mentioned activities.

The form results were gathered from 86 post-graduate students, enrolled in Bucharest's Academy of Economic Studies - ASE, IT Security Master, in February 2013, as well as from PhD candidates and members of ASE's Economic Informatics and Cybernetics Department.

Using discrete values that specify the degree of compliance with the specified properties, as well as the constraint of assigning unique values, allow for the building of qualitative models that avoid subjective valuing within predetermined ranges, therefore serving to increasing the relevancy of the study.

5 invalid or incomplete answers were eliminated from the operations related to the processing of the questionnaire's answers, structured as presented in Table 1.

DOI: 10.12948/issn14531305/17.2.2013.13

The questionnaire is structured as follows:

Table 1. DIA development features

No.	Order the following security characteristics based on criteria related to the extent of their usage or implementation in distributed applications you have interacted with:	Order[1-15]
1	Minimizing operational logic in interface elements	
2	Ensuring backup for stored information	
3	Correlating encryption key complexity to information sensitivity	
4	Test simulations of production environments	
5	Customized software security modules or assemblies	
6	Operational risk data analysis elements [...the presence/extent of...]	
7	Error logging, including operational parameters	
8	Software structures relating to error handling	
9	Operational parameters monitoring modules	
10	The architectural separation of communication elements from UI and logical layers	
11	Limiting the extent of confidential information that is sent or received as part of operations	
12	Ensuring autonomy for the modules and applications that compose the distributed system	
13	The updating of contextual processes and products	
14	Risk assessment and incident recovery budgeting	
15	Procedural norms regarding the decoupling of environments and related -information	

The correlation between the priority assigned to a given process or DIA property and associated weights is presented in Table 2.

Table 2. Order-weight

Order	Weight
1	12
2	10
3	8
4	7
5	6
6	5
7	4
8	3
9	2
10	1
11	0
12	0
13	0
14	0
15	0

The questionnaire's content and the reasoning behind the choosing of the 15 processes or properties is detailed in the following sections. In order to minimize the influence on the manner in which respondents prioritized their DIA development aspects assessment, the questions were presented in a random order, as pertaining to the subject area.

a. *Minimizing operational logic in interface elements* refers the practice of separating the graphical user interface from operational logic, such as formulas, complex validation rules, enquiries that require the interaction with stored information. Advantages of this approach, alongside the clear structuring of the modules comprising the distributed application, include limiting the system's susceptibility to brute force attacks and hiding the implementation details from being reflected in the structure of the – potentially public – interfaces. On the other hand, development practices aimed at optimizing resource consumption and response times require that information be validated as soon as possible. User interfaces contain validation constructs such as regular expressions that provide the first level of information quality control.

b. *Ensuring backup for stored information* is a wide-spread and critical approach to managing risk. Depending on the nature of the system, however, details regarding the frequency and extent of stored information are required in order to assess its relevancy and the extent of

DOI: 10.12948/issn14531305/17.2.2013.13

protection provided. Asynchronous operations are less affected by temporary data loss or malevolent changes than synchronous, real-time services.

c. Correlating encryption key complexity to information sensitivity refers the practice of optimizing performance by correlating security measures to potential losses. Considering the nature of communication content and, in a security-oriented approach, the *lifespan* of information sensitivity as being the extent of time in which secrecy is required, encryption is implemented as a temporary measure of protection. A session-relevant content is protected in order to prevent its data from being acquired and mishandled within a shorter time-span than proprietary user information.

d. Test simulations of production environments provide developers and organizations with an evaluation of the resources needed to sustain an optimal running medium, as well as with the contextual parameters these future activities. The processing and storage capabilities of development environments are not usually strained to the extent required by the full-scale usage of the system. Susceptibility to service denial or assessment of autonomous behavior in the application's components is tested through such an approach.

e. *Customized software security modules or assemblies* indicate the priority that distributed application developers and owners assign to the building and usage of specialized software constructs, designed to protect the *core* assemblies of the system and unhindered by the operational logic processes.

f. *Operational risk data analysis elements* refer the modules that provide a global or particular view on the probability of damage to the operational state of the system or quality of information.

g. *Error logging, including operational parameters* relates to the practice of storing information relating to the nature of incidents and the application's context, session informational content and resource consumption at the moment that an error occurs.

h. *Software structures relating to error handling* refer the inclusion of predetermined or dynamic policies that condition system behavior depending on the extent and nature of an error.

i. *Operational parameters monitoring modules* relate to the previous two elements, allowing for the construction and refinement of risk assessment models, establishing correlations between the nature of an incident and the response or effects felt through-out the system's elements.

j. *The architectural separation of communication elements from UI and logical layers* allows for an increased autonomy and optimized resource consumption. Implementing specialized communication modules induces a greater degree of reusability and flexibility, as standards in messaging change independently of logical or presentation features.

k. *Limiting the extent of confidential information that is sent or received as part of operations* allows for the minimization of risks derived from unauthorized access to confidential data. The extent of the interactions that characterize the usage of a particular distributed application depends on the span of activities that the system mediates, as well as on the degree of implemented information redundancy.

l. *Ensuring autonomy for the modules and applications that compose the distributed system* allows for the minimization of incident frequency and reduces recovery costs, the errors being localized to individual components and not necessarily to the whole application.

m. *The updating of contextual processes and products* measures the degree in which organizations allocate resources to the active prevention of errors by avoiding obsolescence in the components and technologies used in both the

DOI: 10.12948/issn14531305/17.2.2013.13

development and the usage of distributed applications.

n. *Risk assessment and incident recovery budgeting* underlines an organization's awareness of the potential damages to functionality and data quality that arise from operations outside the intended scope of the application, as well as the particularities of its implementation and usage.

o. *Procedural norms regarding the decoupling of environments and related – information* ensure the protection of LIVE data and resources through the replication of development and/or testing resources outside the usage environments. This approach is challenged by external operational dependencies within testing environments and requires provisioning for stress tests and production emulation tools.

p. Restricting answers to unique values assigned through the ordering mechanism, as well as the interdependent nature of the questionnaire's items,

allows for the building and comparison of dependency graphs based on individual answers

5 Assessing the Security Process Restructuring Effort

The influence of security factors in DIAs is analyzed through the study of their components process-based interactions, resulting in the building of graphs that reflect the study respondents' view or, in operational incident analysis, of impacted components or architectural modules.

With respect to the 15 DIA characteristics present in the before mentioned questionnaire, a hierarchical ordering based on the consecutive order of appearance with respect to each other. Table 3 details the results.

Table 3. Frequency of consecutive answers

Q\Q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	11	3	5	7	7	1	6	5	9	2	3	6	3	7
2	3	0	10	7	2	6	7	3	4	6	4	9	8	6	2
3	4	6	0	12	9	3	7	1	7	5	8	3	7	2	7
4	7	2	7	0	4	5	7	6	3	3	7	8	4	7	7
5	2	5	7	2	0	6	5	10	4	5	4	5	4	5	12
6	7	3	5	3	9	0	6	4	4	4	10	1	5	8	2
7	8	5	7	3	4	12	0	11	5	3	6	5	6	1	1
8	1	7	2	3	2	4	13	0	9	3	4	8	5	2	7
9	6	3	2	6	3	5	4	10	0	6	6	4	6	7	4
10	5	9	6	6	2	4	8	4	12	0	10	3	3	1	5
11	6	6	3	6	5	6	1	3	2	10	0	11	6	4	5
12	7	5	5	6	6	4	3	6	2	9	3	0	6	7	4
13	4	4	5	6	4	8	1	3	3	5	7	8	0	7	7
14	8	5	9	8	4	2	3	3	6	4	4	5	4	0	7
15	3	4	5	2	10	3	5	5	7	3	3	6	4	14	0

This information serves in composing a graph of n-element links, as specified or obtained through operational analysis, with the purpose of determining the influence of security factors and the assessment of weights used in evaluating incident impact.

The following model details on the graph composing methods.

Let $V = \{q_1, \dots, q_n\}$ be the collection of graph nodes, corresponding to the m studied properties or processes.

Let $A = \{a_1, \dots, a_t\}, t \leq n^2 - n$, describe the collection of arcs formed by the bidirectional links between elements of V in the graph. A's elements take the form

$$a(q_i, q_j), \qquad i, j \leq n, \qquad q_i, q_j \in V,$$

where q_i is the source node and q_j represents the destination node.

DOI: 10.12948/issn14531305/17.2.2013.13

Let $G_m^n = (V, A), m \leq n$, be a digraph, where m denotes the maximum number of arc pairs, members of collection A, linking elements of V, allowed in the computing of process influence, and having n nodes.

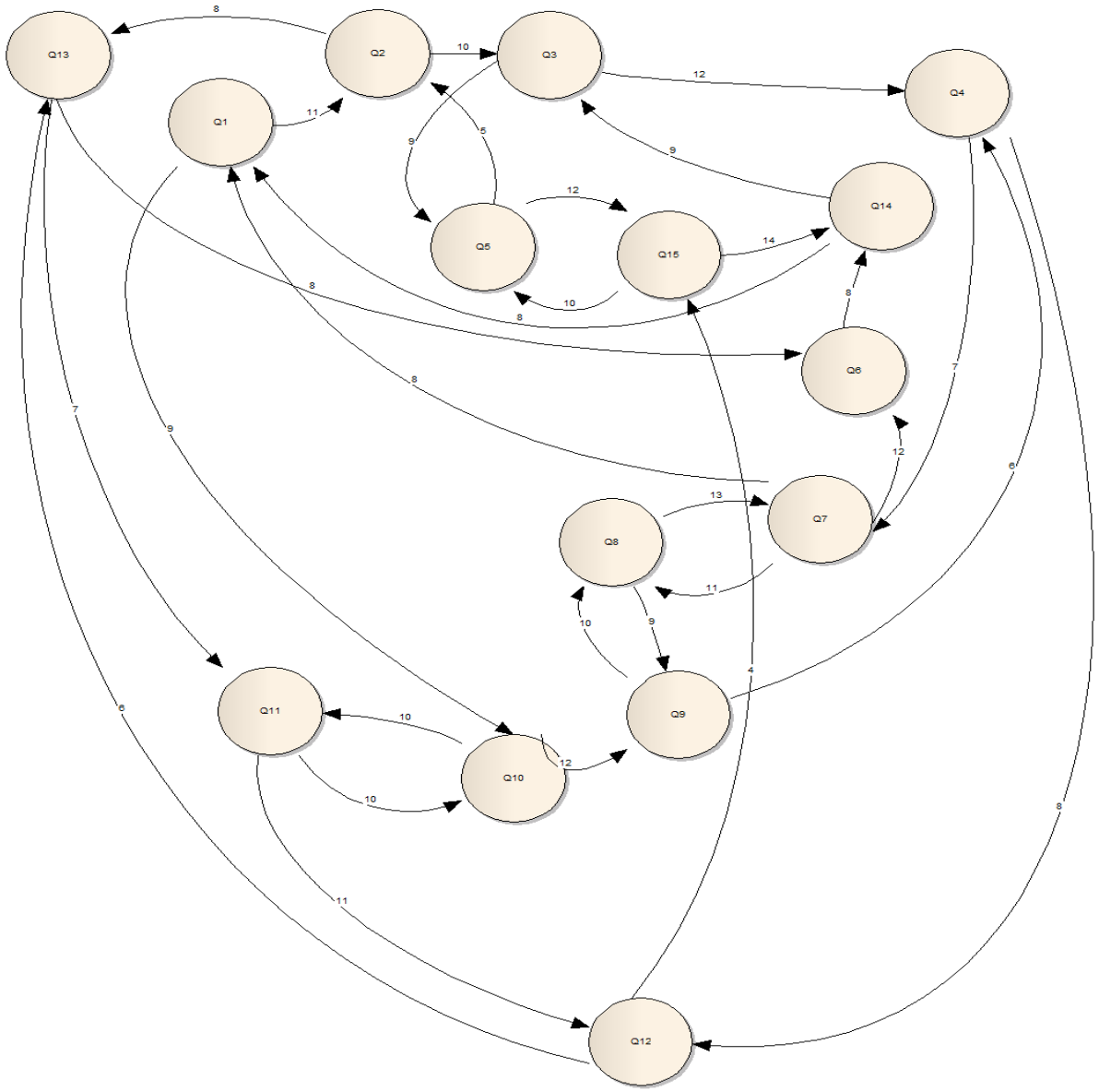

Fig. 2. Digraph G_2^{15}

The rules used in assigning the arcs are:

- all factors are represented;
- the digraph is finished when any present node has a maximum of m arc pairs assigned; if isolated nodes or sections remain, the model is not relevant for the m restriction chosen and the latter must be increased;
- the frequency scores assigned as per the methodology presented in table 3 are ordered in a decreasing manner;
- corresponding arcs are drawn, keeping the restriction of m arcs;
- in case more than m arcs have a given mode as source or destination, the ones corresponding to the highest frequencies are kept;
- if the scores of two or more candidate arcs are identical for the last remaining position, the total S score for the respective node is calculated based on the formula:

DOI: 10.12948/issn14531305/17.2.2013.13

$$S_i = \sum_{k=1}^{n} sa(i,k)\,, i \le n$$

where $sa(i,k)$ represents the score for arc $a(q_i, q_k)$. If the scores of two or more arcs are equal, the arcs are kept and the scores of destination nodes are evaluated, as completed until the current step, or by a post-evaluation if no maximum score is obtained, until a differentiating criteria is computed. The arc that leads to the highest scoring area is kept;

- the nodes remaining with an incomplete number of arcs, due to the assignment of all possible positions for source nodes, are left in their current state, as shown in Figure 2 for nodes Q_5 and Q_6.

Once the graph is completed, the effect of one factor on the others in the collection is reflected by the computation of a maximum frequency or weight sum, reflecting the manner in which they are linked, as shown by the formula

$$f(q_i, q_j) = \sum_{l=1}^{n} \sum_{u=1}^{n} sa_{ij}(l,u),$$

$$sa_{ij}(l,u) = \begin{cases} 0, & a(i,l)\ is\ not\ on\ the\ road\ between\ q_i\ and\ q_j\ or\ leads\ to\ an\ aleardy\ reffered\ node \\ sa(l,u), & a(i,l)\ is\ part\ of\ the\ road\ between\ q_i\ and\ q_j \end{cases}$$

where

$sa(l,u)$ - score for arc $a(q_l, q_u)$.

The arc scores reflect, alternatively, weights of incident impact factors in DIA components or the product between the frequency of successive hierarchy positions and the measured incident occurrence effects. Through increasing the number of maximum arcs allowed for each node, factors that influence the analyzed processes are identified.

In the maximal version, in which $G_m^n\ m = n$, a complete and symmetrical digraph is obtained, in which each node is linked to all remaining ones in the collection.The road between any two nodes has the length of 1, with the sum associated to the arcs determining the direct or indirect influence on all others.

6 Conclusions

The paper presents a model for evaluating the influence of security factors in DIAs, as well as assessing the effects or the impact that users assign to them. The questionnaire's target respondents were chosen with the aim of homogenizing the primary, unfiltered information, which has its format validated as part of the ulterior steps in the procedure.

The composing of the form used for the study included separating these factors or properties, as well as avoiding presenting them in a biased order, which could serve as a suggestion. Not grouping them in sub-categories served the same purpose. Imposing a ranking system of assessment allows for the minimization of random choice or evaluation on behalf of the respondents.

The further improving of the DIA risk assessment study through a semantic approach has the potential of helping identify and measure in a qualitative spectrum the factors that lead to damage, through the composing of patterns in user activities or DIA module response.

Comparability of the primary information is ensured through its aggregation, while keeping individual sets of answers, and by its submission to the same set of refining stages. The methodology built for DIA security influencing factors analysis is applicable to every answer grouping in the study sub-set.

Assigned weights are stable and available for all application classes only if statistical testing reveals the constant character of answers, this feature being reflected through the building of homogenous sample groups

DOI: 10.12948/issn14531305/17.2.2013.13

as related to the security personnel that serves the input.

Building a graph-based model allows for both an increased performance in visualizing the distribution and correlation of studied properties, as well as the gradual impact of these items in aspects that relate to risk occurrence in DIA security. Through extending the model by the usage of weights and its application on historic data, pathways or risk maps are derived for the modules and processes that compose distributed IT & C applications.

References

[1] M.P. Lima; J.M.N. David, B.T. Dantas, *Risk Management and Context in a Collaborative Project Management Environment for Software Development*, Brazilian Symposium of Collaborative Systems - Simposio Brasileiro de Sistemas Colaborativos (SBSC), 5-8 October 2010, pg. 95 - 102, ISBN 978-1-4244-8445-4

[2] Lina Wang; *The current situation and accounting risk prevention of Clean Development Mechanism in China*, 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 8-10 August 2011, pg. 3431 - 3434, ISBN 978-1-4577-0535-9

[3] Yu Wang; Xianguo Tuo; Taifei Zhao; *A concrete model of software risk development*, 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), 9-11 July 2010, pg. 472 - 474, ISBN 978-1-4244-5537-9

[4] Chun-Hui Wu; *Exploring impacts of software development process maturity on project risk*, IEEE International Conference on Industrial Engineering and Engineering Management. IEEM 2008, 8-11 December 2008, pg. 1033 - 1037, ISBN 978-1-4244-2629-4

[5] A. Lopez, J. Nicolas; A. Toval, *Risks and Safeguards for the Requirements Engineering Process in Global Software Development*, Fourth IEEE International Conference on Global Software Engineering. ICGSE 2009, 13-16 July 2009, pg. 394 - 399, ISBN 978-0-7695-3710-8

[6] Huo Cuifeng; Teng Yanyan; *Risks and risk management in the development of information system*, Second International Conference on Communication Systems, Networks and Applications (ICCSNA), Volume 2, June 29 – July 1 2010, pg. 345 - 349, ISBN 978-1-4244-7475-2

[7] Oxford English Dictionary, www.oed.com, accessed May 2013.

[8] Ye Tao; *A Study of Software Development Project Risk Management*, International Seminar on Future Information Technology and Management Engineering. FITME '08, 20-20 November 2008, pg. 309 - 312, ISBN 978-0-7695-3480-0

[9] M. Benaroch, A. Appari, *Financial Pricing of Software Development Risk Factors*, IEEE Software, Vol. 27, Issue 5, October 2010, pg. 65 - 73, ISSN 0740-7459

[10] A. A. Keshlaf, S. Riddle, *Risk Management for Web and Distributed Software Development Projects*, 2010 Fifth International Conference on Internet Monitoring and Protection (ICIMP), 9-15 May 2010, pg. 22 - 28,ISBN 978-1-4244-6726-6

[11] A. K. T. Hui, D.B. Liu, *A Bayesian belief network model and tool to evaluate risk and impact in software development projects*, 2004 Annual Symposium on Reliability and Maintainability- RAMS, 26-29 January 2004, pg. 297 - 301, ISBN 0-7803-8215-3

[12] S. Islam, S. H. Houmb, D. Mendez-Fernandez, M. M. A. Joarder, *Offshore-outsourced software development risk management model*, 12th International Conference on Computers and Information Technology. ICCIT '09, 21-23 December 2009, pg. 514 - 519, ISBN 978-1-4244-6281-0

[13] S. Hoermann, M. Aust, M. Schermann, H. Krcmar, *Comparing Risks in Individual Software Development and*

DOI: 10.12948/issn14531305/17.2.2013.13

Standard Software Implementation Projects: A Delphi Study, 45th Hawaii International Conference on System Science (HICSS), 4-7 January 2012, ISBN 978-1-4577-1925-7

[14] B. Boehm, J. Bhuta, *Balancing Opportunities and Risks in Component-Based Software Development*, IEEE Software, Volume 25, Issue 6, December 2008, pg. 56 - 63, ISSN 0740-7459

[15] J. A. Ibrahim, M. Majid, A. H. Hashim, 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, pg. 401 - 406, ISBN 978-1-4244-8652-6

[16] C. A. Tanasie, *Restructurarea riscurilor informatice în contextul crizei economice*, Şcoala Doctorală ASE, October 2010 workshop.

[17] C. A. Tănasie, "Post-release distributed software applications risk management," *Proceedings of the Eleventh International Conference on Informatics in Economy IE 2012*, 10-11 May 2012, ISSN 2284-7472.

Ion IVAN has graduated the Faculty of Economic Computation and Economic Cybernetics in 1970. He holds a PhD diploma in Economics from 1978 and he had gone through all didactic positions since 1970 when he joined the staff of the Bucharest Academy of Economic Studies, teaching assistant in 1970, senior lecturer in 1978, assistant professor in 1991 and full professor in 1993. Currently he is full Professor of Economic Informatics within the Department of Computer Science in Economics at Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies. He is the author of more than 25 books and over 75 journal articles in the field of software quality management, software metrics and informatics audit. His work focuses on the analysis of quality of software applications. He has participated in the scientific committee of more than 20 Conferences on Informatics and he has coordinated the appearance of 3 proceedings volumes for International Conferences. From 1994 he is PhD coordinator in the field of Economic Informatics. His main interest fields are: software metrics, optimization of informatics applications, developments and assessment of the text entities, efficiency implementation analysis of the ethical codes in informatics field, software quality management and data quality management.

Catalin Alexandru TĂNASIE is a graduate 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.

DOI: 10.12948/issn14531305/17.2.2013.13