Autotesting of the citizen oriented informatics applications

Ion IVAN, PhD Professor Economic Informatics Department
ionivan@ase.ro
Bogdan VINTILA, PhD Candidate Economic Informatics Department
vb@vintilabogdan.ro
Cristian CIUREA, PhD Candidate Economic Informatics Department
cristian.ciurea@ie.ase.ro
Dragos PALAGHITA, PhD Candidate Economic Informatics Department
mail@dragospalaghita.ro
Sorin PAVEL, PhD Candidate Economic Informatics Department
pavelsorin@gmail.com

Abstract: The objective of the auto-testing process is defined. The requirements of the testing and auto-testing processes for the citizen oriented informatics applications are presented. Differences between these processes are established and aspects regarding the auto-testing inputs, outputs and tasks for each stage of the development cycle are being detailed. For each component of the informatics application data sets are created and the auto-testing process proceeds. The team of developers compares the obtained results with the ones from specifications and proceeds to repair the errors after which the development of the software component is continued. The results of the auto-testing processes are included into the documentation that is transmitted when a software component is transferred towards an aggregation process in order to be included in a complex structure. The costs of the auto-testing processes are estimated in order to demonstrate the efficiency of the inclusion of the auto-testing as part of the citizen oriented informatics applications' development cycle.

Keywords: auto-testing, efficiency, costs, effects, software quality.

1. Specification elaboration for COIA

In [1] specifications are an extremely important component in software development. They represent the foundation of the project on which the entire software construction is built. If the foundations have weak points the entire construction will have as well. In order to ensure the fulfillment of all user requirements and to maximize the satisfaction degree felt by them, it is necessary that all their requirements are found in specifications together with methods an resolution techniques[2]. Including, besides requirements, of methods and procedures of accomplishing them which ensures obtaining in the end a complete software product, coherent, unitary and which ensures the highest degree of user satisfaction.

The specifications are constructions developed by specialists with high qualifications and rich experience and include:

- details regarding the problem which needs a solution;
- input data;
- problem models and solving procedures;
- auxiliary processing;
- control points along processing;
- result structures.

For software code elaboration there are techniques and methods [3]. Specification elaboration is directly influenced by the development method used to develop the software product. For each method the specifications are detailed with specific elements [4]. Using the *Waterfall* technique, specification definitions assumes:

- defining the problem that requires solving;
- defining the functionalities of the software product;
- defining input data;
- characterizing the target group;
- defining constraints;
- defining external entities;
- defining specific requirements.

Because the method assumes a linear approach there are no specific elements presented.

The *Prototyping* method assumes developing a prototype of the application. The prototype is used for application functionality testing and the timely identification of errors and functional deficiencies. The work flow is divided in smaller packages and is easier to manage. The prototype is the result of each work stage. By prototype the results of the respective stage are evaluated. If the functionalities are developed and the prototype does not present any dis-functionalities in the testing process the next stage begins in which supplementary functionalities are implemented. The software development process ends when the software product implements all functionalities existent in specifications and there are no problems identified in the testing process which are not solved. One of the advantages of this method is the availability at any time of the application that implements part of the functionalities. In the used specifications, this method states the prototypes that are developed and the set of functionalities each prototype possesses. For each prototype test data sets are defined.

The *incremental* assumes the realization of prototypes which implement large sets of functionalities and for developing them is necessary to establish a waterfall type workflow. Each prototype is improved as a result of a waterfall type workflow which consists of several stages. The used specifications in this method must include functionalities fulfilled by each prototype and their development methods.

The *Spiral* method assumes the development of prototypes realizing on each stage the same steps. For each stage an analysis of interested parts, objectives and restrictions is made. The work possibilities are identified and the risks are remediated. The prototype is developed and evaluated to verify if it fulfils the existing requirements in specifications for the current version. After the prototype is developed according to the specifications

the next stage is planned. In this method it is necessary for the specifications to include functionalities that different versions of prototypes must implement.

Regardless of the software development method specifications are the starting point for any software product. The high quality of specifications and respecting them leads to obtaining software products with a superior quality that ensure superior levels of user satisfaction.

Compared to software products for classical applications COIA specifications are elaborated keeping in mind several particularities like:

- the members of the user target group interact directly with the application without documentation; because COIA do not require previous user training, the specifications need to take this restriction into account and ensure the possibility of using the application by any user; the specifications must ensure obtaining friendly interfaces, clear, concise;
- the application determines resource allocation; resource allocation determines paying increase attention to the functioning of the application without any problems; a deep analysis of methods, techniques, processes and proper testing ensures that the application will function with no problems in current use;
- the efficiency of investors strictly related to the ones who are using it; because the application bring income only to the extent in which it is used by citizens, ensuring a high degree of satisfaction is vital; the user satisfaction degree is influenced by the quality of the application;
- functions online; the functioning of the application in the online environment imposes supplementary restrictions related to the possibility of data input, result display, user interaction; the specifications must insure the users that there are diverse work modalities going around the restrictions imposed by online work;
- has a very high security level; the very high security level is imposed by the fact that COIA complete resource allocation; most of the citizen oriented application imply electronic payments; if an application does not guarantee user security, operations that he executes and his resources the user will lean towards another secure application that solves the same problem;
- it is permanently available; permanent availability of the application imposes supplementary requests in the development process; specification must be very clear and the testing process; must be precisely defined such that it identifies all the dis-functionalities of the application;
- the number of user is very high; the very high number of users imposes running the application without incident; incidents of any kind lead to the orientation of users toward other application which performs correctly; the specifications must highlight all aspects of the problem that the application solves and insure an irreproachable functioning.

If the particularities of citizen oriented applications are considered in the development cycle, the resulting application has a high quality level and offers users increased satisfaction.

2. Developing the COIA project

Developing the COIA project assumes detailing specifications in order to ease the work of programmers, to decide on used technologies, to decide on owned components that will be used in the new application, to identify open source components that are to be used. Developers must elaborate code for the requirements in the specifications. The don't have the liberty to choose the programming language or used technologies only to the extent to which it is stated in the specifications. The whole COIA development process is based on specifications.

Based on the specifications:

- the processing functions are identified;
- interface types are defined;
- the application structure is sketched;
- the programming language used to develop the application is decided upon;
- the data base management system is chosen;
- owned reusable components are identified;
- usable open source components that are identified;
- the team is defined;
- the Gantt diagram is developed;
- the quality control system is established.

The processing functions are identified based on the specifications. In the specifications the target group requirements are presented. Based on these requirements the necessary processing is identified by specialists. The development of a sole requirement needs, usually, the development of multiple data processing, communication with external entities or with several data base servers. For each requirement of the target group the processing is detailed, their effects, intermediary results and connections that are made with other requirements.

Interface types must be defined in this stage even though the modules have different interfaces the application has an unitary aspect and offers access to users to all functionalities. Because the functions they fulfill and the user interaction modality differs from application to application the types of interfaces are very different [5]:

- graphical user interfaces; are used to interact with users based on inputted commands with the help of peripheral input devices and display results graphically on screen; the most used graphical interfaces are object oriented and application oriented;
- web interfaces; represent web pages through which the users communicate the input data to the application and visualize results; the web pages are visualized with a web browser, the users needing hardware equipment or supplementary software products; the new technologies as Ajax, Adobe Flex, Java, Microsoft .NET increase the degree of interactivity in interfaces of web users;
- command line; command line user interfaces allow the command input under the form of character arrays from the key board and displaying the results under the same form; these interfaces are specific to applications used for administration; an advantage of these application is the possibility of executing very detailed commands, which will cause the overcrowding of a graphical interface;

- tactile interfaces; these represent forms of returning results under tactile stimulus form; the used stimulus are forces, vibrations and movements; these application are used in simulators;
- touch based interfaces; are graphical interfaces that use a screen to capture touches for receiving commands; given the ease in interaction they are used in many domains.

In order for the application to provide users with functionalities it implements it must have an adequate interface.

The application structure is defined in this stage. The structure of informatics citizen oriented applications differ according to offered functionality and the domain in which they function [6]. Citizen oriented application have:

- simple linear structure: this assumes the existence of a series of steps which the user must follow, without comebacks to earlier steps in order to solve the problem; this structure is appropriate for simple information applications or elementary processing;
- linear structure and simple connections between components; assumes solving problems by following a succession of steps, but it is allowed to go back to the previous step in order to make modifications; this structure is more flexible then the simple linear one and applicable to a lot ore application types;
- linear structure with multiple connections; similar to the simple connection structure only going back to any previous step is possible; this facility allows rerunning the solving process of the problem regardless of the step to which it has gotten to;
- tree structure with simple connections; allows the ramification from a given node; navigation is allowed only on the vertical from the top to the bottom; this structure is appropriate to display data according to selection criteria;
- tree structure with double connections; allows progress from bottom – down and from bottom – up; this feature allows going back to previous steps and selecting branches that make other processing;
- tree structure with multiple connections; allows the displacement in all direction within the application for an increased flexibility; although displacement between modules is possible in any direction it is limited by the application logic and its well functioning; this application structure is the most complex and allows developing complete citizen oriented informatics applications.

The programming language in which the application is developed is direct influence factor to performance, security, functionality and application limits. The programming language must be chosen such that it permits the implementation of all requirements in the specifications. The used technologies that are usable through the programming language must be considered in order to avoid choosing an inadequate language. Citizen oriented application security is a very important aspect for their success, so a high security level is necessary. The programming language and used technologies must ensure data security, user, made processing, allocated resources and made transactions. If

the programming language is inadequately chosen and the application security level is low the users will lean towards another application that solves the same problem and has a higher security level.

Choosing the database management system is made taking into consideration the data needs of the informatics application. For a simple information application the data base support is low because the requirements are low. For applications that make processing, hold a history of user made operations need to store historical data, the database management system must be robust, with extended functionalities which ensures data security.

Identification of owned reusable components is given by the development of application with similar processing or functionality. For a software product development company it is very important to make an economy of resources in the development process such that the price of the finite product will be competitive and its quality unaffected. Reusing modules or code sequences developed previously leads to reducing the development time by eliminating the code elaboration code stage and the testing stage Because the component was used in another application has passed through the testing process and error correction. As much as the used models use parameterization their generality increases and the extend to which they are used in other products increases as well.

Identifying usable open source components is imposed by the short tome in which the application must be developed and by the high complexity of some modules. The open source components are developed by teams of developers and tested by the whole target group and the actualizations for error correction are often [7]. After a relatively short period from the launch the open source components are preferment, stable, secure thus reliable for inclusion in complex citizen oriented applications. Given the extensive testing to which these components are undergoing it is not necessary for the developers of the application to retake the whole testing process and only specific aspects of the developed application.

Defining the work team is made in accordance to size, complexity and time allocated to the development of the informatics application. The team must be sufficiently large in order to allow the development of the application without overworking its members. Also the team must contain persons with training that allows them to solve the processing required by the application.

The Gantt graph is planning and control instrument for the development project activities. The graph contains activities, lengths, which succeed in order of dependencies such that the necessary time is reduced. The graph is also an instrument for project evolution control and completion degree..

The quality control system is necessary to ensure a high level of quality for citizen oriented informatics applications. The quality control system must insure the correct functioning of the application as well as the characteristics of citizen oriented informatics applications. A well defined quality control system eliminates application dis-functionalities and ensures a high level of user satisfaction.

After the completion of the COIA project code elaboration is the next stage where the application will be defined through modules and programs.

3. Code elaboration

Code elaboration is the stage in the development cycle in which the components of the developed application catch shape. The programmers write code according to specifications and develop modules which communicate in order to form the application like a whole. The results of this stage although according to specifications differ considerably from programmer to programmer according to the level of training and experience. High level programmers know the programming language in detail and implement the requirements made in the specifications efficiently from a resource and time point of view. The experience leads to rapidly solving problems and choosing the best solutions.

In order to obtain good results in the code development stage the specifications must be complete and correct. If the specifications are incomplete and incorrect the code elaboration is error prone:

- incomplete modules; these offer part of the specification requested functionality in a correct manner, but not all processing is made by the module and all results are obtained;
- incorrect modules; even though processing is done by the module, part or all results are incorrect, which lead to the necessity of rebuilding specifications and retaking the code elaboration stage for the specific module;
- inefficient resource allocation; inefficient time and space resource allocation is inevitable if the specifications are not clear and concise; the processor time resource is inefficiently allocated by making redundant processing, the need to remake complex computations which justify storing intermediary results, not using new techniques in made processing; the memory space is inefficiently used by making larger than necessary allocations, by not allocation memory spaces that are not used or by doing static allocations; the quality of specifications and the experience respectively training of the programmer are very important for an efficient allocation of time, processor and memory allocations;
- incomplete or incorrect process results; incomplete results are caused by making computations only to a specific point of the computational expression; incorrect results are caused by a misevaluation of expressions; incomplete or incorrect results that propagate through the application lead do obtaining final erroneous results and thus to the lack of correctness in the application; the testing process is very important in identifying the incomplete or incorrect results of the made processing;
- other processing results than expected; these are caused by the ambiguity of specifications; the programmer uses other data and processing then the correct ones; in order to avoid such incidents it is necessary to review the specifications and rebuild the specific module;
- the lack of security in the resulting module; for any modern informatics application security is a very important aspect because the users avoid unsecure applications; ensuring security is done by detailing in the specifications the requests and used technologies; inadequate code elaboration for ensuring security and not considering all vulnerabilities of the module leads to developing unsecure modules and thus decreasing the application security level; for citizen oriented

informatics applications it unacceptable to include in the application an unsecure module;

- low computational precision; it is caused by using variables which do not concur with size of data that needs to be stored; it is necessary to consider the value range that variables accept and use in computation types of data of adequate sizes; in case of making intermediary computations there is the possibility that the variables that store them, event though they have the same operands, will not be able to store the result of the operation; in this case larger data types must be used; using low granularity data types leads to precision reduction; precision is important because in citizen oriented applications the number of transactions is high and procession errors get propagated and lead to substantial errors on a given time period;
- limited modules; the limitation is given by the number of simultaneous processing that the module undertakes, by the number of connections for communication, by the parameterization capacity; the number or simultaneous processing is restricted by using static variables, by resource inefficient use, by exclusive access to certain resources; the parameterization capacity is important for the update and maintenance process of the module; a high degree of parameterization leads to short term maintenance process which bare a reduced complexity.

Developing quality specifications and following them by programmers lead to obtaining a high quality informatics application that respects user demands and ensures a high degree of satisfaction.

In order to bring efficiency in code elaboration programmers use integration environments for software development. These are instruments for software development used to assist the code elaboration process and developing final applications. Integrated development environments include[8]:

- source code editor;
- compiler;
- automated construction instruments;
- debugging instruments.

The source code editor in an instrument that offers complex instruments for editing source code. Object oriented programming assumes defining complex classes, object instantiation, processing and working with complex data types. The code editor ensures visual elements which improve the programmers' efficiency. Recognizing key words of the language used improves code visibility and helps isolate certain code segments that make specific processing. Syntax errors are highlighted by the editor. Not all logical errors are signaled by the source code editor. Another important function of the source code editor is auto-complete. This assumes displaying available options filtered according to the already inputted source code. Most source code editors offer the possibility of editing more files simultaneously.

The compiler is the software instrument that compiles the source code file in order for it to generate an executable file or a dynamic linked library. The compiler is platform specific.

The automated construction completes the automation of the applications' or module construction process. They automatically execute the compiler, link-editor and other

instruments necessary for building the application. If there are construction errors these are signaled to the programmer in a friendly manner.
Debugging instruments are very important for programmers because they allow them to monitor the behavior of the application in the testing stage. Testing ensures the identification of logical errors.
For all programming languages there are free and with payment integrated development environments. Portability is not a problem because there are integrated development environments for all software platforms. Integrated development environments ensure team work for large and very large projects. Files are stored centralized on a server and team members work simultaneously.
Integrated development environments lead to time economy in the code elaboration and in the testing stage because a lot more errors get noticed in due time.

Code elaboration is as specification definition influenced by the development methodology used. For the *Waterfall* method code elaboration is a repetitive process meaning that once the code is elaborated it is modified to correct errors discovered during testing. The process repeats until there are no more errors during testing.
For the *Prototyping* method the code elaboration is repeated for error fixing in the testing process an for developing different prototypes of the application or its modules. The process repeats until the application implements all requirements in specification and the testing process discovers no more errors.
The *Incremental* method assumes the same code elaboration processes, only they take longer to complete and have lower frequencies. Incrementing the initial prototype leads to obtaining the application in the end.
The *Spiral* method assumes repetitive unfolding the code elaboration stage due to the development of multiple prototypes of the application and for error fixing.

4. The launch

The launch is the process through which the application is placed into operation, tested with real data and users are trained.
Launch tasks include:

- identifying a set of representative test users;
- activation of functions and procedures;
- effective work;
- stream activation;
- the operation under supervision;
- independent operation; the application has counters that keep a record of streams;
- the counters are analyzed to see if there are errors.

Even though the application has passed through the testing stage it is necessary to be tested with real data such that the correct functioning of the application will be certain. Identifying a set of users to complete testing with real data is very important because the members of this group must activate all the branches of this application. If the user set is not properly identified the application is not fully activated and there is the possibility of not identifying existing defects that were left out of the testing process. All functions and procedures of the application must be activated. An option of the application assumes the

application assumes activating several functions and procedures according to the parameters inputted or selected by the user. Activating all functions and procedures by repetitively using the same option of the application but with different parameters leads to total testing and ensuring a high degree of quality.
Users of the test set work effectively with the informatics application and signal all problems they encounter during use. The development team fixes the problems that the users face and come back with an improved version of the application that goes through the same testing procedures using real data. The effective working procedure is very important because when considering distributed application its behavior is analyzed when a large number of users access it. Although a module works correctly under the development team testing procedure, when it is used simultaneously by other users there is the possibility of problems. Testing in these conditions is either real by the activation of the system by a lot of users, either simulated by sending multiple requests to the module by a specialized testing program.
The streams inside the application must also be analyzed in order to have the certitude that there are no limitations. It is possible that in some cases the application components are overcrowded and thus the data stream will produce deadlocks. In order to avoid this situation it is necessary for the application to be tested in real work conditions. Based on the real situations determined on a similar application it is possible to develop a user interaction simulation application such that the deficiencies of streams are noticed. If deadlocks appear in the streams these are fixed such that the application will correspond to user requirements.
After the problems identified by the set of test users are fixed the application is operational but its behavior is supervised by counters that register user activity. The counters are placed in decision points of the application such that the path of a user is rebuilt based on their values. The application is well developed of a high number of users leave the application through a terminal node. This proves that the respective user solved his problem, If there are nodes that are no accessed, they either make redundant processing either the users can't access them thus the application is not well developed. The node that causes the deadlock is analyzed in order to identify eventual problems that prohibit users to move forward to child node. A uniform repartition of users over the nodes of the application and a high degree of finalization of the problem characterize a well developed application that offers a high degree of user satisfaction.

5. Auto-testing the high dimension matrices computation application

The matrices computation application oriented towards large dimensions, ACMMD, created the necessary platform for auto-testing matrix computations divided in blocks. The problem of dividing matrices is raised when economical models used assume working with matrices whose number of lines or columns are in the numbers of hundreds. In the application the matrices exclusively contain real or integer, positive or negative numbers and are identified through a specific name given by the system or by the user.

The type of matrices that undergo auto-testing is independent from the testing process. This way the application receives as input diagonal matrices, predominantly

diagonal matrices, rare matrices, triangular matrices, symmetrical matrices, unity matrices and regular matrices.

The functionalities of the application allow defining and manipulation of matrices in algebraic computations, which means;

- defining the dimensions of the matrix and inputting the values of each element;
- loading predefined matrices from files;
- visualization of stored matrices, redefining and modifying them;
- uni-matrix computations like opposed matrix, inverse matrix, pseudo-inverse matrix;
- multi matrix computation like addition, multiplication or subtraction;
- result visualization.

The tree which represents the logical structure of the application for matrices computations is presented in figure 1 and has as its root the default page that any user accesses when he visits the web page of the application. From here the user chooses to benefit from the application without registering case in which he is allowed to declare an use maximum two matrices or either become a registered user and benefit from unlimited use both physically and functionally.

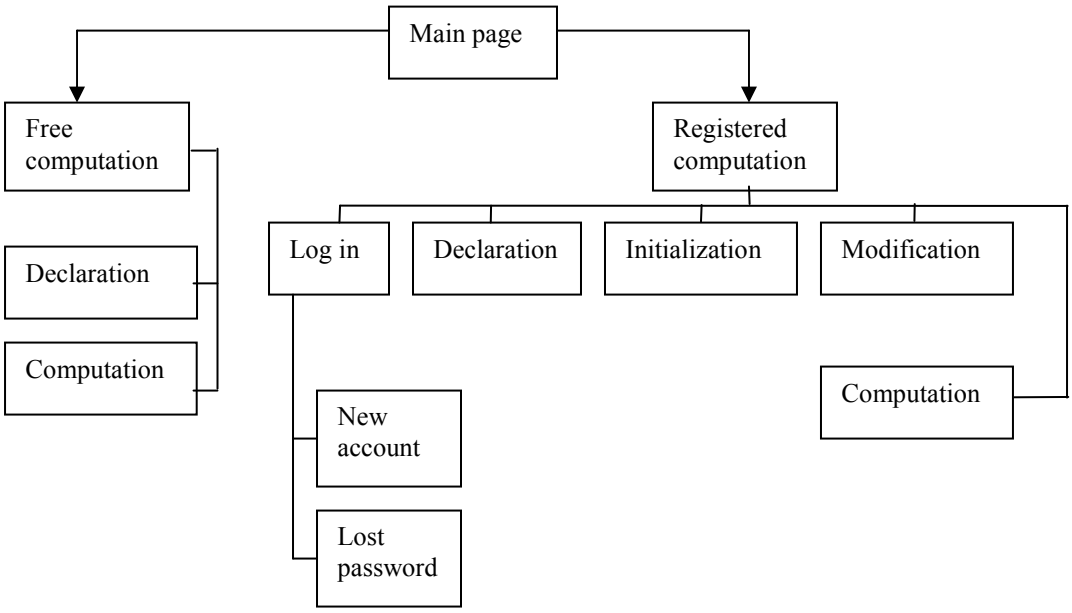

Figure 1 – Logical structure of the applciation

Auto-testing ACMMD is refers to the verification process of the correctness belonging to matrix computation after and before the result was involved in any other matrix computations. This way error propagation is avoided and a timely signal of their presence is raised.

This way in order to compute the inverse of a matrix auto-testing resources to calling a procedure to multiply the initial matrix with the resulting one. If the product matrix is the unity matrix then auto-testing proves the computation as correct. In order for

auto-testing to be complete the set of test matrices must be representative, meaning it must include:

- k, k+1, k+2, …, 2k, 2k+1, …,mk lines and columns matrices;
- diagonal matrices;
- symmetrical matrices;
- triangular matrices with non null diagonal elements;
- rare matrices;
- positively defined matrices;
- general full matrices.

Considering the test set S_1 formed from different matrices but non-singularly. The auto-testing process is presented in table 1.

Table 1 – Auto-testing process

Initial matrix	Matrix type	Inverse matrix	Auto-testing result	Interpretation
M_5	5x5 full, general	M_5^{-1}	I_5	Correct
M_{50}	50x50 full, general	M_{50}^{-1}	I_{50}	Correct
M_{61}	61x61 full, general	M_{61}^{-1}	I_{61}	Correct
$MD_{50;1}$	50x50, only main diagonal different from 0	$MD_{50;1}^{-1}$	I_{50}	Correct
$MD_{50;3}$	50x50, with a strip of 3 elements on the main diagonal different than 0	$MD_{50;3}^{-1}$	I_{50}	Correct
$MD_{61;3}$	61x61, with a strip of 3 elements on the main diagonal different than 0	$MD_{61;3}^{-1}$	I_{61}	Correct
$MS_{50;P}$	50x50 full, symmetrical considering the main diagonal	$MS_{50;P}^{-1}$	I_{50}	Correct
$MS_{61;S}$	61x61 full, symmetrical towards the secondary diagonal	$MS_{61;S}^{-1}$	I_{61}	Correct
MR_{61}	61x61 rare (minimum 16% of elements are different than 0)	MR_{61}^{-1}	I_{61}	Correct
MR_{50}	50x50 rare (minimum 16% of elements are different than 0)	MR_{50}^{-1}	I_{50}	Correct

The matrix test data set S_1 is used for auto-testing of the computation procedure of the inverse of a matrix and shows its degree of correctness. Auto-testing is implemented also for the multiplication of two matrices.

If $A*B=C$ then auto testing will check $A=C*B^{-1}$ or $B=A^{-1}*C$.

Auto-testing of the computation for the pseudo-inverse A^{+} resumes to simultaneously checking the four specific properties of the pseudo-inverse matrix:

1. $AA^{+}A=A$,where AA^{+} isn't necessarily the unity matrix;
2. $A^{+}AA^{+}=A^{+}$;
3. $(AA^{+})^{T}=AA^{+}$, meaning AA^{+} is symmetrical;
4. $(A^{+}A)^{T}=A^{+}A$, meaning $A^{+}A$ is symmetrical.

Thus the efficiency and reliability of ACMMD is increased through auto-testing even though the running time is higher.

6. Conclusions

Citizen oriented informatics application present particularities in the development process. The particularities are presented in all the stages of the development process and in order to obtain a high quality application it is needed to consider them. Specifications are seen as the building blocks on which the application is developed. Regardless how well the construction is developed if the foundation is not well developed the whole construction is in time subject to destruction. The same thing happens with informatics applications that have badly defined specifications. Even though for a short period the application has some sort of success in time its deficiencies come out and the users orientate towards other applications. The specifications for informatics applications must consider the development methodology used and adapt to it. Elaborating the application project is in turn a very important stage in the development cycle of informatics application because this is the moment in which resource allocations, term establishment, control and quality systems and structure of the application are designed. The code elaboration stage is influenced directly and to a great extent by specification quality. If the specifications were well developed the code elaboration goes along easily and the resulting components have a high quality degree. In case the specifications were not properly developed then code elaboration is tedious and the resulting components often have low quality. The launch assumes the use of the application by a set of test user and supervising the application to identify problems which come up during regular use. After the problems had been identified from current use they are fixed. Auto-testing of the matrix computation application leads to error identification and fixing. Through this the application registers an increase in quality levels and so a rise in user satisfaction.

Bibliography

[1] Software Requirements Specification. [Online]. HYPERLINK "http://sepo.spawar.navy.mil/MIL-STD-498_DIDs/DI-IPSC-81433A%20_SRS.pdf" http://sepo.spawar.navy.mil/MIL-STD-498_DIDs/DI-IPSC-81433A%20_SRS.pdf

[2] Wikipedia. [Online]. HYPERLINK "http://en.wikipedia.org/wiki/Software_engineering" http://en.wikipedia.org/wiki/Software_engineering

[3] Wikipedia. [Online]. HYPERLINK "http://en.wikipedia.org/wiki/Formal_methods" http://en.wikipedia.org/wiki/Formal_methods

[4] Wikipedia. [Online]. HYPERLINK "http://en.wikipedia.org/wiki/Software_development_methodology" http://en.wikipedia.org/wiki/Software_development_methodology

[5] Wikipedia. [Online]. HYPERLINK "http://en.wikipedia.org/wiki/User_interface" http://en.wikipedia.org/wiki/User_interface

[6] I. Ion, V. Bogdan, and P. Dragos, "Types of citizen oriented informatics applications," 2009.

[7] P. Kavanagh, *Open source software: implementation and management*, Illustrated ed. Digital Press, 2004.

[8] Wikipedia. [Online]. HYPERLINK "http://en.wikipedia.org/wiki/Integrated_development_environment" http://en.wikipedia.org/wiki/Integrated_development_environment