Quantifying Update Effects in Citizen-Oriented Software

Ion Ivan
Sorin Pavel
Academy of Economic Studies, Bucharest

Abstract. *Defining citizen-oriented software. Detailing technical issues regarding update process in this kind of software. Presenting different effects triggered by types of update. Building model for update costs estimation, including producer-side and consumer-side effects. Analyzing model applicability on INVMAT – large scale matrix inversion software. Proposing a model for update effects estimation. Specifying ways for softening effects of inaccurate updates.*

Key words: update; costs; model; online software.

JEL Codes: M15.
REL Codes: 10A.

1. Citizen-oriented software

Knowledge-based society is described by:

- generalizing participation in finding solutions to citizen own problems within human-computer interaction;
- increasing number of software applications available to citizens for assigning and managing resources, together with decreasing number of software applications destined to general information;
- building favorable context for database integration in order to reduce informational system redundancy;
- developing new applications by recycling glossaries and interfaces, in order to reduce the time spent on software adapting and to maximize interoperability;
- saving and activating data without refilling forms;
- creating software which maximizes client satisfaction and leads the producer interest towards commissions taken depending on the customer actions.

Types of citizen-oriented software applications include:

- e-commerce applications, which cover large variety of transactions: buying a book, booking a hotel room, ordering a computer component, bidding an instrument or cash withdrawing from an ATM; because of the important information exchanged through this type of transactions, e-commerce is regulated by applicable laws and is constrained to fulfill security standards; the best-known e-commerce applications are specialized in buying/selling goods or services: www.amazon.com rated as the best e-commerce site, www.bestbuy.com, www.circuitcity.com or www.quelle.com;
- software for utilities payments which offers ways to pay the bills for mobile phones, internet, electricity and so on; the strengths of these systems are given by:
  - little time for payment transaction;
  - bills archives;
  - quick visualization of the payments history or current payment state;
  - quick confirmation of payment completion;
  - real time support.

These applications are part of local government – http://plati.primarie6.ro, software companies – www.e-facturi.ro, or banks – www.citibank.ro, www.chase.com, www.pacifictrustbank.com, and implement secure methods of dealing with transactions.

- e-banking applications that integrates the two above-mentioned services, plus other types of transactions which operate directly on the personal bank account of the customer – deposits, reports, statements, currency exchanges; the user has:
  - access to his account 24/7, regardless of the bank opening hours;
  - reduced commissions than the ones available at bank counters;
  - total control upon personal account information.

Depending on the situations they are used in, e-banking services – www.ingonline.ro, www.raiffeisenonline.ro,

www.bancpost.ro – cover internet-banking, home-banking and mobile-banking;

- e-learning software for virtual education and professional distance-training that offers scheduled methods of teaching/ learning through usage of information technologies and suspension of physical presence of the students; e-learning facilities include:

- availability because teaching topics are always achievable;

- flexibility through ways of approaching topic structure depending on the students interests;

- effectiveness by granting access for all students to the same learning resources, thus reducing shipping costs;

- consistency by permanently updating and normalizing the process.

E-learning solutions are increasingly sough-after as they adapt to regulations and customer requirements: www.siveco.ro, www.academiaonline.ro, www.timsoft.ro;

- information and utility software:

- news feed (newspapers, TV, shows) which use different communication systems to broadcast news and useful information; these kind of on-line applications:

- are constantly updated for the latest news;

- keep track of news for future visualization;

- use multimedia communication – real-time photo/audio/video;

- are flexible and easy to integrate in websites and e-mail.

The most news feed software – www.stirionline.net, www.hotnews.ro, www.realitateatvonline.ro, www.evz.ro – is available for free which attracts more users than the classic ways of information;

- on-line dictionaries which provide:

- the same definitions as the printed dictionaries;

- increased searching facilities;

- minimum storage space;

- free access to information.

Depending on the nature, dictionaries are:

- explanatory – www.dexonline.ro, www.dictionary.com, www.merriam-webster.com

- multimedia – http://visual.merriam-webster.com, www.music.vt.edu/ musicdictionary/

- reference – www.reference.com

- real-time automatic translators which combine word dictionaries with grammar rules of the selected languages; although the final output is not 100% correct, automatic translators are preferred for quick and simple translations: www.translation2.paralink.com, www.etranslator.ro, www.google.translate.com.

The numerous advantages of the online applications determine the citizen to use them instead of other classic ways. The quality of these applications is given by:

- availability because the user is no more bound to the opening hours of institution counters and his needs are met any time they occur;

- viability because of their trustworthiness;
- generality because the services are for all citizens, regardless of their level of knowledge or friendliness with the online methods;
- continuity by using familiar interfaces, by emulating real life documents – which makes handling applications intuitive – and by providing non-stop support;
- portability by using communication technologies – radio waves, mobile devices – and services which detach the user from a certain place;
- correctness by managing data-flows to assure complete fulfillment of the user's objectives, without jeopardizing his informational integrity;

The high level of quality found in this kind of applications is due to the detailed process of software testing. Citizen-oriented applications follow certain steps described in the software development process:

- requirements analysis interacts with the client/citizen to record his needs and to elaborate solutions; depending on the accuracy of the use cases description, the problem is well-defined – description matches the exact functionality expected in specified situations along with details of behavior; sub-defined – certain use cases are skipped; over-defined – description contains redundant data;
- the type and destination of input data is settled, the application's vocabulary is build and the output data types are designed;
- project design elaboration starts from requirements analysis and chooses the building resources concerning application architecture, programming language and technologies used;
- implementation of the product by generating or recycling code and building application modules;
- modules assembly and software crystallization;
- global testing in order to assure correct functioning with different testing methods which cover all use cases;
- documentation and system requirements papers;
- software installation and means of support/update creation.

When finished, the application is presented for customer trial. If any deficiency is detected, the software is modified.

2. Updating operations

Use cases of citizen-oriented software generate payments, allocate resources or information for decision taking and reflect laws, regulations, formulae, needed data, market or policies modification.

The life cycle of this kind of software includes:

- creation, which supposes following the software development process with its specific phases and procedures and has the software product as output;

- update, which means software personalization in order to maximize effective use of specific customers;
- maintenance or resolving conflict situations met while using the software, caused by both internal or external sources;
- reengineering, which means changing the architecture structure, the information technologies or the algorithms used within the application;
- disabling or dropping the software because of its incapacity of fulfilling customer's needs;
- incorporation by including some modules or entire application within another software directly or with some modifications.

Updating a citizen-oriented application means:

- introducing new modules for new functionalities;
- introducing new fields within interfaces;
- deleting modules that are no longer being used;
- deleting fields from interfaces;
- introducing new data in database;
- modifying data;
- modifying authentication methods;
- introducing filters for more accurate validations in order to obtain complete data and citizen guidance within application.

The update process is completed by software producers, by authorized operators or by citizen himself and needs to be accomplished in real-time, correct, complete and verifiable because of the severe consequences in case of inaccurate update.

Types of errors that occur while updating are:

- modifying different data than expected;
- performing incomplete modification of the data;
- executing undesirable data transfers;
- hiding vital information for decision-taking.

The effects of inaccurate updates vary considering the type of application and their tracing depends on their importance within the company.

3. Update effects

The update process progress and its output have a notable impact on any company activity. In case of a correct, complete and synchronized update:

- sales grow;
- stocks decrease – especially hard-selling products;
- customer satisfaction rise;
- all resources are assigned;
- savings are kept unaltered.

Effects of incorrect updates vary on the type of application where they occur:

- in a product price management application:

- unwanted products are being sold cheaper because discounts are applied to product P_i instead of product P_j;

- differences are recorded between the price at the store shelf and the price at the check-out;
- profit is decreasing.

▪ in a railway ticketing software:

- tickets are bought for inexistent trains;
- tickets are bought for already taken seats;
- tickets are bought for wrong level of comfort (class ½);
- the customer is specified a wrong time of train departure/arrival.

▪ in an admission exam listing application:

- names are not listed;
- names are listed within wrong faculty;
- names are misspelled;
- grades are incorrect.

Corrections required by inaccurate updates are:

▪ reversible – the customer benefits from software actions even though delayed;

▪ irreversible

- the customer is paid back the money;
- the customer has lost all resources invested.

For update effects quantification, the following notations are used:

Q – total amount of products – by products meaning whatever data, record or object that can be updated;

N – amount of products that have to be updated;

M – amount of products updated;

$\Delta = N - M$

$\Delta > 0$ – there are Δ products out of N, which, although they had to be updated, they have not.

$\Delta < 0$ – there are Δ products besides N that have been updated additionally;

$P_1, P_2, ..., P_M$ – old values of updated products;

$X_1, X_2, ..., X_N$ – new values needed to be introduced;

$R_1, R_2, ..., R_M$ – new values updated;

K – amount of products which have the updated value higher than the intended value;

L – amount of products which have the updated value lower than the intended value;

C – amount of products correctly updated.

Considering the case where update objects are products and their value is their price, the effects are studied for higher updating – new values are higher than old ones – and lower updating – new values are lower than old ones.

The $X_i < P_i$ case describes price dropping because of the stock liquidation or special offers. If $R_i < X_i < P_i$, then the new value updated is lower than the one desired. Loss quantification is given by following formula:

$$PP = \sum_{i=1}^{L} V_i (X_i - R_i)$$

where:

V_i – amount of product i sold;

Although small prices attract buyers, the profit of the company is decreasing below the expected level with every product sold.

If $X_i < R_i < P_i$, the updated value is higher than expected but lower than the old value. The loss is measured by:

$$PP = \sum_{i=1}^{K} V_i(R_i - X_i)$$

The loss is smaller than the first case, but:

- buyers don't fully appreciate the discounts/the offer;
- the price policy objectives (buyer attraction, stock liquidation) are not fulfilled.

From the customer's point of view, he pays more than is needed and has to be

- returned the money;
- save the difference for future shopping.

If $X_i < P_i < R_i$, the actual updated value is higher than the old value, although it should have been lower – a price cut was intended. The loss formula is the same:

$$PP = \sum_{i=1}^{K} V_i(R_i - X_i)$$

The effects are emphasized:

- demand for goods is dropping, the buyers are aiming other goods;
- stocks are growing;
- profit decreases as the products are not sold, because a price-cut was expected.

The buyers are paying more, losing money while the seller has to compensate them.

In the $X_i = R_i < P_i$ case, the actual updated value is the same with the intended one. No losses are recorded and chances that the company's price policy is fulfilled are growing. Sales expand and stocks decrease while the long-term profit level appreciates.

The price growth is described by $X_i > P_i$ – the new updated value is higher than the old one. If $R_i > X_i > P_i$, then the new value is higher than the intended one. The prices are growing more than expected. Each product sold generates a surplus given by:

$$S = \sum_{i=1}^{K} V_i(R_i - X_i)$$

This artificial price growth leads to an increased profit only in the case of un-substitute goods. Generally speaking, any rootless price growth generates loss for both sides: sellers and buyers.

$X_i > R_i > P_i$ means that the updated value is higher than the old value but still lower than the intended value. This situation creates also a difference of:

$$S = \sum_{i=1}^{L} V_i(X_i - R_i)$$

The company price policy is not finding its target.

If $X_i > P_i > R_i$, the new value, instead of being higher, is lower than the old one. The surplus increases:

$$S = \sum_{i=1}^{L} V_i(X_i - R_i)$$

The effects are opposite than the ones expected:

- demand grows;

- stocks drop;
- sales increase;
- profit decrease compared with the foreseen value.

For $X_i = R_i > P_i$, the updated value is similar with the one expected. No loss or surplus is recorded. The company's price policy is reached and the company's profit increase on the long term.

Table 1 records a database containing the products that have to be updated, their old price, the new one and the one intended:

Updated values

Table 1

Product	Old price	New updated price	New intended price
P_1	100	50	150
P_2	200	250	100
P_3	100	100	150
P_4	100	100	90
P_5	100	100	100

For P_5 the update is correct and complete. The new updated value is the one intended and the one needed. This is the ideal situation because it maximizes the profit, it sustains the company's price policy and minimizes customer dissatisfaction.

There are products that have differences between recorded and intended values. If the modification consists of a price growth – P_1, P_3 – the company is losing money for each sold product. If the price is cut – P_2, P_4 – it leads to surplus. Both cases generate deficiencies for the customer and the company.

If the update is not completed – P_3, P_4 – all intended effects are canceled.

For N updatable products, the following cases are possible:

- C products are correctly updated; if N = C, the update is complete and correct with the above-mentioned effects;
- K products are updated with a higher value than in the documents, but the rest N – K = C are correct;
- L products are updated with a lower value than in the documents, but the rest N – L = C are correct;
- K products are updated with a higher value, L products with a lower value and C products are correct:

N= K + L + C;

In this case, the effects need to be treated separately. If the difference between surplus and loss is positive, meaning:

$$S = \sum_{i=1}^{K} V_i(R_i - X_i) > PP = \sum_{i=1}^{L} V_i(X_i - R_i)$$

then the effects are monetary surplus accumulation, unsold stocks existence, profit outrun and price policy miss.

If the loss is greater than surplus:

$$PP = \sum_{i=1}^{L} V_i(X_i - R_i) > S = \sum_{i=1}^{K} V_i(R_i - X_i)$$

Then the demand grows, the stocks decrease, the profit shrinks and the objectives of price policy are missed.

The update cost includes:

- writing source-codes expenses;
- database updating expenses;
- correction expenses;
- cheap products generated losses;
- transportation, waiting expenses;

- customer letdown losses;
- expenses for software errors
- wrong formulae;
- wrong dataflow;

Computing expenses model requires the exact number of customers N, the number of correctly updated products P and the real costs C:

$$CC = \sum_{\substack{i=1 \\ j=1}}^{N,P} n_i \times v_j + v_{P+1}$$

where:

n_i – number of products bought by client i, i = 1...N;

v_i – difference between intended value and real price of product i, i=1…P;

v_{P+1} – fixed expense for repairing mistake.

If the expected company profit is Pr and the expenses caused by the inaccurate update are bigger, the following relation is true:

$$\sum_{\substack{i=1 \\ j=1}}^{N,P} n_i v_j + v_{P+1} > \sum_{\substack{i=1 \\ j=1}}^{N} n_i d_j$$

where:

d_j – difference between old value and intended new value.

It results that if every update satisfies the relation – update expenses outrun the profit – the update is ineffective.

4. Update effects estimation for INVMAT application

Citizen-oriented software is first and fore mostly preoccupied about the specific needs of the customer. INVMAT is an application designed to automate matrix calculations where user declares, initializes and manipulates matrices.

When declaring the matrices, the user chooses the type of account he is using: FREE account or PLUS account. The FREE account lets the user declare only two matrices – named A and B – and their maxim dimensions are 10x10. The PLUS account lets the user declare how many matrices he needs and name them appropriately. The declaration is made considering the form of the matrix: unit matrix, rare matrix, diagonal matrix, symmetric matrix. The upgrade from FREE to PLUS account is done by registering the user (with a username and password).

The INVMAT software functionalities include:

- defining matrix dimensions;
- specifying every element's value;
- loading matrix values from file;
- visualization of matrices;
- updating the matrices;
- unimatrix calculi: opposite matrix, inverse matrix, pseudoinverse matrix;
- multimatrix calculi: addition, subtraction, multiplication;
- visualization of results.

The meaning of matrix values is defined by the users: prices, consumption values, rates, budgets etc.

The matrix below contains the utilities consumption values by month for a small company within the first half of the year:

Monthly utility consumptions in monetary units

Table 2

	January	February	Mars	April	May	June
Rent	300	300	300	300	300	300
Water	120	123	117	121	119	122
Electricity	79	82	77	78	81	80
Gas	12	15	11	13	15	12
Parking	50	50	50	50	50	50
Other utilities	8	9	12	9	2	10

For the second half of the year, the rent and the parking fees are increasing. The update is faulty done. The real new values are: 370 mu for rent and 70 mu for parking. The new updated values are: 340 mu for rent and 80 mu for parking.

By applying the above-mentioned formula:

$$S = \sum_{i=1}^{L} V_i(X_i - R_i)$$

It results an annual difference of 80 mu for rent and 60 mu for parking.

The differences are projected in the company's budget and leads to differences in the assets and liabilities balance. The cover expenses will affect the planned budget level and the expenses policy of the company.

5. Ways of softening the inaccurate updates effects

In order to define accurate/inaccurate updates, let T be a text that implies actions A_1, A_2, A_3, ..., A_k. If a person operates on the database actions A_1, A_2, A_3, ..., A_k, then the update is accurate. If the actions performed are A_1, A_2, ..., A_{i-1}, A_{i+1}, ..., A_k then the update is partially accurate. If for every A_i where i = 1,k the following relation is true: $A_i \cap B = \emptyset$, where B is the performed operation vector – B_1, B_2, ..., B_j – the update is inaccurate.

The inaccurate updates are caused by both the subjective elements of the operator:

- incomplete receiving of the input documents containing updatable values;
- incomplete cover of the list with the updatable values;
- negligence in introducing new values;
- unknowing the update procedure;
- lack of saving the new value;
- updating the wrong database;

and the objective elements of the computer or surroundings:

- absence of database/internet connection;
- using default settings;
- absence of automated saving;
- interrupting the update process due to time expiration/ electricity fluctuation;

- existence of an input device failure.

In order to avoid and reduce inaccurate updates, these causes must be treated and diminished by:

- thoroughly documentation of the update process: defining update's objectives, methods and implied parts;
- clear specification of updatable objects, of the old values and the new ones;
- authorizing update documentation – management validation of the process and values;
- assigning the proper person for implementing the update process;
- localizing updatable objects and assuring connection;
- marking updated objects on the update documentation;
- verifying values after update;
- completing update process – saving, form filling, disconnection;
- Assuring quality computers, UPS devices, external memory;

There are even more ways to verify and control the update process, ways that connects with the software structure/ functionalities:

- authorizing the person that updates, to avoid modifications from an unqualified person and to record the author of the update;
- validating the new input values by comparing them with a certain type, range, minimum/maximum value, standard deviation etc.;
- implementing automatic saving;
- error signaling with proper messages;
- interrupting updates when connection with database is lost;
- creating back-up copies of the old values;
- keeping update logs.

Following the above-mentioned methods does not exclude the human error element, but it diminishes the chances of generating large-scale effects in the update process.

6. Conclusion

Citizen-oriented software is becoming more and more used and developed, which draws even more attention to its core elements. Update is part of the important processes in software lifecycle. Its consequences lead to either improved performance or to severe errors for both company and customer.

To prevent undesired situations, it is recommended to stress the software testing processes, despite costs increasing. The diversity of the test datasets should be aimed along with the software structure coverage. The software audit role increases while the software quality is improved.

The present article is part of research on techniques of large, inhomogeneous datasets oriented software development, within Phd. school, contract no. 50/oct. 1, 2008.

References

Ivan, I., Milodin, D., Popa, M., „Operation on Text Entities", *Informatică Economică*, vol. 11, nr. 1, 2007, pp. 14-20, ISSN 1453-1305

Ivan, I., Boja, C., „Practica optimizării aplicațiilor informatice", ASE, București, 2007, ISBN 978-973-594-932-7

Ivan, I., Boja, C., „Optimizarea bicriterială a software, *Informatică Economică*, vol. 10, nr. 1, 2006, pp. 17-24, ISSN 1453-1305

Ivan, I., Amancei, C. (2006). *Stabilitatea coeficienților modelului global de calitate software*, ASE, București

Ivan, I, Rădulescu, I. (2006). *Analiza comparată a complexității entităților text Generate prin tehnici de programare*, ASE, București 2006, p. 164, ISBN 973-594-754-4

Ivan, I., Noșca, Gh., Popa, M. (2006). *Managementul calității aplicațiilor informatice*, ASE, București

Ivan, I., Cristescu, M., Popa, M., „Modele pentru estimarea costului fiabilității software", *Revista Română de Informatică și Automatică*, vol. 15, nr. 1, 2005, pp. 41-46

Ivan, I., Popa, M., „Uniformitatea, caracteristica de calitate a entităților text", *Informatică Economică*, vol. 9, nr. 1, 2005

Ivan, I., Popa, M., „Entități text, dezvoltare, evaluare, analiză", ASE, București, 2005, ISBN 973-594-663-7, p. 587

Ivan, I., Boja, C. (2004). *Statistical Methods in Software Analysis*, ASE, București

Ivan, I., Pocatilu, P., Amitroaie, M., „Metrici ale societății informaționale", *Informatică Economică*, vol. 5, nr. 4, 2001, pp. 33-40

Ivan, I., Teodorescu, L. (1999). *Software Quality Management*, INFOREC, București

Maydanchik, A., „Data Quality Assesment*", Tehnics Publications LLC*, 2007

Goodchild, M.F., Clarke, K.C. (2002). *Data Quality in Massive Data Sets*, Kluwer Academic Publishers

Beall, J., „Metadata and Data Quality Problems in the Digital Library", *Journal of Digital Information*, Vol. 6, Nr. 3, iunie 2005

Herzog, T.N., Scheuren, F.J., Winkler, W.E. (2008). *Data Quality and Record Linkage Tehniques*, Springer New York