Evaluation of Normalized Systems Theory applied on Evolutionary IS Implementation

André Filipe Saraiva Varela
56918, Master in Information Systems and Computer Engineering, Instituto Superior Técnico, Alameda

Abstract. Nowadays, to follow continuously changing markets, organizations need to make changes and increase complexity of their Information Systems (IS). These systems, nevertheless, have plenty of dependencies that make changes harder. This dissertation goal is to evaluate if the Normalized Systems theory (NST) allows organizations to deliver fast and effective changes, as well as increase the complexity of the Information Systems (e.g. increase number of functionalities, requirements and interfaces) - making them agile and evolutionary. The proposed methodology consists in applying NST in the development of an E-Voting System (EVS). Afterwards, by using another system (similar to the one developed) it was possible to prove the usefulness of the theory in the development of evolutionary information systems, through the analysis of some quantitative software evaluation metrics (applied to both).

Keywords: Normalized Systems theory, Complexity, Change, Agility, Evolution, Information System, Electronic Voting

1 Introduction

Nowadays, organizations face some problems in their IS. They have a lot of dependencies which make changes and the increase of complexity much more difficult, not allowing to respond to continuously changing markets.

Therefore, the problem addressed in this work is:

- Concerning the maintenance phase, does the application of Normalized Systems theory (to IS development) allow changes in the system and an increase on its complexity, combined with reduced time and effort?\footnote{In IS maintenance phase, effort is the difficulty to execute some tasks.}

Based on the problem presented, two research hypotheses have been developed to guide and help with the realization of this work. These hypotheses are:

H1. It’s possible to apply concepts of NST in the development of IS;
H2. It’s possible to add/change functionalities in developed IS, combined with less effort and time comparatively to another non-normalized system.

The main objective of this work is focused on finding a solution, allowing organizations to act fast and effectively in response to changes as well as the increase of complexity of the IS - making them agile and evolutionary.
In order to fulfill this objective, we intend to study the NST and its features and apply them to a case study. The research method that will be used is the Action Research [I], in order to evaluate this method’s potential in the creation of evolutionary IS.

The scope of the case study is the Electronic Voting (EV). According to that, there will be made a mapping between the e-voting business process, and the theory elements; implementation of a Normalized Voting System, based on the mapping done earlier. Finally, by using another EVS will be possible to evaluate the usage of the theory through a quantitative analysis of some software evaluation metrics, applied to both systems.

2 State-of-the-Art

In this chapter we will address the inflexibility the IS as well as some software evaluation metrics.

2.1 Information Systems Inflexibility

Modern organizations are responsible to create the most advanced products and services, always searching for new business changes. This is happening because the IS have become vital to the efficiency of most organizations.

However, as referred in Manny Lehman law [2 3], there are some maintenance problems with these IS.

This difficulty results from the overhead of dependencies that the IS have [1 5]. This way, changing or adding new business objectives is a hard task - respecting time and effort - that can lead to the development of a new system [4].

IS must support organizations that behave like agile companies, because it is the only way these organizations can act fast, in terms of complexity and change. But some mechanisms to decrease the dimension of this problem are needed. The NST - studied in this work - addresses this problem.

2.2 Software Metrics

Forthwith we will address some software evaluation metrics. Some will be used in validation of the proposed solution to measure the easiness of the software maintenance, in order to evaluate the usefulness of the NST, like those:

- Function points [7];
- Code lines [8];
- Cyclomatic complexity [9];
- Software maintenance and development costs [8 10];
- Time to conclude a task [8 11];
- Number of errors [8 11];

With the new OO programming languages, some new software metrics emerged:

- Number of methods per class [11];
- System total number of classes [11];
- Weighted methods per class [12];
- Inheritance tree depth of a class [13];
3 Normalized Systems Theory

In order to meet the work’s objectives, we aim to study the features of the NST and apply them in a case study, to evaluate this theory ability to create evolutive IS.

3.1 Applying the theory in the development of Information Systems

The NST has some characteristics for the design, development and maintenance of IS, and the systems must [13-15]:

- Be organized according to five high level elements: Data, Actions, Workflow, Connectors e Triggers;
- Support 4 theorems: Separation of Concerns, Data Release Transparency, Action Version Transparency and Separation of States;
- Be based in the concept of modularity to ease the information hiding and to allow system separation of concerns.

These characteristics allow a IS to support a set of anticipated changes.

3.2 Business Process Mapping

Next we describe how the NST elements are mapped [16], to support the business process flow.

Data Elements: every flow is combined with one and only one Data Element. This Data Element is one object with a life cycle and has an attribute - state - that is updated after each flow operation.

Action Elements: every flow operation is a sole Action Element and every Action Element is one sole Functional task.

We have the following types of action elements in business process flows:

- Standard Action: the IS does a concrete action.
- Bridge Actions: the IS creates another data element (with life cycle), that will be processed in his own flow.
- Manual Actions: a human-user is needed to do the action and define the state of the Data Element (with life cycle), through a user interface.
- External Actions: another process from a different IS, executes the action that defines the state of the Data Element (with life cycle)

Trigger Elements: a trigger element implies the existence of a timer. A timer represents a time description operating in a single Data Element (with life cycle).

Workflow Elements: a Workflow Element is responsible for the flow execution of all instances of a Data Element (with life cycle).

4 Case Study

This investigation is carried out in a business context, in partnership with the Agency of Administration Modernization - AMA\(^2\), that is responsible since 2007 to promote the Electronic Voting (EV) services development in Portugal [17].

\(^2\) The url of AMA’s website is http://www.ama.pt/
But there is indecision between the type of EVS to develop, either a "face-to-face" approach (electronic voting in a ballot box) or one that does not require user presence (ATM - Automatic Teller Machine -).

Internet, SMS - Short Message Service -, telephone). This isn't only due the fact of political issues, but as well as some security issues - privacy, confidentiality and voting authentication - inherent in such systems.

This indecision has contributed to the practical application of the NST to EV, being important to develop a system that can move between these two different voting types in the future, with minimal changes in its structure.

5 Solution Proposal

In this chapter we propose a model that has some characteristics from the NST. To do so, a mapping between the business process related to the EV and the elements of the proposed model is done. The idea is to implement the mapping done, allowing us to evaluate if the NST grant agility to IS.

5.1 Normalized Systems Theory Model

As we can see in the figure [4] we have the 5 proposed elements plus 3 types of tasks, and their representation is based on the definition proposed by the NST:

- **Workflow Element**: represents a sequence of Action Elements, Connectors and Triggers, that are called in order to maintain the state on the Data Element on which they operate - to grant separation of concerns;
- **Data Element**: represents a set of attributes or data fields including connections to other Data Elements that can do support tasks. This element should be encapsulated to ensure the Data Version Transparency theorem;
- **Action Element**: represents only one Functional task to ensure the separation of concerns theorem, and can contain multiple support tasks. This element should be encapsulated to ensure the Version of actions transparency theorem.
- **Trigger Element**: represents the activation of an Action Element or Workflow in a periodic base, based on time;
- **Connector Element**: consists on an IO task;
- **IO Task**: responsible for the man-machine and machine-machine IO;
- **Support Task**: performs a transversal concern in IS;
- **Functional Task**: responsible to ensure IS functionality.

The main objective of the proposed model is to allow the implementation of evolutive IS, trying to accomplish two of the three characteristics that composes the NST (41), because it is organized according with the specified five high-level elements and supports the four theorems (chapter 4).

However, it cannot represent the final feature of the theory - modularity - and the same will be fulfilled in the following sections.

5.2 Business Process mapped in Business flows

In this point the mapping is accomplished, as explained in the chapter 42, for two of the four business process from the traditional voting (paper based). It consists on the mapping between
two processes: the voting (which implies three sub-processes: voter identification, voter validation on electoral notebook, filling in a vote) and the counting of votes; on the elements of the proposed model - figure 9.

Each of the traditional processes described, gave rise to new business processes, each with its own specific flow - referent to a single Data Element with a life cycle. This mapping is described on the table 9.

<table>
<thead>
<tr>
<th>Traditional business process</th>
<th>Business Process of System to develop</th>
<th>Business Flow</th>
<th>Data Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voter identification</td>
<td>User registration on the system</td>
<td>ClientFlow</td>
<td>ClientElement</td>
</tr>
<tr>
<td></td>
<td>User authentication on the system</td>
<td>AuthenticationFlow</td>
<td>LoginElement</td>
</tr>
<tr>
<td>Voter validation</td>
<td>Voter registration in the electoral notebook</td>
<td>ElectorFlow</td>
<td>ElectorElement</td>
</tr>
<tr>
<td></td>
<td>Voter validation on the electoral notebook</td>
<td>ValidationFlow</td>
<td>ValidationElement</td>
</tr>
<tr>
<td>Filling in a vote</td>
<td>Create a ballot</td>
<td>BallotFlow</td>
<td>BallotElement</td>
</tr>
<tr>
<td></td>
<td>Realization of the vote by a voter</td>
<td>VoteFlow</td>
<td>VoteElement</td>
</tr>
<tr>
<td>Vote counting</td>
<td>Process every vote received</td>
<td>ProcessVoteFlow</td>
<td>VoteElement</td>
</tr>
<tr>
<td></td>
<td>Calculate the election results</td>
<td>ResultFlow</td>
<td>ResultElement</td>
</tr>
<tr>
<td></td>
<td>Start/stop the ballot box</td>
<td>BallotBoxFlow</td>
<td>BallotBoxElement</td>
</tr>
</tbody>
</table>

Table 1. Mapping between the business process and the business flows.

It is important to refer that the VoteElement entity of the ProcessVoteFlow flow is not the same of the VoteFlow flow, being independent entities.
5.3 System Design

One of the key concepts of the theory is modularity, referred on chapter \(^1\) that facilitates information hiding and intended to separate the different system concerns.

Thereby, based on the mapping of table \(^1\) there were identified the following modules:

- **Authentication Server**: encompasses the processes of system user registration and user authentication;
- **Electoral Notebook Server**: includes elector registration and validation in electoral notebook;
- **Vote Server**: enables ballot creation and the act of voting;
- **Ballot Box Server**: encompasses received vote processing from Vote Server, calculation of election results and start / stop ballot box.

6 Validation

After EVS implementation based on the solution proposed in the last chapter, it’s time to evaluate NST to know if, in fact, it grants IS maintenance a more efficient and effective way to solve our problem.

It was necessary a similar EVS, to validate the NST. The other prototype was developed by André Bricoso, student of Professor Carlos Ribeiro, with the purpose of ensuring the voting verifiability \(^1\).

We applied to both systems some direct metrics, through execution of six tests to compare the results obtained.

Within the metrics identified in Chapter \(^2\) the only ones used were those that best fit the three basic characteristics, which intend to evaluate software maintenance facility, respectively:

- **Time to make a change**: analysis and conclusion times for a given task;
- **Amount of code added**: number of code lines, number of created methods and number of created classes;
- **Number of dependencies found with changes**: couplings added between modules, number of errors and cyclomatic complexity.

The performed tests were based on some anticipated changes, that can usually occur in a system:

- **Add a new attribute**: add client’s date of birth;
- **Add a new Data Element**: add Envelope entity which joins vote and voter;
- **Add a new Connector Element**: add a new system Interface - voting confirmation;
- **Add a new Action Element**: send Envelope entity to Ballot Box;
- **Change Functional Task**: Google credentials system authentication;
- **Add a new Workflow Element**: construction of an entity Envelope.

6.1 Results

Relatively to obtained times, it is visible that time to make changes was considerably lower on the system based on the NST. This happened because the system was more organized, with modules
division of concerns and with few dependencies between them. Excessive dependencies between modules was the reason why the auxiliary system received high time for testing.

Couplings complicate the execution of maintenance tasks since they are synonyms of dependencies and thus cause "combinatorial" effects on the system. The fact that the auxiliary system (used in test comparisons) had a larger number of couplings in exactly the same tests where the greatest number of errors and the highest cyclomatic complexity were detected; this may imply that with a greater number of possible paths, more likely it will be to have dependencies in the system. Consequently, the greater the likelihood of errors to arise in the maintenance phase - which implies greater effort to make changes in IS.

Our normalized system implied the implementation of more classes and methods to perform a given action, in comparison to a non-normalized system. Nevertheless, it was possible to verify through the times obtained that the maintenance tasks were performed faster.

6.2 General Discussion of Results

The EVS developed was used in the AEIST elections, validating the hypothesis H1 (3), because the model developed (figure 11 and modules described (chapter 53) - which represent the three characteristics of NST (chapter 53) - allowed to develop an IS.

It appears that the greater the number of dependencies is, the greater the effort required to make changes in IS and consequently, the greater the time taken to execute. Thus, we conclude that the NST is useful in avoiding dependencies - combinatorial effects - and therefore facilitating a rapid evolution of IS.

If the auxiliary system was greater (larger scale), we can assume that the system could have a greater number paths, and possibly more couplings - based on the Law of Lehman - and obtained errors. This would imply a greater effort and therefore more time to perform the tests. Given the functionality separation in the developed system, the time and effort to perform maintenance tasks it would be practically the same, whatever their scale - stability concept defended by NST.

It is thus validated the hypothesis H2 (4), since it was possible with reduced time and effort to change/add functionality in a normalized IS. On the other hand, non-normalized systems may constitute a major impediment to changes as well as complexity increase.

7 Conclusions

This chapter aims to present the main conclusions of this work and its research contributions.

7.1 Main Conclusions

The greater the number of dependencies greater the effort required to make a change in IS and the greater the time taken to execute it.

Based on the results obtained in chapter 13 we can say that if an IS is organized around five high-level elements, supports the four theorems and is based on the concept of modularity, for the separation of concerns14, then it will be able to support a set of anticipated changes. This situation allows the IS to evolve, in other words, the ability to respond quickly and effectively to change and increasing complexity - making the IS agile and stable.

3 These three factors contribute to the weak dependency between modules and strong relationship between items of the same module (weak coupling and strong cohesion.)
Therefore, the initial objective was met and the answer to the problem of this dissertation was found:

- *Concerning the maintenance phase, the application of Normalized Systems theory (to IS development) allows changes in the system and an increase on its complexity, combined with reduced time and effort!*

We should take NST in consideration to create and operate software, in order to mitigate the effect of the Lehman's law. Otherwise, organizations will likely to have create their IS from scratch again, having in mind that it can be tricky to make changes in systems with too many dependencies.

### 7.2 Research Contributions

The main contribution of this work, which is important for the scientific community, focuses on the approval of a method that avoids maintenance problems in organizations - NST. The perception of the theory in an organization can change the way that we think and build an IS.

By the solution presented and the tests developed we can extract the remaining contributions:

- Construction of a base model for the implementation of any IS, based on elements and theorems defined in the NST - figure 1. The application of this model and the concept of modularity in the development of IS, allows organizations to obtain several advantages, including:
  - Ensuring the evolutive capacities of the companies;
  - Enabling performing tasks related to the software development and project management.
- Development of an EVS based in model from the NST.

### 8 Future Work

It should be noted that this work meets all the requirements as originally proposed. However, due to the perception of the implementation and evaluation that was made, there are several aspects that can be improved or even added. These aspects will be addressed in this section. Some are related to the prototype and others with the NST.

#### 8.1 EVS Prototype

**Interface:** One of the aspects considered as a negative point in the prototype developed and used in elections to the direction of AEIST, was the existence of a very simple interface. Thus, one possible way would be to improve the appearance of the considered interface.

**Security:** A major problem that leads to poor adhesion of countries to EVS is security weaknesses in the act of voting. The scope of work does not consider security aspects, however, this area should be studied and included in the system.

**Transaction:** The concept of transaction should be used in the system to ensure that in case of failure, the system does not achieve an inconsistent state. Notice that this inconsistency refers to data elements with incorrect states.
Validation: It was important to make a validation with more than one EVS, in order to obtain more accurate results. This was necessary because the scope of the used system was security - verifiability of the vote -, and not the evaluation of the system’s agility.

Nevertheless, it was also important to apply the theory in other areas (not related to EV), to obtain more concrete results on the effectiveness and efficiency of the NST.

8.2 Normalized Systems Theory

Mapping Business Processes in the Elements of the NST: The representation of the mapping of business processes in business flows is not formalized, being suggested in its formalization.

Normalization of Information Systems Architectures: This work proved that the application of the theory in the development of an IS is important to minimize the number of dependencies and to achieve stability. In order to minimize them, even more, it would be interesting to extend the core concepts of the NST to the upper layers of Enterprise Architecture.

For example, using the extension mechanism of the modeling language for enterprise architecture - ArchiMate, it would be possible to apply the theory’s elements and theorems in order to ensure that a change of a given business process does not have repercussions associated with the size of the architecture, in the future, remaining constant as it grows.

Comparisons between the NST and others Theories: It would be interesting to compare the NST with other theories related to the agility and maintenance of IS in order to find the best solution to resolve the maintenance problems described in chapter 2.

References