A proposal for changes in the current system of Suframa collection for an object-oriented Platform Using the DDD Project patterns and SOLID.

PERES, Paulo Júnior de Jesus [1]

PINTO, Aurélio Guimarães [2]

FREITAS, Caio Guimarães [3]

LEITE, Francisco Canindé da Silva [4]

SILVA, Francisco Eronildo da [5]

OLIVEIRA, Geveson de Souza [6]

RIBEIRO, Dallas dos Santos [7]

ALMEIDA, Cristiany Caliri de [8]

MORAIS, Gilvanete Melo de [9]

PERES, Paulo Júnior de Jesus; et.al. A proposal for changes in the current system of Suframa
collection for an object-oriented Platform Using the DDD Project patterns and SOLID. Multidisciplinary Core scientific journal of knowledge. 07 Edition. 02 year, vol. 03. pp 36-43, October 2017. ISSN: 0959-2448

Summary

The collection of a Federal Agency as Suframa is extremely complex, as there are several legal and regulatory integration necessary for a smooth operation. Therefore, it is justified to use a robust system that meets the business rules and necessary developments in this dynamic process. For this reason, this article seeks to offer a viable proposition from the point of view of software architecture for developing a solution with quality code using the DDD projects and SOLID patterns.

1. Introduction

The current software that manages the collection held by the Suframa was developed in mid-1990 with the platform the Cobol programming language and the SQL database DBB, which is an earlier version than DB2 from IBM. According to the Coordination Authority computer, such a system operates on low platform, i.e. on a Mainframe IBM Z22.

Before the software in operation at the institution, we can highlight the following problems:

- Non-relational database and data storage limitation;
- Structured programming language;
- Lack of versioning of the codes;
- Lack of integrated tests;
- Difficulty of labor to perform system maintenance.

As is well known, a software of this dimension is a "live" in need of constant developments. In this light, it has been found by computer Coordination of Suframa that such software does not meet the business needs more neither the technological needs, between the needs of the main developments are as follows:

- Integration with web platforms through Web services;
- Integration with platforms and mobiles
- Processing, generation and export of reports in various formats.

According to the electronic site of the institution, the Superintendency of the Manaus free Zone (Suframa) – is a local authority of the Federal Government under the Ministry of industry, Trade and services that aims to build a model of regional development use natural resources in a sustainable manner ensuring economic viability and improving the quality of life of the population. Currently the Suframa has like scope the States of Amazonas, Acre, Rondônia, Roraima and Amapá.
On the importance of the municipality for the region, and the large volume of resources collected, justifies the development of a new system for collecting used modern software architecture patterns and technologies. Therefore, the course of this article will be proposed a strategy for development of the software mentioned in this article.

2. Software architecture patterns

Before we define which patterns will be using, it is important to define what is a software design pattern, thus, according to Gamma (2000) design standard are generic solutions to the most common and recurring problems in software development Object-oriented (OOP) by a team of software development.

The main intention of the project is to reduce the risk of errors in software through the use of techniques widely tested and approved by software development community.

Second Gamma (2001), currently there are several design patterns, and a lot of information available for research, among these patterns we can highlight:

- Dependency Injection or Dependency Injection;
- Factory Method;
- Bulder;
- Abstracty Factory;
- Singleton;
- Prototype;
- Command;
- Iterator;
- Strategy;
- Visitor among others;

Amid the multitude of existing standards, this article seeks to highlight the suggested patterns as a basis for the development of the new system of collection of the Superintendency of the Manaus free trade zone which are: Domain-Driven Design (DDD) and SOLID, because the use of both imply in the use of various Software development practices.

In this way, does not mean that the course of the project may not be used other standards, just by teaching issues will be addressed these two patterns in this article. Thus, in the following chapters we will address deeply such standards.

2.1. The Default Domain-Driven Design (DDD)

According to Evans (2016) Domain-driven development (DDD) is a disciplined approach to software
design that addresses a series of concepts and techniques always focusing in the field of software.

In this pattern, the focus of the development is in the area of the system, i.e. the rules of the software business.

A great advantage of the DDD is that your approach is fully object-oriented, i.e. developers must build the software using object-oriented programming, in this way, second Cukier (2010) the benefits achieved are the following:

- **Code alignment with the business**: the involvement of developers with connoisseurs/domain experts is paramount when DDD, this is a feature of agile development;
- **Encourage reuse**: the codes developed, can be reused in a transparent manner.
- **Minimal coupling**: after modeling the data model well developed, organized, the "layers" of the software can communicate without dependency through the use of interfaces;
- **Technology independence**: DDD does not focus on the technology used in the development of the software, but in business rules.

However, your goal is to reach the area code necessary for the software to be developed using an architecture that allows the application of the concepts of DDD, so second Cukier (2010) the pattern suggests the use of four main layers are:

![Figure 1-proposed Architecture for the DDD. Source: Cukier (2010)](image)

In the figure above, the following are clarifications second Cukier (2010):

- **User interface** – is the layer responsible for direct interaction with the user, as the software screens, Web Services SOA or REST, mobile applications or desktops, among others;
- **Application**-works directly with the domain layer in order to isolate the area of layer "User
“Interface”, it is important for the papers stay distinct, for example: there's no need to know which interface the domain class must perform certain action. In this way, the application layer is a kind of mediator between the domain and the interface;

- **Domain** – is the layer that contains all the business rules of the application, i.e. everything that reflects functional requirements, the calculations are handled in this layer;
- **Infrastructure** – responsible for the interaction with the technical infrastructure, for example, data access, access to file systems, email submissions, among others.

These layers are not mandatory, depending on the need of the project can be removed or added new layers. What should be kept in mind is the need for imperative programming geared to the domain.

But there is something to be noted, according to Pires (2016) DDD programming is not a layered architecture nor a prescription ready for use in any software to be developed. The DDD focuses on domain-oriented development, this is the focus and not the layers.

Before such concepts, we reach the main benefits earned by using the DDD that second Cukier (2010) is that the induction to the development of software within the scenario of continuous improvement, making it an extremely useful tool to develop software for quality and that meets the needs of the client.

So, apply the standard of projects means rescue truly programming DDD object-oriented, because your implementation the development team is required to apply other standards such as: Repository, Decorator, Strategy, State among others.

### 2.2. SOLID

According to Martin (2008), Robert c. Martin identified five principles of object-oriented programming that seek the improvement of codes, such principles are SRP, OCP, LSP, ISP and DIP. Before these principles, Michael Feathers noted that could be created the acronym called SOLID containing the five principles. The following image demonstrates the meaning of the acronym:
Follows the definition of each part of the acronym according to the publication of Martin (2008):

- **Single responsibility Principle or single responsibility principle**: a class should have one, and only one reason to change. This principle a class must have only a single purpose, for example: when creating a calculator should not make a class math operations and put all her operations, but rather, make a class for each operation;

- **Open Closed Principle or Open-Closed Principle**: one should be always open to extension and closed for implementation, that's why, when you make changes to an existing code runs the risk of an error impact on several other code fragments;

- **Liskov Substitution Principle or Liskov substitution principle**: bases classes should always be replaceable by its foundations, that is, a class B Inherits class A january 1 can always be overridden by class in A code implementation;

- **Interface Segregation Principle or principle of segregation of Interfaces**: many specific interfaces are better than one single interface, as this practice aims to reduce the coupling of codes;

- **Dependency Inversion Principle or dependency inversion principle**: one should always depend on an abstraction and never of an implementation.

When applying the SOLID, you get the following benefits to your project: facility to perform maintenance and project developments, ease of automated testing, less effort to code development and maximum reuse of code.

### 3. Proposed new collection System

The purpose of this article is to provide the knowledge base for the development of new software for management of the collection held by the Superintendency of the Manaus free trade zone-Suframa using the paradigm of object-oriented programming by applying the standards of DDD and SOLID projects in developing the project.

In this proposal, following a plan of action based on 5W1H administration model, we would have the following planning for further development:
Table 01—action plan for development of the new system.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Define programming language and</td>
<td>Standardize software development.</td>
<td>Technical meetings</td>
<td>On coordination of</td>
<td>ICT managers, programmers and</td>
<td>First week</td>
</tr>
<tr>
<td>technologies for the project</td>
<td></td>
<td></td>
<td>Informatics</td>
<td>software architects</td>
<td></td>
</tr>
<tr>
<td>Preparing the development</td>
<td>Create mechanisms for</td>
<td>Installation of servers,</td>
<td>On the workstation of</td>
<td>Configuration and support team of ICT</td>
<td>Second</td>
</tr>
<tr>
<td>environment and approval</td>
<td>the team to work on the project</td>
<td>software and frameworks</td>
<td>the members of the</td>
<td></td>
<td>week</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>development team and</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>on the servers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software requirements gathering</td>
<td>Document the information to be</td>
<td>Meetings and interviews</td>
<td>Suframa</td>
<td>Requirements analysts</td>
<td>The week 10week 3.</td>
</tr>
<tr>
<td>Encoding</td>
<td>programmed in the field</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approval and corrections</td>
<td>Create the software</td>
<td>Coding and testing</td>
<td>On Suframa</td>
<td>Software Factory</td>
<td>Of the week 22</td>
</tr>
<tr>
<td></td>
<td>for what has been developed is</td>
<td></td>
<td></td>
<td></td>
<td>week 11.</td>
</tr>
<tr>
<td></td>
<td>really what the Suframa needs</td>
<td></td>
<td></td>
<td></td>
<td>Of the week 23</td>
</tr>
<tr>
<td></td>
<td>and perform corrections</td>
<td></td>
<td></td>
<td></td>
<td>to 25.</td>
</tr>
<tr>
<td>Deployment</td>
<td>Make the system</td>
<td>Doing the build</td>
<td>Hosting systems</td>
<td>Configuration analyst</td>
<td>Week 26.</td>
</tr>
<tr>
<td></td>
<td>in the production in the production</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Drawn by the author.

As the table above, it appears that the project will take around six and a half months to complete, however it should be noted that this plan of action contains macros, actions would be needed other plans to detail every action in your macro execution.

In view of the action of "coding" pointed on 01 table, the use of DDD, this proposal recommends be created the following layers:

- Presentation layer—this layer will be the user's vision, and for web 2.0 Angular framework will be used and to mobile devices (Android/IOS) using the Ionic framework;
- Layer of services – will be responsible for providing the presentation layer all webservice in REST format and will have the role of control using the dependency injection framework *ninject*;
Application layer – responsible for application implementations, as log record and domain encapsulation;
Domain layer-contains the implementation of business rules, it will be the entities, Interfaces, and domain services;
Infrastructure tier – responsible for the data access layer using an ORM tool, by repositories, by sending e-mails and authentication.

In all the layers should be applied the principles of SOLID.

**Conclusion**

This technical article sought to simply draw up a concrete proposal of migration from a legacy system in low platform development for a modern platform using as a base the standards of Software development projects oriented to development Domain (DDD) and SOLID.

It is understood that this proposal serves as the basis not only for the collection of the Superintendency of the Manaus free zone, but also as the basis for other projects of migration and evolution of software running in open for further adjustments and future re-issues by other authors.

**References**


Available at: <http: cieam.com.br/?u="suframa-tenta-retomar-cobranca-da-taxa-de-servicos-administrativos_-suspensa-pelo-stf">accessed on 21 April 2017.</http:>

Available at: <http: www.eduardopires.net.br/2012/06/ddd-ddd-bdd/>"accessed on 21 April 2017.</http:>

Available at: <http: www.eduardopires.net.br/2016/08/ddd-nao-e-arquitetura-em-camadas/>"accessed on 21 April 2017.</http:>
Available at: <http: www.agileandart.com/2010/07/16/ddd-introducao-a-domain-driven-design/="">accessed on 21 April 2017.</http:>

[1] Graduated in computer science, it acts as a public server on SUFRAMA, as Administrative Analyst-you. Database specialist for ULBRA.

[2] Graduated in computer science, it acts as a public server on SUFRAMA, as Administrative Analyst-you.

[3] Graduated in computer science, it acts as a public server on SUFRAMA, as Administrative Analyst-you.

[4] Graduated in computer science, it acts as a public server on SUFRAMA, as Administrative Analyst-you.

[5] Graduated in computer science, it acts as a public server on SUFRAMA, as Administrative Analyst-you.


[7] Graduated in computer science, acts as server SUFRAMA, as Administrative Analyst-you.

[8] Graduated in business administration, acts as public servant on SUFRAMA, as administrator.

[9] Graduated in economics, acts as public servant on SUFRAMA, as an economist.