SOFTWARE EROSION ON A STRUCTURAL ANALYSIS SOFTWARE - A CASE STUDY

Flavio Torres da Fonseca [1], Roque Luiz da Silva Pitangueira [2], Samuel Silva Penna [3]

[1] Universidade Federal de Minas Gerais, flaviotf@dees.ufmg.br
[2] Universidade Federal de Minas Gerais, roque@dees.ufmg.br
[3] Universidade Federal de Minas Gerais, spenna@dees.ufmg.br

Abstract

The INSANE (Interactive Structural Analysis Software) is being developed by the Structural Engineering Department of Federal University of Minas Gerais (DEES/UFMG) since 2002. Through these years, the system has evolved from a simple 2D frame element analysis software to a very complex system, with many different methods, elements, constitutive models and solution types. However, the system also suffered from a well known phenomena called software erosion: a slow deterioration of software performance over time. In this paper, the INSANE system will be briefly described and a diagnosis of its actual state will be presented. Software erosion causes, consequences and possible solutions or preventions will be discussed.

1. A brief history of the INSANE project

The INSANE (Interactive Structural Analysis Software) software started to be developed in 2002, by the Structural Engineering Department of Federal University of Minas Gerais (DEES/UFMG), as an undergraduate research program. Its initial objective was to create a simple 2D frame element analysis software to be used as an academic resource in Structural Analysis courses [1, 2]. However, since the beginning, its software architecture and class organization were planned to be as general as possible, in order to enable its expansion.

Since then, many undergraduate and graduate researchers have worked on it, resulting in 29 Master Thesis and 5 Doctoral Dissertations [3].

Today, the INSANE is a complex system with many methods (Finite Element Method - FEM, Generalized Element Method - GFEM, Boundary Element Method - BEM and Mesh Free Methods), linear and non linear (physically and geometrically) analysis, static and dynamic analysis, as well as many different constitutive models [4, 5, 6, 7, 8, 9].

Despite many efforts have been put in creating a friendly and complete user interface [10], its development did not follow the advances of the analysis modulus, so the last stable version of INSANE was released in 2004 and presents only the frame elements interface.

2. Software erosion on the INSANE source code

Software erosion can be described as a slow deterioration of software performance over time. It is a phenomena that has been studied by computer scientists for decades and has been detected not only in academic softwares, but also in well known open sources and proprietary projects, as Mozilla web browser, VIM text editor, ANT and version 2.4 of Linux kernel [11, 12, 13].

The biggest problem with software erosion is that its effects accumulate over time, making the source code harder to maintain and to understand.

One of the main causes of this phenomena in the INSANE comes from the project's own nature: an academic project, in which each developer has an individual objective and a rigid time schedule. The rotativity of contributors is very high, so the understanding of the system as a whole and the formation of a developing culture is not achieved. Another point is the lack of sense of community. Some people tend to think only in its own work and do not worry about the effects of the modifications on other developers' work.

In the INSANE source code the most commons signs of software erosion are cyclic dependencies, dead code (unused code, large number of commented lines) and code clones (identical or near-identical code fragments).

A sign of erosion which appeared after the adoption of Git as the version control system is the huge number of branches that were created and that are never updated with the master branch modifications or nerver merged into master. There are also some orphan branches which were created, modified and abandoned.

Another problem in INSANE code, but that is not direct related to software erosion but indirectly contributes for it, is the small quantity of implemented unit tests. To maintain a stable code and enable refactoring, it is imperative to have tests, to ensure that the software works properly.

3. Recommendations to avoid and reverse software erosion

The first step to avoid the software erosion process is to put into the mind of all developers that the INSANE system is bigger than their individual work and that other people will need to understand and use their source code. By knowing this, it is expected that all contributors will create good 'legacy' code for the upcoming developers.

Another step is to introduce code review process, which can be done within the adopted version control system (Git and GitLab). This will force the code to be well written and dead code to be removed.

To reverse the software erosion, the existing source code must be refactored. This refactorization comprises dead code removal, proper documentation of source code, numerical methods optimization and software architecture modification. For this last change, one of the possibilities is to adopt a plug-in based architecture, which will modularize and encapsulate even more the source code, protecting the core classes and methods.

In order to guarantee that the code is correct and functional, unit tests must be implemented.

4. Conclusion

The INSANE source code has suffered a heavy process of software erosion and a lot of work must be done in order to make it useful, understandable and expandable. A complete refactoring and modification of its software architecture, adopting a plug-in based architecture, is the best option and will be done in the next few years.

5. Acknowledgments

The authors would like to acknowledge CNPq (National Council of Scientific and Technological Development), CAPES (Coordination of Improvement of Higher Education Personnel), FAPEMIG (Minas Gerais State Research Foundation) and PROPEEs/UFMG (Structural Engineering Graduate Program of the Federal University of Minas Gerais) for financial supports.

References

[1] F.T. Fonseca, R.L.S. Pitangueira. *Um programa gráfico interativo para modelos estruturais de barras*, XXV CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering, 2004.

[2] R.L.S. Pitangueira, K. Caldas. *Projeto de software livre para modelos do Método dos Elementos Finitos*, XXVI CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering, 2005.

[3] INSANE project website. `http://www.insane.dees.ufmg.br` [Accessed: 31/05/2018].

[4] F.T. Fonseca, R.L.S. Pitangueira. *Insane: uma plataforma para computação científica*, X Encontro de Modelagem Computacional, 2007.

[5] P.D. Alves, F.B. Barros, R.L.S. Pitangueira. *An object-oriented approach to the Generalized Finite Element Method*, Advances in Engineering Software, 1-18, 2013.

[6] A.B. Monteiro, A.R.V. Wolenski, F.B. Barros, R.L.S. Pitangueira, S.S. Penna. *A computational framework for G/XFEM material nonlinear analysis*, Advances in Engineering Software, 380-393, 2017.

[7] R.G. Peixoto, F.E.S. Anacleto, G.O. Ribeiro, R.L.S. Pitangueira, S.S. Penna. *A solution strategy for nonlinear implicit BEM formulation using a unified constitutive modelling framework*, Engineering Analysis with Boundary Elements, v. 64, p. 295-310, 2016.

[8] Silva, R.P. *Análise Não-Linear de Estruturas de Concreto por meio do Método Element Free Galerkin*, Doctoral dissertation, Federal University of de Minas Gerais, Belo Horizonte, Brazil, 2012.

[9] L. Gori, S.S. Penna, R.L.S. Pitangueira. *A computational framework for constitutive modelling*, Computers & Structures, 1-23, 2017.

[10] S.S. Penna, R.L.S. Pitangueira. *Projeto orientado a objetos de um pós-processador para modelos do Método dos Elementos Finitos*, XXVII CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering, 2006.

[11] M. Dalgarno. *When good architecture goes bad*, Methods & Tools, Editor, 27-34 , 2009.

[12] L. Silva, D. Balasubramaniam. *Controlling software architecture erosion: A survey*, The Journal of Systems and Software, 132-151, 2012.

[13] R. Terra, M.T. Valente, K. Czarnecki, R.S. Bigonha. *Recommending refactorings to reverse software architecture erosion*, 16th European Conference on Software Maintenance and Reengineering, IEEE, 335-340, 2012.