Information Hiding and Modula

June 26, 2013.

José Francisco García Juliá

jfgj1@hotmail.es

The information hiding principle can be applied completely using the Modula language.

Key words: Information hiding, Modula.

In 1971, David Parnas published his first paper in information hiding (IH) [1]. The paper did explain that it was information distribution that made systems "dirty" by establishing almost invisible connections between supposedly independent modules [2]. In 1972, he published a seminal paper on the decomposition of a system into modules [3] (where a module is a work assignment for an individual or a group), based in his principle of information hiding: the basic idea is to hide information that is likely to change. One needs to structure the system so that all arbitrary or changeable or very detailed information is hidden. The "solid" or lasting information is specified as interfaces that communicate the modules.

In applications with many modules, these are organized into a hierarchy. Each of the top-level modules would have a clearly defined secret, and would be divided into smaller modules using the same principle. This results in a tree structure. The (maintenance) programmer can find the relevant modules by selecting a module at each level and moving on to a more detailed decomposition. The structure is depicted in a document called the module guide [4]. IH is the only principle that guarantees the construction of successful, cost reduced, maintainable and reusable (software) systems.

On the other hand, in 1976, Niklaus Wirth published the first paper in Modula, a language for modular programming [5]. In this report, a Modula program is formulated as a module which, in its turn, is defined as a collection of procedures. In its heading, a module contains two lists of identifiers: the define-list mentions all module objects (constants, types, variables, procedures) that are to be accessible (visible) outside the module. The use-list mentions all objects declared outside the module that are to be visible inside. This facility provides an effective means to make available selectively those objects that represent an intended abstraction, and to hide those objects that are considered as details of implementation. A module encapsulates those parts that are non-essential to the remainder of a program, or that are even to be protected from inadvertent access. Modules may be nested. Modula has additional facilities for multiprogramming: processes, interface modules and signals.

The define and use lists of the Modula programs of Wirth are the interfaces of Parnas that communicate the modules.

Although Modula was intended primarily for programming dedicated computer systems, including process control systems on smaller machines, adding a facility for processing large databases it can be converted into a general purpose language. Then,

the entire module guide of Parnas could be compiled as a Modula program of Wirth, instead of using the specification technique of Parnas [6]. When the number of modules is very large, they may be compiled in groups following the hierarchy (tree structure), and executed from a control program, in a main memory (virtual) partition, previously batched, where the job control language (JCL) statements contains the library addresses of the corresponding modules.

Many programmers think that all the programs written in object oriented (O-O) languages fulfill with the IH principle, and that a class is like a module of IH or Modula. This is not true necessarily. Note that the Simula language, which was the first object oriented language, was published in 1966 [7], six years earlier that the IH principle was established [3]. Most of the objects (class instances) are only the reflection of a functional decomposition (use cases) of the system under study, and not the reflection of the IH principle.

In summary, the IH principle can be applied completely using Modula. IH and Modula (and its successors) fit perfectly.

[1] Parnas, D.L., "Information Distributions Aspects of Design Methodology", Proceedings of IFIP Congress 71, 1971, Booklet TA-3, pp. 26-30.

[2] Parnas, D.L., "The Secret History of Information Hiding", pp. 399-409, M. Broy, E. Denert (Eds.): Software Pioneers, Springer-Verlag, Berlin, Heidelberg 2002.

[3] Parnas, D.L., "On the Criteria to be Used in Decomposing Systems into Modules", Communications of the ACM, vol. 15, no. 12, pp. 1053-1058, 1972.

[4] Parnas, D.L., Clements, P.C., Weiss, D.M., "The Modular Structure of Complex Systems", Proceedings of 7th International Conference on Software Engineering, March 1984, 408-417.

[5] Wirth, N., "Modula: A language for modular multiprogramming", Institut für Informatik, ETH Zürich, 1976.

[6] Parnas, D.L., "A Technique for Software Module Specification with Examples", Communications of the ACM, vol. 15, no. 5, pp. 330-336, 1972.

[7] Dahl, Ol.-J., Nygaard, K., "SIMULA - An Algol-based Simulation Language", Communications of the ACM, vol. 9, no. 9, pp. 671-678, 1966.