Thesis Overview:

Scientific Software: Legacy Software Maintenance

Mariano Méndez
School of Computer Science, National University of La Plata, Argentina
PhD Thesis in Computer Science[1], 21/4/2016
Advisor: Fernando G. Tinetti
{mmendez,fernando}@lidi.info.unlp.edu.ar

Software production and maintenance is one of the most widely studied topics in computer science. Taking into account that scientists were the first ones to perform this task, even before the Computer Science discipline obtained its name, scientific software production still remains a challenge. This phenomenon may stem from the fact that some authors realized that there is a "gap" between scientific production techniques and industry software techniques [1]. This research work is founded on two relevant contributions to this subject. The first one, the Aristotelian analysis performed by Frederick Brooks on software essence that has been beautifully described in [2]. The second one was proposed by Ralph Johnson in his article "***Software development is program transformation***." Based on these two great contributions, Change- Driven Development is proposed as a new agile methodology born as a new approach to maintain and develop Scientific Software. This process is based on the principles of software essence - Changeability, Complexity, Intangibility and Conformity- as well as on integrated development tools, and automated source code transformations implemented as refactorings. This new, agile approach takes change as a working unit devised to drive the entire development process, which is performed in a four-stage cycle. One of the most interesting approaches to apply Change Driven Development on scientific software is to update, modernize and even parallelize sequential programs that were written 20 or 30 years ago and are still running in production environments. This vision towards software construction is considered to be an approach driven by change as they run counter to the ones utilized by classic methodologies which are driven by a plan. An important aspect in the light of this vision to be borne in mind is that change should be viewed as an essential software feature. Along the same lines, the project development process is centered on two of the four essential features of software [2]. To this end, this process should be guided by changes or transformations to be applied on software. New software development born out of already existent one is more widespread than the one started from nothing [3]. In fact, building libraries and components that can not be re-utilized would not make sense.

A Change-Driven Development Process could be thought of as an Agile Methodology when it comes to maintenance and scientific software development-related purposes. There is a high likelihood that this process may be extended to any kind of software. It is characterized by being **Focused on Change**: Given that change is one of the essential properties of software [2] CDD possesses Change as a minimal unit of work. One change is supposed to belong to one of these four characterizations: Corrective Change, Adaptive Change, Perfective Change, and Preventive Change. One change will be, as a result, the unit of work in which the five Software Development Core Activities will be performed: Requirement, Analysis, Design, Implementation, Testing and Deployment. Each one of these activities will have a certain percentage of fulfillment within a given type of change, See Figure 1.

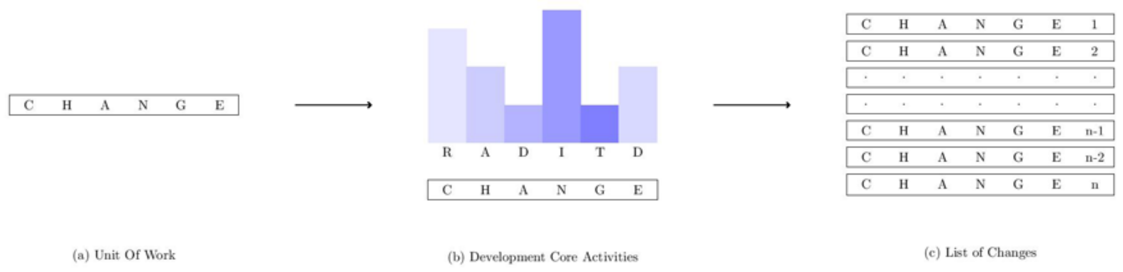

Figure 1

It is **Tool Centered**: In this development process, tools are considered as important as change itself. It is almost inconceivable to believe that nowadays software can be constructed without making use of some kind of development tool. From this perspective, tools are conceived of as a means to apply the process itself. Programmers and other members of development teams must rely on such tools for the purposes of identification, comprehension, transformation and software verification. Finally, given that this is an **Iterative and Incremental Process;** broadly speaking, the spirit of this technique resides in the construction of a problem solving skeleton and in the fulfillment of small work cycles (called iterations). Out of performing these tasks, the solution to the problem will arise out of the application of the process in its entirety [4].

Taking into account the features that a change-driven development process must have, a first description of such process will be thoroughly analyzed. Let us contemplate in this approach that a change is considered to be any variation in the state of any artifact that is involved in the software development process. A change or transformation as a concept or vision is applicable to more artifacts other than source code. This process can be performed either from an already existent list of changes or a special iteration can be produced to obtain such list. As a following step, starting from a change to be applied to the system, this process comprises four stages. These stages are: Comprehension, Transformation, Verification, and Feedback. Once these four stages have been covered, a change will be applied into software, see Figure 2. This cycle could be applied indefinitely as required. Within the four stages of the change-driven development software process, a specific workflow has been defined when working with scientific software, the defined workflow comprises the following steps: the establishment of an initial version of source code, the source code transformation, the verification of the obtained source code, the numerical result validation, the Acceptance/Rejection of the applied change based of numerical results, and finally the documentation. A set of case of studies has been presented; the first example proposed in the thesis can be seen on Table 1.

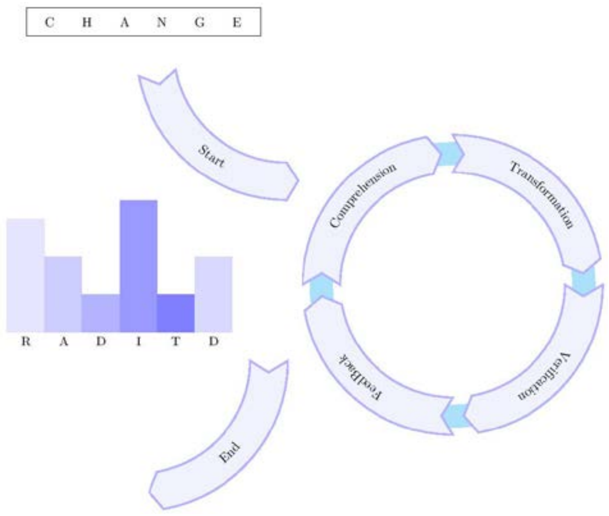

Figure 2

FORTRAN 77	Fortran 90 obtained by application of CDD
PROGRAM FIRST	program FIRST
DATA N/25/, H/1.0/, X/0.5/	implicit none
F = SIN(X)	real:: d,e,f,g,h,q,x
G = COS(X)	integer :: i,n
DO 2 I = 1,N	parameter (n = 25)
H = 0.25*H	x=0.5
D = SIN(X + H) - F	h=1.0
Q = D/H	f = SIN(x)
E = ABS(G - Q)	g = COS(x)
PRINT *,H,D,Q,E	do i = 1,n
2 CONTINUE	h = 0.25*h
STOP	d = SIN(x + h) - f
END	q = d/h
	e = ABS(g - q)
	print *,h,d,q,e
	end do
	end program FIRST

Table 1

Four source code examples have been thoroughly explained by using Change-Driven Development in this research work. Furthermore this research study will even venture into parallelizing an originally sequentially written Fortran program by applying such process and source code transformations.

References

[1] Segal, Judith. "Scientists and software engineers: A tale of two cultures." (2008).

[2] Brooks, F. *No silver bullet*. April, 1987.

[3] Johnson, Ralph E. "Software development is program transformation."*Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 2010.

[4] Basil, Victor R., and Albert J. Turner. "Iterative enhancement: A practical technique for software development." *IEEE Transactions on Software Engineering* 4 (1975): 390-396.