Unleashing the Potential Impact of Nonessential Self-contained Software Units and Flexible Precedence Relations upon the Value of Software

Antonio Juarez Alencar, Rafael Alcemar do Nascimento, Eber Assis Schmitz,
Alexandre Luis Correa, and Angélica F. S. Dias
Email: juarezalencar@nce.ufrj.br, rafael.alcemar@gmail.com, eber@nce.ufrj.br,
alexcorr@yahoo.com, angelica@nce.ufrj.br

Abstract— **This paper evaluates the impact of nonessential minimum marketable features modules (NMMF) and nonessential architectural elements (NAEs) on software projects; shows that the value-creation path of these self-contained software units may be quite different from that of essential software units; and discusses the impact of early NMMF and NAE identification on the value of software projects to business and the deployment of business strategy. Moreover, the paper demonstrates that the existence of flexible precedence relations among MMFs and AEs may also be exploited to further increase the value of software development initiatives.**

Index Terms— **Value-based software engineering, nonessential software units, minimum marketable feature modules, economics of software engineering.**

I. INTRODUCTION

Despite the vital role that IT can play in the business landscape, funding software development projects in the highly competitive environment in which companies do business these days has become a central issue for both managers and practitioners alike. In many markets stakeholders and investors are claiming not only for better financial returns, but also for shorter investment periods, faster time to market, less risk and improved ability to adapt to new market conditions quickly and efficiently [1], [2]. As a consequence, it is becoming increasingly difficult to get software development projects funded, unless they provide clearly defined value to business [3].

Consistent with this view many attempts have been made to bring financial discipline to software development [4]. While some proposals are broadly based on project financial valuation metrics such as net present value, return on investment and, more recently, real options theory [5], [6], others take a more holistic view of software development and advocate the use of value-based metrics [7]–[9].

Nevertheless, all these attempts fall short of acknowledging that requirement prioritization and modularization play a crucial role in the business value of software. While the requirements satisfied by a module are paramount to determine its value, the order in which these modules are implemented dictates how soon this value can be appropriated.

One notable exception is presented by Dennne and Cleland-Huang [3] who suggest the use of thorough financial analysis to maximize the value of software projects composed of minimum marketable feature modules (MMFs), i.e. modules containing small sets of features that have value to business. In their work Dennne and Cleland-Huang show that the implementation order of MMFs may change quite substantially the value of such projects. The ideas of Dennne and Cleland-Huang were later extended by Alencar *et al.* [10], who use a *branch & bound* technique to overcome some of the limitations imposed by the method proposed by the former [11].

Nonetheless, both Denne and Cleland-Huang, and Alencar *et al.* failed to recognize that (a) it is not always the case that all modules are essential to the development of a software project; (b) if a module is nonessential to a software project, its development may or may not be rightfully pursued by the project manager during the project life cycle; (c) the value of nonessential modules may vary according to the set of modules that have preceded its development and (d) when implemented, instead of creating their own cash flow stream, nonessential modules may contribute to the value of a software just by positively influencing the value of essential modules.

This paper is a step towards filling this gap by uncovering the value of nonessential minimum marketable features modules (NMMFs) and nonessential architectural elements (NAEs), whose value-creation path may be quite different from those of essential self-contained software units. Moreover, this paper shows how the early identification of NMMFs and NAEs during the project life cycle may affect the final value of software projets, benefit software development as a whole and help to shape business strategy. In addition, this paper demonstrates how the combination of flexible precedence relations and nonessential self-contained software units may be used to increase the value of software even further.

The remainder of this paper is organized as follows. Section II presents a review of the principal concepts and methods used in this paper. Section III introduces a real-world inspired example that helps in understanding the role played by NMMFs, NAEs and flexible precedence relations in software development. Section IV presents the conclusions of this paper.

II. Minimum Marketable Features

According to Denne and Cleland-Huang [3] MMFs are self-contained software units that create value to business in one or several of the following areas: (a) competitive differentiation (b) revenue generation (c) cost savings (d) brand projection and (e) enhanced customer loyalty. Although an MMF is a self-contained unit, it is often the case that it can only be developed after other project parts have been completed. These project parts may be either other MMFs or the architectural infrastructure, i.e. the set of basic features that offers no direct value to customers, but that are required by the MMFs.

The architectural infrastructure itself can usually be divided into self-contained deliverable elements. These elements, called *architectural elements*, or AEs for short, enable the architecture to be delivered according to demand, further reducing the initial investment needed to run a project. Moreover, the total value brought to a business by a software consisting of several interdependent MMFs and AEs, each one with its own cash flow stream and precedence restrictions, is highly dependent on the development order of these units.

For instance, Figure 1 presents the precedence diagram of self-contained software units comprising a loan control system. The meaning of each software unit is described in Table I. In this particular example $SU_1, \cdots, SU_5$ are MMFs and SU_6 is an AE. The reasons why these particular software units are MMFs and AEs is an object of discussion in Section III.

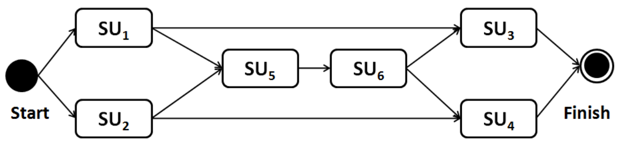

Figure 1. Loan control software precedence diagram.

In the diagram an arrow going from one MMF or AE to another, e.g. $SU_5 \rightarrow SU_6$, indicates that the development of the former (SU_5) must precede the development of the latter (SU_6).

Table II indicates all possible development sequences for the loan control software, considering that (a) it takes exactly one period to develop each software unit; (b) only one software unit can be developed per period; (c) the first software unit is developed in period one and (d) there is no delay between the completion of a software unit and the beginning of the development of the next.

Table III shows the undiscounted cash flow elements of each MMF and AE in the model introduced in Figure 1. For example, according to the information presented in Table III, SU_1 requires an initial investment of US \$50 thousand. Once its development is completed at the end of the first period, it provides a series of positive returns until the fifteenth period, when the salary-loan software becomes obsolete and is replaced by a new and more advanced tool. Hubbard [12] shows how the cash flow elements of software projects may be properly estimated.

Because it is improper to perform arithmetic operations on monetary values without taking into account an interest rate, in order to compare the business value of different MMFs and the investment required by AEs one has to resort to their discounted cash-flow [13]. Table IV shows the sum of the discounted cash-flow of each software unit in Figure 1, considering a discount rate of 2% per period. Such a sum is the net present value (NPV) of all cash flow elements of the software unit. In order to make understanding easier, the figures presented in Table IV have been rounded to the nearest integer value. The remaining figures presented in this paper follow the same convention.

For instance, according to the information presented in Table IV, if SU_1 is developed in the first period, it yields a NPV of US \$1,045 thousand i.e.

$$\frac{-50}{(1+2\%)^1} + \frac{50}{(1+2\%)^2} + \frac{70}{(1+2\%)^3} + \cdots + \frac{50}{(1+2\%)^{15}}$$

On the other hand, if SU_1 is developed in the second period, it yields a NPV of \$987 thousand, in the third

TABLE I.
Description of the software units comprising the loan control software

Software Unit		
Id	**Name**	**Description**
SU_1	Apply for a loan	Collects the necessary data to grant a loan to a customer
SU_2	Apply for refinancing	Collects the data needed to grant the refinancing of an existing loan
SU_3	Accept loan conditions	Allows customer to accept or decline the loan conditions proposed by the lender
SU_4	Accept refinancing conditions	Allows customer to accept or decline the refinancing conditions proposed by the lender
SU_5	Quick credit analysis	Figures the likelihood of a customer paying back a loan in accordance with certain installment values and dates
SU_6	Check for funding availability	Checks whether the lender has the necessary funds to grant a loan to a given customer

TABLE II.
Scheduling options

Scheduling Options	Period					
	1	2	3	4	5	6
1	SU_1	SU_2	SU_5	SU_6	SU_3	SU_4
2	SU_1	SU_2	SU_5	SU_6	SU_4	SU_3
3	SU_2	SU_1	SU_5	SU_6	SU_3	SU_4
4	SU_2	SU_1	SU_5	SU_6	SU_4	SU_3

TABLE III.
Software-unit cash-flow elements

Cash Flow Elements (US \$1,000)					
Id	**Period**				
	1	**2**	**3**	**4 to 14**	**15**
SU_1	-50	50	70	100	50
SU_2	-70	20	28	40	20
SU_3	-30	500	700	1,500	1,000
SU_4	-40	200	280	600	200
SU_5	-80	90	120	180	90
SU_6	-10	0	0	0	0

TABLE IV.
SOFTWARE-UNIT NET-PRESENT VALUE

Net Present Value (US $1,000)						
Id	**Period**					
	1	**2**	**3**	**4**	**5**	**6**
SU_1	1,045	987	894	802	712	623
SU_2	368	346	309	274	239	204
SU_3	16,001	14,944	13,537	12,157	10,804	9,478
SU_4	6,221	5,950	5,388	4,836	4,296	3,766
SU_5	1,885	1,781	1,613	1,447	1,285	1,126
SU_6	-10	-10	-10	-9	-9	-9

$894 thousand and so on.

Obviously, not all software units can be developed in the first period. The precedence diagram presented in Figure 1 indicates that only SU_1 and SU_2 can be developed at that time. Because in this example each software unit requires exactly one period to be developed, SU_5 cannot be developed until the third period. Furthermore, each particular sequence of software units yield its own NPV. For instance, the sequence

$$SU_1 \rightarrow SU_2 \rightarrow SU_5 \rightarrow SU_6 \rightarrow SU_3 \rightarrow SU_4$$

yields $17,564 thousand, which is the highest NPV among all possible development sequences.

III. AN EXAMPLE

According to Robert Hall (1764-1831), the British clergyman: "The innocence of the intention abates nothing of the mischief of the example". As a result, the discussion presented in this paper is introduced step-by-step with the help of a real-world inspired example regarding the development of a mobile software application[1].

Step 1: Context information

"Salary loans" are low-risk low-interest multi-purpose loans granted to qualified employees of accredited companies, in which payment is made on installments via salary deduction [14]. In this respect consider an international financial institution such as CITIBANK, BARCLAYS, HSBC, ABN AMRO, UBS and many others that make salary loans available to their customers. For the purpose of this paper this organization is named LOANS "R" US, or L_RU.

As soon as an application for a salary loan is received, L_RU figures the likelihood of the applying customer paying back the requested loan in accordance with acceptable installment values and dates. Next, L_RU makes sure that it has enough funds to grant the loan that the customer is asking for (as by law financial institutions cannot lend money above a certain limit, so as to preserve its financial health). Finally, the applying customer is presented with the conditions under which a salary loan may be granted, if any. At this point, they have the option of accepting or declining the loan offer.

[1]One of the authors of this paper has successfully managed a project that is very similar to the one described in this section for a major financial institution in South America, which kindly asked not to have its name disclosed.

Financial institutions that lend money also tend to offer refinancing services for existing loan agreements as a way of relieving the financial pressure on both current and future customers. L_RU is no exception to the rule. As the refinancing operation may involve a loan agreement issued by a third party, this operation is frequently referred to as "debt purchase".

In order to provide an adequate competitive response to recent competition moves into its salary-loan market and further develop its loan business, L_RU has decided to build a new salary-loan mobile web-based software. The company believes that if it acts fast, this software may not only improve its turnover considerably, but also favorably reshape the competitive landscape of the salary-loan business. Figure 1 introduces a model containing the software units that comprise the new salary-loan software.

Step 2: Determining how each software unit may generate revenue

Table I describes the meaning of each software unit introduced in Figure 1. One should note that the first five software units listed in that table generate revenue for L_RU in the following manner:

- *SU_1 and SU_2* - every client applying for a salary-loan is asked whether they agree to receive new product offers from both L_RU and its associated companies from time to time. Willing customers may then be targeted by new marketing campaigns, generating sales opportunities for L_RU and revenue in the form of fees due to the use of L_RU's customer database by the associated companies;
- *SU_3 and SU_4* - once a client accepts a loan or refinancing offer, they generate revenue in the form of due interest and
- *SU_5* - the results of the "Quick credit analysis" software unit are used to enrich the L_RU's customer database. Credit analysis information is particularly important to marketing campaigns that depend upon the financial heath of willing customers, especially those campaigns that allow customers to pay for a product or service on installments. L_RU's associated companies are required to pay a premium fee to access the enriched database.

Therefore SU_1, SU_2, $\cdots$, SU_5 are all minimum marketable features modules, or MMFs. On the other hand, SU_6, "Check for funding availability", also describes a self-contained software unit, which provides an essential service to the intended behavior of the software, but neither L_RU's customers nor associated companies are willing to pay for the service it provides. Therefore, SU_6 describes an architectural element (AE).

Step 3: Planning the software implementation

Although the project team assembled by L_RU to run the salary-loan software project has initially considered adopting the software units described in Table I together with the precedence diagram presented in Figure 1, they soon realized that the behavior of SU_5 does not actually depend on whether the salary loan under consideration is the result of a new application or a request for refinancing. Therefore, the development of SU_5 may start when the development of either SU_1 or SU_2 is completed, or even when the development of both of them are completed. Note that this new perspective on the dependencies required by SU_5 is actually a real option, and options may quite substantially change the value of software projects [15]. Hence the L_RU's project team decided to amend the precedence diagram presented in Figure 1. Figure 2 shows the updated precedence diagram for the salary-loan software project.

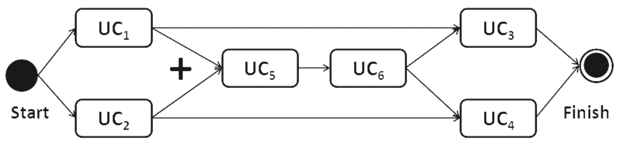

Figure 2. Salary-loan project precedence diagram.

One should note that SU_5's flexible precedence requirement is properly signaled by the presence of the "+" symbol (inclusive or) in the diagram. It is also important to note that the modules presented in Figure 2 comprise the set of modules that are essential to the development of the salary-loan software, as (according to current markets conditions) this is the smallest set of self-contained modules that bears a fair chance of providing long-term lawful return on the investment to be made by L_RU in their development [16]. Table V indicates all possible development sequences for the salary-loan software. There are fourteen possibilities altogether. It is important to mention that as modules SU_1, SU_2, $\cdots$, SU_6 are all essential to the intended behavior of the software, therefore all of them have to be developed eventually.

TABLE V.
SCHEDULING OPTIONS

Scheduling Options	Period					
	1	2	3	4	5	6
1	SU_1	SU_2	SU_5	SU_6	SU_3	SU_4
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
7	SU_1	SU_5	SU_6	SU_3	SU_2	SU_4
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
14	SU_2	SU_5	SU_6	SU_4	SU_1	SU_3

Step 4: Evaluating the different scheduling options

Table VI shows the undiscounted cash flow elements of each MMF and AE in the model introduced in Figure 2 as estimated by L_RU's project team.

Because SU_5 generates revenue according to the number of customers in the L_RU's customer database (see Step 3), its cash flow elements vary according to the context in which SU_5 is developed. The greater the number of customers in the database, the higher the revenue generated by SU_5. Therefore

- If SU_1 precedes the development of SU_5, which is indicated by $SU_1 \rightarrow SU_5$ in Table VI, then SU_5 yields US \$60 thousand in the second period, US \$90 thousand in the third period, US \$130 in the fourth period and so on;
- If it is SU_2 that precedes the development of SU_5, i.e. $SU_2 \rightarrow SU_5$, then SU_5 yields smaller values each period. It yields US \$ 20 thousand in the second period, US \$ 30 thousand in the third period, US \$ 50 in the fourth period and so on;
- However, if both SU_1 and SU_2 precede the development of SU_5, i.e. $(SU_1,SU_2) \rightarrow SU_5$, then the value yielded by SU_5 in each period reaches its highest values. SU_5 yields US \$ 90 thousand in the second period, US \$ 120 thousand in the third period, US \$ 180 in the forth period and so on.

TABLE VI.
SOFTWARE-UNIT UNDISCOUNTED CASH-FLOW ELEMENTS

Cash Flow Elements (US \$ 1,000)					
Id	Period				
	1	2	3	4 to 14	15
SU_1	-50	50	70	100	50
SU_2	-70	20	28	40	20
SU_3	-30	500	700	1,500	1,000
SU_4	-40	200	280	600	200
$SU_1 \rightarrow SU_5$	-80	60	90	130	60
$SU_2 \rightarrow SU_5$	-80	20	30	50	20
$(SU_1,SU_2) \rightarrow SU_5$	-80	90	120	180	90
SU_6	-10	0	0	0	0

In all circumstances the investment required by SU_5 remains the same, i.e. US \$ 80 thousand. Table VII shows the net present value (NPV) of each MMF and AE presented in Figure 2 according to the period in which their development starts, considering an interest rate of 2% per period and that, due to shortage of funds, only one development team has been allocated to work in the project.

TABLE VII.
SOFTWARE-UNIT NET PRESENT VALUE

Net Present Value (US \$ 1,000)				
Id	Period			
	1	2	$\cdots$	6
SU_1	1,045	987	$\cdots$	623
SU_2	368	346	$\cdots$	204
SU_3	16,001	14,944	$\cdots$	9,478
SU_4	6,221	5,950	$\cdots$	3,766
$SU_1 \rightarrow SU_5$	1,334	1,263	$\cdots$	792
$SU_2 \rightarrow SU_5$	454	430	$\cdots$	253
$(SU_1,SU_2) \rightarrow SU_5$	1,885	1,781	$\cdots$	1,126
SU_6	-10	-10	$\cdots$	-9

Table VIII relates the scheduling options introduced in Table V to their respective NPV. For example, in the seventh option SU_1 is developed first. Next, SU_5, SU_6, SU_3, SU_2 and SU_4 are developed in the second, third, forth, fifth and sixth periods respectively, yielding an NPV of \$18,460 thousand, which the highest NPV among all scheduling options and, as a result, the logical choice for the L_RU's project team.

Obviously this NPV can only be reached because L_RU is a well established company that runs its loan business with the support of trustworthy business processes, which can be temporarily used to support the development of the salary-loan software. For example, when the development of SU_1 or SU_2 is completed, applying customers do not need to wait for the development of the remaining modules to be informed of the conditions in which they may be granted a loan. At this point L_RU's loan processes take over and provide the adequate answer. This holds true for any combination of previously developed modules.

Step 5: Dealing with the nonessential software units

Common sense and experience has shown that there are two major ways of loosing money in the loan business: lending money to those who will not pay back their loans and failing to lend money to those who will [17]. Therefore, before committing to a particular scheduling option based upon a model that only considers essential software units, the L_RU's project team decided to examine the effect of nonessential software units that deal with these two different aspects of the loan business on the returns yielded by the salary-loan software. Table IX describes the nonessential software units conceived by the project team.

There is only one nonessential minimum-marketable-feature module (NMMF) among the nonessential software units, i.e, SU_7, "Thorough credit analysis", which further enriches the L_RU's customer database with information that opens opportunities for new marketing campaigns from both L_RU's and its associated companies.

As the revenue generated by SU_7 depends upon the number of trustworthy customers it is able to identify, the more customers who have their credit extensively analyzed, the better. Hence the revenue generated by SU_7 is influenced by the development of both SU_1,"Apply for a loan", and SU_2, "Apply for refinancing", in the same way

TABLE VIII.
SCHEDULING OPTION NPV

Scheduling Option	**NPV (US $ 1,000)**
1	17,564
2	16,769
⋮	⋮
7	18,460
⋮	⋮
14	15,814

TABLE IX.
NONESSENTIAL SOFTWARE UNIT DESCRIPTIONS

Software Units		
Id	**Name**	**Description**
SU_7	Thorough credit analysis	Performs a more detailed credit analysis on applications that have been turned down by the "Quick credit analysis" software unit, with the intention of granting a loan to applying customers, if this turns out to be possible
SU_8	Contract life insurance	With the intention of granting loans, include the cost of life insurance in loan applications turned down by the "Quick credit analysis" software unit due to the short life expectance of loan applicants
SU_9	Handle approved loans	Determines the earliest possible date on which L_RU can grant a loan or a refinancing request that has been turned down by the "Check for funding availability" due to temporary lack of funds

that SU_5, "Quick credit analysis", is. Table X presents the cash flow elements of SU_7. The revenue generated by this particular software unit stems from the premium fee paid by L_RU's associated companies to access the enrichment customer database.

Surprisingly, this is not the only way in which SU_7 contributes to the value of the salary-loan project. If implemented, "Thorough credit analysis" increases the number of approved loans and refinancing applications. The more loans and refinancing applications are approved, the higher the returns yielded by "Accept loan conditions" and "Accept refinancing conditions". Therefore, "Thorough credit analysis" also helps to increase the revenue generated by these two MMFs.

TABLE X.
SU_7 CASH-FLOW ELEMENTS

Cash Flow Elements (US $ 1,000)					
Id	**Period**				
	1	**2**	**3**	**4 to 9**	**10**
$SU_1 \rightarrow SU_7$	-90	65	75	90	65
$SU_2 \rightarrow SU_7$	-90	25	30	35	25
$(SU_1,SU_2) \rightarrow SU_7$	-90	90	100	130	90

On the other hand, "Handle approved loans" and "Contract life insurance" are self-contained units whose services neither customers nor associated companies are willing to pay for. Therefore, they are indeed nonessential architectural elements, or NAEs for short. Even though these two software units are unable to generate revenue on their own, they do contribute to the value of the salary-loan software by influencing the revenue generated by other MMFs.

In a similar fashion to "Thorough credit analysis", both "Handle approved loans" and "Contract life insurance" increase the number of approved loans and refinancing applications. As a result, they further improve the revenue generated by both "Accept loan conditions" and "Accept refinancing conditions". Table XI presents the estimated impact of the development of nonessential software units on the revenue generated by essential software units.

TABLE XI.
THE ESTIMATED FINANCIAL IMPACT OF NONESSENTIAL SOFTWARE UNITS.

		Essential Software Units	
		SU_3 (Acpt. loan cond.)	**SU_4** (Acpt. refinancing cond.)
Non-essential Software Units	**SU_7** (Thorough credit analysis)	+20%	+15%
	SU_8 (Contract life insurance)	+15%	+10%
	SU_9 (Handle approved loans)	+25%	+20%

For example, according to the information displayed in Table XI, from the moment "Thorough credit analysis" is implemented the revenue generated by "Accept loan conditions" and "Accept refinancing conditions" increase 20% and 15% respectively.

Step 6: Replanning the software implementation

Figure 3 introduces the new precedence diagram for the salary-loan software. This diagram takes into consideration the existence of both essential and nonessential software units. A gray background with a dashed-line frame around a software unit identification tag such as SU_7 nor SU_8 indicates a nonessential software unit, which may or may not be developed.

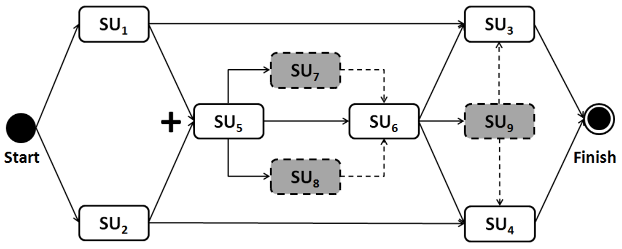

Figure 3. A precedence diagram containing both essential and nonessential software units.

However, if a nonessential software unit is developed, than the dependencies that it creates must be complied with. For example, if neither SU_7 nor SU_8 are developed, than the development of SU_6 can start right after the development of SU_5 is completed. Nevertheless, if SU_7 is developed and SU_8 is not, then the development of SU_6 can only start when the development of both SU_5 and SU_7 are completed, and so on. Table XII presents the scheduling options for the salary-loan software considering the existence of both essential and nonessential software units. There are 144 different scheduling options altogether.

TABLE XII.
NEW SCHEDULING OPTIONS

#	Period								
	1	2	3	4	5	6	7	8	9
1	SU_1	SU_2	SU_5	SU_6	SU_3	SU_4			
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
81	SU_2	SU_1	SU_5	SU_7	SU_8	SU_6	SU_3	SU_4	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
144	SU_2	SU_5	SU_7	SU_9	SU_6	SU_1	SU_8	SU_4	SU_3

Step 7: Which scheduling option does provide the highest NPV?

Table XIII shows the undiscounted cash flow elements of each MMF, AE, NMMF and NAE introduced in Figure 3, as estimated by L_RU's project team.

Table XIV shows the net present value (NPV) of each MMF, AE, NMMF and NAE presented in Figure 3 according to the period in which their development starts, considering an interest rate of 2% per period. It should be noted that the number of periods that takes to develop the whole software has potentially increased from six to nine periods, because of the possible introduction of nonessential MMFs and AEs (check Table XII).

TABLE XIII.
NEW SOFTWARE-UNIT CASH-FLOW ELEMENTS

New Cash Flow-Elements (US $ 1,000)					
Software Unit Id	**Period**				
	1	**2**	**3**	**4 to 14**	**15**
SU_1	-50	50	70	100	50
SU_2	-70	20	28	40	20
SU_3	-30	500	700	1,500	1,000
$SU_7 \rightarrow SU_3$	-30	600	840	1,800	1,200
$SU_8 \rightarrow SU_3$	-30	625	875	1,875	1,250
$SU_9 \rightarrow SU_3$	-30	575	805	1,725	1,150
$(SU_7,SU_8) \rightarrow SU_3$	-30	725	1,015	2,175	1,450
⋮	⋮	⋮	⋮	⋮	⋮
SU_9	-10	0	0	0	0

TABLE XIV.
ESSENTIAL AND NONESSENTIAL SOFTWARE UNIT NPV

Net Present Value (US $ 1,000)				
Software Unit Id	**Period**			
	1	**2**	···	**15**
SU_1	1,045	987	···	369
SU_2	368	346	···	105
SU_3	16,001	14,944	···	5,653
$SU_7 \rightarrow SU_3$	19,207	17,939	···	6,788
$SU_8 \rightarrow SU_3$	20,009	18,688	···	7,072
$SU_9 \rightarrow SU_3$	18,406	17,190	···	6,504
$(SU_7,SU_8) \rightarrow SU_3$	23,215	21,683	···	8,208
⋮	⋮	⋮	⋮	⋮
SU_9	-10	-10	⋮	-9

Among all the scheduling options introduced in Table XII, the most logical option for L_RU is to develop the software units comprising the salary-loan software in the following order:

$$SU_2 \rightarrow SU_1 \rightarrow SU_5 \rightarrow SU_7 \rightarrow SU_8 \rightarrow SU_6 \rightarrow SU_3 \rightarrow SU_4,$$

which yields an NPV of US $ 19,651. One should note that this scheduling option does not contemplate the development of SU_9, "Handle approved loan". According to the estimates generated by L_RU's project team, the cost of developing SU_9 is not compensated by its positive impact on the revenue generated by SU_3 and SU_4.

IV. CONCLUSIONS

Table XV compares the value of the different alternatives considered by L_RU's project team for the development of the salary-loan software.

Observe that, by and large, what nonessential self-contained software units and flexible precedence relations do is to allow project managers to postpone the development of less valuable software units until this becomes imperative. As a result, the value of the more profitable developments paths may be pursued and appropriated earlier.

For example, in the salary-loan software project the precedence relations of "Quick credit analysis" is flexible in the sense that the development of this software unit may start when the development of either "Apply for a loan" or "Apply for refinancing" is completed, or even when the development of both are completed. This flexibility alone increases the value of the salary-loan software from US $17,564 to US $18,460 thousand at no extra cost. Subsequently, with the introduction of more flexibility in the form of the nonessential MMFs and AEs the project value jumps to US $19,651 thousand.

All of this supports the claim that the introduction of flexible precedence relations as well as the presence of NMMFs and NAEs among essential software units is likely to increase the business value of software projects.

Identifying nonessential software units is not as hard as it may seem at first sight. If one has specific goals to be reached by a software project that are clearly stated, non essential software units are fundamentally those that can be taken away from the project without hindering the likelihood of it fulfilling its stated goal [18], [19].

However, despite the considerable body of research that is available in the literature on the impact of flexibility upon the valuation, selection and management of software projects [5], [15], [20], the introduction of flexibility via nonessential self-contained software units and flexible precedence relations has remained largely unexplored.

This paper goes forward in filling this gap by helping organizations to get the most out of their software projects, and, as a result, to increase their level of efficiency in the use of information technology. By being conscious of the potential value brought by nonessential self-contained software units and flexible dependency relations, companies can develop larger and more complex projects from a relatively smaller investment.

References

[1] W. R. Priya Kurien and V. S. Purushottam, "The case for re-examining IT effectiveness," *Journal of Business Strategy*, vol. 25, no. 2, pp. 29–36, 2004.

[2] M. A. Cusumano, *The Business of Software*. Free Press, 2004.

[3] M. Denne and J. Cleland-Huang, "Financially informed requirements prioritization," in *Proceedings of the 27th International Conference on Software Engineering (ICSE)*. St. Louis, MO, USA: ACM Press, 15-21 May, 2005, pp. 710–711.

[4] W. V. Grembergen, *Information Technology Evaluation Methods and Management*. Idea Group Publishing, 2001.

[5] M. Benaroch, S. Shah, and M. Jeffery, "On the valuation of multistage information technology investments embedding nested real options," *Journal of Management Information Systems*, vol. 23, no. 1, pp. 239–261, Summer 2006.

TABLE XV.
ALTERNATIVES CONSIDERED BY THE DEVELOPMENT TEAM OF THE SALARY-LOAN SOFTWARE

#	Preced. Diag.	Flexibility	Logical Choice NPV ($1,000)
1	Fig. 1	None	$17,564
2	Fig. 2	The precedence dependency relation between SU_1, SU_2 and SU_5 is acknowledged to be flexible	$18,460
3	Fig. 3	The precedence dependency relation between SU_1, SU_2 and SU_5 is acknowledged to be flexible, and NMMFs and NAEs are introduced	$19,651

[6] S. Dekleva, "Justifying investments in IT," *Journal of Information Technology Management*, vol. XVI, no. 3, pp. 1–8, 2005.

[7] R. J. Benson, T. Bugnitz, and B. Walton, *From Business Strategy to IT Action: Right Decisions for a Better Bottom Line*. Wiley, 2004.

[8] K. Milis and R. Mercken, "The use of the balanced scorecard for the evaluation of information and communication technology projects," *International Journal of Project Management*, vol. 22, no. 2, pp. 87–97, February 2004.

[9] K. M. Rosacker and D. L. Olson, "An empirical assessment of IT project selection and evaluation methods in state government," *PM Journal*, vol. 29, no. 1, pp. 49–58, February 2008.

[10] A. J. Alencar, E. A. Schmitz, and E. P. de Abreu, "Maximizing the business value of software projects," in *10th Inter. Conf. on Enterprise Info. Systems.*, vol. ISAS-2. Barcelona, Spain: INSTICC, June 12-16, 2008, pp. 162–169.

[11] M. Denne and J. Cleland-Huang, "The incremental funding method: Data-driven software development," *Software, IEEE*, vol. 21, no. 3, pp. 39– 47, 2004.

[12] D. W. Hubbard, *How to Measure Anything*. Wiley, 2007.

[13] A. A. Groppelli and E. Nikbakht, *Finance*, 5th ed. Barron's, 2006.

[14] C. L. Peterson, "Usury law, payday loans, and statutory sleight of hand," *Minnesota Law Review*, vol. 92, no. 4, April 2008.

[15] R. G. Fichman, M. Keil, and A. Tiwana, "Beyond valuation: Real options thinking in IT project management," *California Management Review*, vol. 47, no. 2, pp. 74–100, 2005.

[16] C. Hibbs, S. Jewett, and M. Sullivan, *The Art of Lean Software Development: A Practical and Incremental Approach*. O'Reilly Media, 2009.

[17] N. Siddiqi, *Credit Risk Scorecards*. Wiley, October 2005.

[18] K. Pohl, G. Bckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. NY, USA: Springer, November 2010.

[19] P. Valente, *Goals Software Construction Process*. VDM, 2009.

[20] L.-C. Wu and C.-S. Ong, "Management of information technology investment: A framework based on a real options and mean–variance theory perspective," *Technovation*, vol. 28, no. 3, pp. 122–134, 2008.

Antonio J. Alencar is a researcher with the Federal University of Rio de Janeiro (UFRJ), Brazil. He received his B.Sc. in Mathematics and M.Sc. in System Engineering and Computer Science from UFRJ. He holds a D.Phil. in Computer Science from Oxford University, England. His research interests include economics of software engineering, IT strategy and risk analysis.

Rafael A. Nascimento is an M.Sc. student with the Federal University of Rio de Janeiro, Brazil. His research interests include software development methodologies and economics of software engineering.

Eber A. Schmitz is a Professor of Computer Science with the Federal University of Rio de Janeiro. He holds a B.Sc. in Electrical Engineering from the Federal University of Rio Grande do Sul, an M.Sc. in Electrical Engineering from the Federal University of Rio de Janeiro and a Ph.D. in Computer Science and Control from the Imperial College of Science, Technology and Medicice, England. His research interests include software modeling tools, business process modeling and stochastic modeling.

Alexandre L. Correa is a Professor of Computer Science with the Federal University of the State of Rio de Janeiro. He holds a B.Sc. in Mathematics and an M.Sc. and a Ph.D. in System Engineering and Computer Science from the Federal University of Rio de Janeiro. His research interests include reverse engineering, system validation and software development tools.

Angélica F. S. Dias is Ph.D. candidate with the Federal University of Rio de Janeiro (UFRJ). She holds a M.Sc. in Computer Science from UFRJ. Her research interests include human-computer interaction and the social impact of new technologies.