Bridging EE and SE: The Goals Approach

Pedro Valente, David Aveiro, Nuno Nunes

1 Software Applications Development Office, University of Madeira, Madeira, Portugal
2 Madeira Interactive Technologies Institute, University of Madeira, Madeira, Portugal
{pvalente, daveiro, nijn} @ uma.pt

Abstract. Organizations still struggle to maintain and develop their software patrimony, the tool that usually makes the difference in the organization’s activity among its market competitors towards survival and consequent desired contribution to the society, something which nowadays is particularly important for those which activities’ heavily rely on information. The information systems that support the tasks that it’s collaborators need to carry out in order to attain the desired products to satisfy its customer’s needs, will usually and preferably be Interactive Information Systems (IISs) that provide high performance tools by means of sophisticated Interaction Spaces (IS) that properly raise the meaning of tasks (efficiency) towards its completeness (effectiveness) and improve the overall efficiency and effectiveness of the organizational production process, and therefore, raise the overall organizational welfare. Goals is an approach that aims at bridging the managing Enterprise Engineering (EE) and the supporting Software Engineering (SE) domains of an organization by means of a conceptual structure that allows an increased awareness of each Software Implementation Decision (SID) implication over the IIS’s architecture (Interaction Spaces (ISs), Business Rules (BRs) and Data Entities (DEs)), to enable the study of related Software Development Effort (SDE) following Business Process Improvements (BPIs) (i.e. Business Process (BP) and/or User Tasks (UT) improvements) by focusing on obtaining increased Return of the Investment (ROI) in each iteration, as a way to attain the objective of producing a reliable Software Development Effort Estimation (SDEE) method that fits the industry. Ultimately, we believe that a change of paradigm in the way that IISs are managed and understood within organizations can result in more feasible and adequate software projects according to the organization’s capabilities, to eventually raise project success (from a precarious 30%) [1] or at least avoid financially catastrophic situations [2].

Keywords: Enterprise Engineering, Software Engineering, Human-Computer Interaction, Business Process Improvement, Software Process Improvement, Software Metrics, Financial Metrics, Modeling.
1 Introduction

Business Process (BP) Improvement (BPI) is a major issue in both Enterprise Engineering (EE) and Software Engineering (SE) domains [3, 4]. The enhancement of a BP is usually a top management initiative that involves changing the sequence of User Tasks (UTs) and/or how these tasks are carried out (in order to reach an goal) and aims at improving the BP’s efficiency and/or effectiveness and therefore the organization’s overall productive capability. This improvement will only be possible if a software project is carried out in order to produce the desired changes, and within this specific Software Development Process (SDP) topic, efficiency and effectiveness are also crucial for changes to occur within the desired lapse of time and at an acceptable price. A BPI needs to meet an adequate Interactive Information System (IIS) support in terms of usability, as if not, distinct problems will probably arise and negatively affect the overall desired augmented performance as usually not sufficient attention is paid by enterprise and/or software engineers [5] to those whom actually use the systems in order to produce the desired products, whether these are experiences, services, goods or commodities [6].

Attaining the desired level of Software Development Process (SDP) performance will only be possible if relevant variables (e.g. usability and requirements stability) are well understood and controlled by those who cooperate to achieve it, i.e. enterprise and software engineers, so that software development instability can be minimized and consequently its productivity maximized, as software development is a challenging or even a persnickety task that characterizes itself by having to deal with a dependency structure (an architecture) in which components rely on the correct functioning of other components to support the proper functioning of the whole IIS. If on one hand, an enterprise engineer may have the necessary deep knowledge on the organization’s current state and the existing desires to improve (to pursue e.g. relevant market advantages), on the other hand, a software engineer may have the deep knowledge on the IIS foundations (e.g. how data is organized, which business rules and user interfaces are easy to change or not, and why), something that can reveal itself as crucial, especially if the organization heavily relies on information accuracy to properly operate.

Sustaining the cooperation between both these professionals is the main objective of our present work in order to pursue the final objective of improving SDP performance, which we believe will only be attained if both EE and SE experts properly cooperate in a functional hierarchy (as requirements are passed from the first to the second), and that this cooperation only will be fluent if the relevant shared components (meaning, way of functioning and importance) are understood by both parts, in a SDP that is also aware of the necessary effort to produce an organizational change. This situation can be compared to what happens in the civil industry where the architect will pass to engineers the requirements of construction, for them to find a solution that will enable the construction of the desired building in a feasible way.

The Goals approach, which we propose as solution to improve software development performance within organizations, relies on the idea that:

*If an organization is aware of its own capabilities to implement an enactment towards its goals, then it will be able to do it in the most “profitable” way for itself.*
In other terms, if an organization is aware of the Software Development Effort (SDE) according to each possible Software Implementation Decision (SID) to enhance a specific BP, and knows how to implement it in most proper way according to the existing resources (whether they are human, information or physical, including monetary and time resources) then it will be able to enhance the Return of the Investment (ROI) that is achieved when a BPI is carried out, as it will be capable to prospect the benefits, and also be aware the costs of this operation.

The fact that only about 40% of projects are completely successful falls on how challenging is the objective to improve SDP performance, as SE has been trying to achieve this objective for about 4 decades following the initial “software crisis” phase in the 70’s [7], highlighting that this problem is still far from being solved. Therefore, we believe that the grounding solution for this problem is to understand organizations as a whole, including its supporting information system, since the system “operates” within the enterprise, and that this can only be achieved if both EE and SE domains can understand each other concerning shared components that define both the enterprise and its supporting information system. Our proposal for an organizational conceptual structure relies on the following five shared components:

- Business Process [BP];
- User Task [UT];
- Interaction Space [IS];
- Business Rule [BR], and;
- Data Entity [DE].

Briefly, the five components relate to each other in the following way.

BPs are a sequence of UTs that are carried out by users in ISs that allow them to manage information according to existing BRs and recurring to existing DEs.

Our approach to increase awareness on SIDs and related SDE and obtain increased ROI from BPI initiatives, and attain the final objective of developing a suitable SDEE method, is presented in the following chapters, as follows:

- Chapter 2 presents our approach in detail;
- Chapter 3 presents the related work from other researchers for this problem;
- Chapter 4 presents our plan and what has been done so far in order to achieve statistical evidence that our approach can enhance the state of the art for this problem;
- Chapter 5 presents the possible conclusions on our work at this point of its evolution.
2 Bridging Enterprise Engineering and Software Engineering: The Goals Approach

There is the common knowledge that a massively relevant number of software projects (around 60%) still fail today, by not being 100% successful (late, over budget, and/or with less than the required features) or not finishing at all [1], and that there are several reasons pointed out following project and related organizational nature analysis (e.g., Resistance to change, Imposition and Commercial Pressures) [5]. We argue however, that there are higher level causes that affect this lack of software development performance in a more powerful way, which are:

(1) A general loss of focus on what can bring project success, especially on obtaining positive Return of Investment (ROI) from the BPI (as only around 22% of software development processes consider ROI natively in their structure [8]), which we believe that can bring an increased alignment between project stakeholders;

(2) Not involving end users in the BPI conception process, and therefore develop worst software and because of that, make people less happy (more stressed for instance), as working tools’ performance, such as the user interface usability, may have a real impact in the worker’s emotional stability (mood) [9], and;

(3) A lack of awareness on the impact of software design decisions in the overall SDE, as there are SIDs that can dramatically increase or decrease the effort of a software project for the development of certain requirements [10] and therefore avoid financial problems or even “catastrophic” situations [2].

Hence, we believe that once organizations are taken care of as a whole, from both managerial and software development perspectives, and are also more aware of the benefits and implications of improving a BP’s performance, and know how to do it in a consistently advantageous way, both software project success and organization welfare will probably be increased.

In the same line of what is pointed out by the Standish Group as a main reason for project failure, i.e. the increasing difficulty of managing great complexity, as large projects (over $10 million) are “more than 10 times more likely to fail” than smaller projects (under $1 million) [1], we also feel that focusing on what can bring project success will definitely lead the way as the solution to increase SDP performance, which can be achieved in the following way:

- **Defining BP-sized projects**, as it is specifically at the level of BPI that both EE and SE professionals “meet” in order to organize respective knowledge and tasks, resulting in more manageable projects with objectives that can be easily settled between those professionals and future users (every project stakeholder).

- **Increasing worker needs comprehension**, which can be achieved by obtaining a common definition to EE and SE of the concept of User Task (UT), in order to understand what the worker (user of the IIS in our scope) needs to accomplish within a certain BP’s goals, to enable improving its working performance.

- **Understanding working spaces**, which can be achieved if the Interaction Spaces (ISs) (user interface in SE) can also be recognized in EE as a valuable organizational artefact in order to better scope the supporting and remaining part of the IIS’s conception, i.e. Business Rules (BRs, business logic in SE) and Data Entities (DEs), as DEs are accessed according to existing dynamics of Business Rules (BRs).
By focusing on achieving ROI from BPI, as this might be the key to encourage BPI within organizations especially if profitable software development patterns can be identified at BP granularity, which increases the hope of successful organizational changes amongst enterprise engineers.

This logic establishes a relation between our 5 concept approach of an organization: BP, UT, IS, BR and DE, as the bridge to enable a wider understanding between the EE and SE domains to enhance SDP performance.

Hence, the Goals approach defends the existence of an organizational development process that supports the IIS’s adequate development to meet BPI needs, while it is aware of its current performance state, and as a result of that, produces the necessary EE and/or SE activities to increase its own performance. Although the main objective behind our work is to develop an SDEE white-box method as the key to understand ROI from BPI, our approach stands on necessary intermediate contributions. Hence, the main contributions to support our approach, which are detailed in the next sections (after section 2.1 Foundations) are the following:

• An approach logic that combines EE, SE and Human-Computer Interaction (HCI) domains efforts to properly support IIS development within organizations (section 2.2 Support).
• An organizational structure to enable IIS development and analysis (section 2.3 Architecture) from an holistic enterprise-sized perspective;
• An enterprise engineering logic to enact management from the perspective of the enterprise’s own performance, towards the IIS’s development and maintenance needs (section 2.4 Engineering).

2.1 Foundations

In order understand the bridging process, it is necessary to understand the context within which the tasks of developing adequate software in order to support profitable enterprises, occur, i.e. the organization and the information system. An organization, as referred in the present document, is the act of organizing the enterprise, including its IIS system. Hence, enterprise engineers and software engineers are not necessarily part of the enterprise, yet, they will be the most important workers of the organization.

Enterprise

An enterprise, is by definition an initiative, complex project, a business or a company. In fact we can look at an enterprise as a project of a business (or company) that constantly (or only when needed) produces a product that is needed by the society. We can assume, from the beginning, that an enterprise, will eventually end its activity, as from its success, competitors, will (also eventually), with regained knowledge and better organization levels, obtain more efficient production processes that will drive the originally successful enterprise to its end. This seems to be inevitable for enterprises e.g. in the automobile field, but not necessarily in the educational field, where universities can sustain themselves for hundreds of years (e.g. Bologna, Oxford, and especially the University of Karueein from 859 AD [11]).
Our definition of enterprise follows the idea that people and resources are usually needed for the BP-based production to occur, and organizational goals to be achieved. We also define an organizational goal, as an objective that relies on one or more BP goals performance, i.e. by maintaining a certain level of production, organizations will achieve certain market goals, usually called strategic goals.

As we believe that it is not necessarily true that an enterprise needs humans to properly work, as, if the only resource would be information, and clients could access the needed information without recurrence to human intervention, then, the only humans needed to maintain the enterprise working, would be those who belong to the organization, i.e. enterprise and software engineers. Hence, we are leaving people and users out of our definition of enterprise. We define an enterprise as follows.

An enterprise is a structure of BPs that share common resources in order to attain organizational goals.

We define and organizational goal as follows.

An organizational goal is an objective that relies on one or more BP goals performance.

Process Use Cases (SE derived) [12] and Activity Modeling (Human-Computer Interaction (HCI) derived) [13] provide methods to identify BPs, activities, humans and resources, in a way that they are compatible with our enterprise definition, i.e. they allow the modeling on how to reach a specific organizational goal or a (BP) goal, and also the organization of the enterprise, by means of the identification of use cases. The Systematic Approach (EE oriented) [14] also applies the same approach in order to properly identify use cases and facilitate software development.

The next figure presents an example of BPs interaction from Process Use Cases.

![Partial view of Process Use Cases’ BP interaction in an organization.](image)

**Fig. 1.** Partial view of Process Use Cases’ BP interaction in an organization.

**Information System (the Software)**

The information system definition that is considered in our approach, is the Interactive Information System (IIS). An IIS is a system that allows human’s interact by means of a User Interface (UI, which is an IS) without the usage of specific written commands i.e. actions can be carried out using e.g. a common mouse (considering a 2D interface i.e. a monitor or a television as examples). Our definition complies with the commonly adopted and recognized SE structure of the MVC pattern [15].
MVC stands for Model (data organization), View (UI) and Controller (business rules and transformation), and complies with our definition for Data Entity, Business Rule and Interaction Space respectively, and are sufficient for the representation of an IIS dependency structure i.e. an architecture. The proposal to represent a software structure and user tasks in a single dependency structure, was initially presented by Wisdom [16], pioneer Human-Centered Software Engineering (HCSE, a SE and HCI bridge), and adopted by our approach, is depicted in the next figure.

![Partial view of a Wisdom' MVC compatible software architecture](image)

**Fig. 2.** Partial view of a Wisdom’ MVC compatible software architecture [20].

### 2.2 Support

In order to support our approach to what we believe that should be a complete SDP bridging EE and SE, that complies with needed management activities and IIS enhancements, we introduce a Software Process Improvement (SPI) initiative that gathers knowledge from the EE, SE and HCI domains, to provide the needed methods for documenting Business Processes (BPs) and related User Tasks (UTs), existing Interaction Spaces (ISs), Business Rules (BRs) and Data Entities (DEs), in terms of a process that will identify BPI needs and produce the needed information for EE and SE activities that will contribute to enhance the organization’s performance.

The methods chosen to support our approach for the modeling and definition of the concepts of: BP, UT, IS, BR and DE (section 2.3. Architecture); plus the logic (section 2.4 Engineering) of the production of the needed modeling artifacts from the initial identification and documentation of BPs up to the final IIS’ definition; are:

- **DEMO [17]** from EE, is a solidly and notably established EE theory and methodology for the complete engineering, definition and modeling of an enterprise;
- **Goals** [18] from SE is a methodology derived from Wisdom [16] that allows the structured design of the IIS from requirements elicitation up to the identification of relevant components and their dependencies definition;
- **Activity Modeling** [13] is a performance-oriented HC (also applied in SE, namely in HCSE) methodology for the conception of detailed user interfaces, based on the analysis of the UTs carried out within the enterprises’ activities.
2.3 Architecture

The specific topic of BPI is the cornerstone of the EE and SE bridge if we wish to establish a language that both professionals find out to be simple and understandable, and also allows the enhancement of managing and operating the organization. We now present our concept conciliation of the BP, UT, IS, BR and DE concepts.

**Business Process (BP)**

EE and SE had distinct approaches in the way they discovered the importance of modeling BPs as an added value to architect both an enterprise and a software system. On the EE perspective, the BP is the natural backbone of the organization (involving at least one transaction in DEMO), whereas in SE, the importance of the BP only became clear when it was clear that tasks carried out in the system could be common for distinct sets of tasks (i.e. BPs) and that they could be re-organized in a BPI.

This concept cannot be defined without the UT definition, as User Tasks provide the granularity level for what is to be considered a manageable set of UTs that lead to a certain BP’s goal within the organization’s architecture. We define BP as follows.

_A BP is a set of complete and meaningful tasks that lead to a goal._

Example of BP: Sell Dinner (at student’s canteen).

**User Task (UT)**

UTs differ in terms of scope from the way they are considered by both EE, SE and HCI domains. In EE they are considered as transaction’s acts whereas in SE they are actions carried out in a user interface, and in HCI they can even be expressed as user intentions and “lowered” up to the level of one single click.

Nevertheless, the conciliation of this concept can be settled for the tree domains by the definition of Use Case from HCI’s Activity Modeling [13], since its granularity defined by “a complete and meaningful task” is suitable for both missions of organizing the BP’s activities in transaction acts, and also conceiving the IIS in SE. A UT is “complete” in the sense that the user cannot do anything more within its responsibility, and is “meaningful” in the sense that it is a concrete step towards achieving the BP’s goal. We define a UT as follows.

_A UT is a meaningful and complete task within a BP._

Example of UTs to the Sell Dinner BP: Student Selects Dinner; Student receives Dinner (from attendee); Student Pays Dinner (to attendee).

**Interaction Space (IS)**

The user interface is a crucial concept in SE and HCI that however, loses strength in the EE domain, in the way that it can be seen as an implementation issue, in the same way that in SE implementation options should not interfere with system’s conception.
Although the user interface is not considered an artifact in EE, we yet believe that the definition of IS, initially provided by Wisdom [16], if extended to 3D spaces to allow activities outside the IIS for the completion of one UT, can provide a valuable concept as it gives support for the execution of BP task in a 2D (directly in the IIS) or a 3D environment, which are both meaningful and complete. We define IS as follows.

An IS is a physical space (3D or 2D) where a complete and meaningful UT occurs.

As examples: The student Selects Dinner in (a 2D) Point of Sell (POS); and the Dinner is delivered to the student by the end of the balcony (in the 3D canteen).

**Business Rule (BR)**

The BR is a concept that is supported in EE by DEMO that has a correspondence in SE considering the decomposition of Use Cases as “User Intentions” and “System Responsibilities” (SRs), as SRs are “what the system must do” to respond to the user’s interaction (or intention) (also known as business logic in SE). This decomposition has a strong parallelism in DEMO in the Perform, Inform and Form data exchange, as the performer responds to addressee’s intention.

Our definition compatibility is achieved by means of what an Action Rule is in DEMO (or in its structured extension DEMOBAKER [19]) and its implementation as SR, which is a “piece of code” to be implemented in any SE programming language (e.g. Java or SQL) as a top-level programmed code structure (e.g. a class) that uses other SRs and DEs to implement a BR. Other finer grained SRs part of the IIS’s architecture, such as data processing, are still within our scope, as the Datalogic and Infoligic concepts, which define data before and after patterned data processing, such as DEs combinations, or statistical processing. We define a BR as follows.

A BR is an ontological structure for the exchange of data between IS and DE.

As an example of BR, a student can only Select one Dinner per day.

**Data Entity (DE)**

DE is a concept that is dominated by both EE and SE domains, as databases, a foundation of SE, are a worldwide accepted concept that also complies the Table concept. The definition of DE is provided by SE as a set of Tables combined in a single structure, compatible with DEMO notion of Data Bank for a Datalogic unit.

Furthermore, the concept of Class Object in EE is also compatible with the notions of Programmable Class (PC) and Table Field (TD) in SE. Extendedly, PC’s methods are compatible with notions of Coordination and Production Acts, as they can trigger other routines on the existing IIS. We define a DE as follows.

A DE is a structure that provides data and acts for BRs.

As an examples of DEs, are the Student Personal Data, and the Student’s Account, involved in the Sell Dinner BP.
2.4 Engineering

The (maintenance) engineering process behind our approach acts in the form of an agent that monitors the organization’s BPs, the UTs and ISs, and is inspired in the DEMO based Control Organization process [20]. This Agent for Enterprise Logic (we called it Angel), after monitoring the organization’s performance for these components, will act in the sense of improving its IIS performance (ISs, BRs, DEs), weather driven from an opportunity to eliminate “dysfunctions” caused by “exceptions” to ensure “viability norms” (Resilience), or, to improve one or more of these components in an Organizational Engineering (OE) process (Microgenesis).

This process, not only allows the identification of BPI needs (Microgenesis), but also tries to avoid them, by introducing minor adjustments (Resilience) in order to solve operational problems and maintain the necessary levels of production to attain the desired organizational goals. Hence, it provides an organizational awareness on its own performance, which will combine EE and SE efforts to increase it by means of the maintenance and development of the complete organizational system, i.e. enterprise and IIS. The next figure depicts the Goals approach.

Figure 3 presents the following:

- The DE, BR, IS, UT and BP stereotypes;
- The science domains (long triangles) that support the implementation and conception of these components (large dots with straight lines);
- The effort to acquire new knowledge (long lines with small dots);
- The extent of knowledge that we think that each science domain should do in the future in order to better support each concept;
- The monitoring on BPs, UTs and ISs, made by the Angel (red long arrows);
- The Angel action over ISs, BRs and DEs (straight lines with small diamonds).
3 Related Work

The related work concerning our approach focus first on existing solutions for the mechanism of obtaining value from distinct possible SIDs for the same requirements development in a BPI initiative, the “point” where enterprise managers (EE professionals) and system developers (SE professionals) “meet” to enhance the organization’s performance. From literature review we found that there is still a lack of focus in obtaining ROI from BPI, as there are still relevant more concerns on how to coordinate change management efforts [5]. Nevertheless, there are approaches that have already elicited the importance of this approach, namely GQM+ and VSDP, which are methods for value estimation, which however should only be considered for the specific purpose of ROI analysis, and not for the logic of the dynamic that is needed in order to document the ROI from BPI mechanism.

GQM+ Strategies for Business Value Analysis [21] is a Value-Based SE (SBVE) derived method that has it focus on the usage of Goal Question Metrics (GQM) combined with Business Goals to understand Business Value, i.e. the importance of each SE project, its ROI and risk factors.

Value-oriented Solution Development Process (VSDP) [22] is a value-aware DEMO-based system development process that defines a value model based on investors needs in order to identify solutions for the problem, and select the most profitable implementation scenario. The following formula (OPEX stands for operational expenditure), focus on obtaining dividends from each (BPI) investment: \(\text{dividend} = \text{customer revenue} - \text{business OPEX} - \text{IT OPEX} - \text{investment}\).

Our approach differs from the previous ones in the perspective that it relies on fomenting organizational self-awareness knowledge, especially in terms of a consistent relation between the management decisions (that mostly affect BPs and UTs) and its implications on the supporting IIS (ISs, BRs and DEs) as the path to understand how to properly organize an enterprise, to provide an holistic approach to the SDP performance and related SDEE problem.

Secondly, and regarding the main contribution of our work i.e. the identification of SDP patterns that allow increased ROI from BPI based on an accurate SDEE, we need to focus on attaining a reliable SDEE white-box method. In this specific field, traditional approaches (notably COCOMO [23], Function Point Analysis [24] and the original UPC method [25] after 2 decades of evolution and application to SE still don’t fit industry purposes [26].

We find the Normalized Systems (NS) theory [27] to be the main contributors for our work in the sense that they provide extensive valuable material for the purposes of developing “stable” and evolvable systems in the way that they aim at developing “ideally” structured systems, and can provide a comparison term regarding final IIS architecture for our future set of sample projects, in our objective of tackling existing environmental problems (e.g. framework changes and requirements stability) that affect IIS’s conception, by means of eliciting the reality within which software development happens, and how to do it in the most simplified and understandable way for EE, SE and also HCI.
4 Research Plan

Software development is an activity where structural design decisions affect the course of the implementation until its end. Structuring software in the correct way is a task that involves acknowledging every single component of its architecture, and understanding that it is a “solid” conceptual structure (that represents code and data) which is not easily transformed, especially if this transformation involves the more important parts, is crucial in our opinion to be able to accurately estimate SDE. Existing SDEE methods [23, 24, 25] contemplate system components, yet, they don’t reflect their importance in a white-box SDEE method that reflects the complete architecture and each component’s implementation and dependencies transformation difficulty, in order to make it (the SDEE method) accurate.

Figure 4 illustrates the difference between looking at a structure as if every single component had the same importance (on the left side), and the same structure reflecting the importance of each component (number of records in DE, number of BRs and “projected” importance in related IS) and their relation to UTs and BPs.

From the analysis of the previous Figure, we can be led to believe that changing the “Fill Personal Data” and “Make Registration” use cases may not take the same effort.

The main objective of our research is to further develop the iUCP [28] method in order to contemplate the full IIS architecture (ISs, BRs and DEs) to increase accuracy. Therefore, our approach to further extend iUCP towards the complete IIS structure and further tackle existing problems affecting negatively software development (where SDE stands for Software Development Effort) are:

**Q1: How can the SDE be related to each architectural component?**

For Q1 our hypothesis are:

- **H1.1**: SDE for DEs depends on the number of existing records;
- **H1.2**: SDE for BRs depends on the number of DEs in which it depends;
- **H1.3**: SDE for ISs depend on the number of BRs in which it depends;
Moreover, and considering the existing software development environment, and as a continuation of previous research were we found that the choice for a new software framework could add up to 75% of the effort of development, and requirements instability could also impact this effort in 50%, whereas a stable maintenance project can reduce the effort for the implementation of the same requirements in 25% [10], we also which to inspect hypothesis related to the environment within software development occurs, namely software framework implementation choices and requirements stability.

The research method to acquire the necessary data to support our hypothesis (where $MRE = (\text{Real effort} - \text{Estimated Effort}) / \text{Real Effort}$) is the following:

- **Project Analysis**
  - Analyze project requirements in the perspective of a BPI;
  - Apply iUCP;
  - Analyze SIDs;
  - Analyze initial and finished architecture;
  - Calculate MRE;
  - Estimate ROI (if BP necessary metrics exist).

Following our previous work we believe that no more than a sample of 50 projects will be needed to achieve a significance of 0.05 (sustaining an effect size around 0.55 with a power of 0.8) in order to achieve statistical evidence for our hypotheses.

Nevertheless, in our path to achieve this SE enhancement, we still need to develop a reference structure based on the NS’s theory that stands as an effort reference for SDE in order to evaluate sample project’s final architecture and related effort, and therefore, be able to understand what other relevant environmental factors that may affect SDP performance.

## 5 Conclusions

It is our belief that from the exchange of knowledge from both EE and SE domains, enough expertise can be produced in order to improve the way that IISs are produced, up to the point where chaos reports will only be a mirage, and organizations will not go into bankruptcy because of failed projects, and the focus will e.g. be on how to give users the best interaction functionalities while decreasing SDE and increasing ROI, BPs and UTs performance, including user experience (UX) levels [29].

Moreover, we also believe that if we can statistically support our research hypothesis, and therefore enhance SDEE accuracy and develop the necessary patterns to understand how to get the best possible ROI from BPI and increase IIS architecture adequacy, our approach may lead to a change of paradigm in the way that BPI and related software patrimony are managed within organizations as development success will be attained in a more gentle and natural way, as envisioned by Christopher Alexander in the generalist (although applied to building construction) Timeless Way of Building process of constructing a system [30], weather it is a country, a city, an enterprise, a BP, its tasks, or an IIS.
We which to emphasize that our approach is not supposed to be an alternative to DEMO concerning the EE domain, as it partially supports the same concepts, yet, only for purposes of reaching the same level of comprehensiveness, but to bring EE closer to the software development reality, and also bringing the SE domain closer to enterprise management complexity reality.

Although this document presents our approach to what we believe to be a feasible project with relevant results, we are also aware that other fields of study, such as semiotics or business engineering can eventually provide new sights on the proposed approach.

We also which to highlight that there is still a relevant an arduous work in our path to attain a consistent white-box SDEE method, especially concerning the definition of our sample projects, and an incisive and rigorous strategy, based on related existing work, namely NS and VSDP, that aims at achieving statistical evidence to support our beliefs of developing better organizational software in a more consistent and profitable way.

Finally, we which to highlight that we intentionally left the User definition out of the scope of our analysis, as we believe that as humans (weather they are “simple” workers or not) are an inherent part of a correct software development process, as not involving Users (or workers) in the development process is not an option. A User is someone or a group of people with the same responsibilities that uses the IIS.

6 References

http://www.guinnessworldrecords.com/world-records/oldest-university (18th April 2015)