

**Submitted to the University of Hertfordshire in partial fulfilment
of the requirement of the degree of Doctor of Philosophy**

School of Physics, Engineering and Computer Science

***Modelling the Political Context in Software
Requirements Engineering***

Rana Salemi (Siadati)

June 2021

Abstract

The influence of stakeholder politics on the outcome of software development projects has been identified as a major issue in the practice of software Requirements Engineering (RE) that has been left substantially overlooked in the field and practice of RE. I argue that *politics and power* are crucial aspects not just influencing RE but being part of it and integral components of the process. A practical solution in the form of a notation that will aid practitioners both to identify explicit and implicit stakeholders and to document the current state of their degree of support for that project is proposed.

An Emoji pictograms-based notation called Political Emoji Notation (PEN) was created which is based on an existing well-known visual language, easily adoptable by requirements engineers. RE professionals will be able to identify and document power, politics and the emotional aspects that come into play during software requirements-related decision-making. PEN is intended solely for the use of RE and their team, ensuring complete confidentiality and avoiding any political engagement with the customer.

The notation is deliberately kept simple, to minimise the learning process and enable practical use without the need for specialised software for drawing diagrams, and is readily accessible and user-friendly. The notation underwent testing using two case studies of unsuccessful software projects well documented in the literature. From the analysis of the two case studies, a set of symptoms has been identified to identify projects that are at risk.

Finally, the thesis proposes potential directions for future research and work based on the implementation of this notation.

Acknowledgements

I extend my sincere thanks to my exceptionally supportive research supervisors, Dr Vito Veneziano and Dr Paul Wernick, for their invaluable advice, continuous help and patience beyond one's imagination at every stage of this PhD study. Undoubtedly, I could not have achieved this without their backing. I am forever grateful.

My special appreciation also goes out to my dear husband, Mohammad, for his encouragement and unconditional support not just during my PhD but throughout my entire academic journey.

To my lovely children, Samyar and Yasmin, for putting up with my studies: my deepest gratitude. I am grateful for their love and motivation that helped me through difficult times.

I would especially like to thank my colleague, Sarah Hussaine, for her kind help and support during the past 18 months.

Last but not least, to my mother and late father for being my first teachers and teaching me to believe in hard work and appreciating human kindness, you are always in my heart.

Table of Contents

List of Figures:	7
Chapter 1: Introduction	8
1.1 My Underlying Thinking	8
1.2 The Thesis Structure in Context	9
Chapter 2: Software Engineering, Software Requirements Engineering and Requirements Engineer	14
2.1 Background.....	14
• 2.1.1 The roots of RE in the (historical) process of Software Development	15
• 2.1.2 Requirements Engineering: evolution and current status	19
• 2.1.3 The Requirements Engineer’s role.....	25
2.2 The emergence of RE as a recognisable discipline	29
2.3 Politics and emotions in Requirements Engineering.....	30
2.4 Where this research comes from.....	32
2.5 Initial Research Goals and Questions.....	35
Chapter 3: Problem, Goals and Questions	38
3.1 Technical and non-technical (or socio-technical) factors	38
3.2 Organisational politics.....	40
3.3 Industry and Academia.....	41
3.4 Professional Bodies	42
3.5 The Requirements Engineer and identifying politics in an Organisation	43
3.6 Support/Champion.....	46
3.7 Research questions and goals (plus why notation)	47
• 3.7.1 Why and how to focus on RE: politics as the conceptual key	47
• 3.7.2 Where is RE from? Politics and power underlying human activities	51
• 3.7.3 What is RE really about? Politics and power in action: the subtle art of negotiation	54
• 3.7.4 Who does the RE job? The politically vested analyst and engineer.....	60
3.8 How a politically aware RE relates to other non-technical factors?.....	61
3.9 A need for tighter definitions? Or a need for better definitions?	62
3.10 Methodology.....	63
• Method 1 - Literature Review and historical analysis: a methodological approach.....	63
• Method 2 - Interviews and personal communications	64
• Method 3 – Development of a simple, intuitive, additional notation to represent and model political relationships (more on following Chapter 4).....	65
• Method 4 – Impact Evaluation on identified case studies.....	66
Chapter 4: Modelling	67
4.1 Modelling politics in software requirements engineering	68
4.2 Discussion.....	74
4.3 Including the emotional side of the political dimension in modelling	78
4.4 Impact evaluation based on case-studies	80

Chapter 5: Applying the Notation	83
5.1 Introduction	83
5.2 Case study 1 - Chris Sauer paper and the case study: impact evaluation on a consolidated study	85
5.3 Case study 2 - the National Program for IT (NPfIT) in the National Health Service (NHS)	106
5.4 Conclusive analysis over the two case studies: look for symptoms!	117
• 1 st symptom: check whether smiling faces decrease along projects' life cycle	117
• 2 nd symptom: look for unaware smiling faces	118
• 3 rd symptom: try to monitor everything by regularly pursuing a helicopter view	118
• 4 th symptom: check whether the number of actors keeps increasing (unreasonably and unexpectedly)	118
Chapter 6: Results and Findings (overall, how well the problems were addressed?)	120
6.1 A background recap to better focus on the aims of my research	120
6.2 Revisiting research questions	121
• 6.2.1 Revisit - Why and how to focus on RE: politics as the conceptual key	121
• 6.2.2 Revisit - Where is RE from? Politics and power underlying human activities	122
• 6.2.3 Revisit - What is RE really about? Politics and power in action: the subtle art of negotiation	122
• 6.2.4 Revisit - Who does the RE job? The politically vested analyst and engineer	123
6.3 Modelling and Applying the notation	124
Chapter 7: Conclusions and Future work	127
7.1 Overall Conclusions	127
7.2 Future work	128
7.3 Contribution to knowledge	129
• 1 st symptom: check whether smiling faces decrease along projects' life cycle	130
• 2 nd symptom: look for unaware smiling faces	130
• 3 rd symptom: try to monitor everything by regularly pursuing a helicopter view	130
• 4 th symptom: check whether the number of actors keeps increasing (unreasonably and unexpectedly)	130
7.4 Closing thought	130
References and Appendices	132
Appendices:	142
A – Publication:	142
• Paper 1:	142
• Paper 2:	142
• Paper 3:	142
B – Poster	143
C – evaluating software tools and methods given certain constraints	144
D – Analysis	148
• Case study 1:	148
• Case study 2:	148

List of Tables:

Table 1: Initial hypotheses and conjectures VS research goals	37
Table 2: Success measurements for RE including and/or referring to politics (derived from Fricker et al, 2015)	55
Table 3: List of negotiation techniques (Fricker et al. 2015).....	60
Table 4: A notation for graphical modelling of politics in requirements engineering	69
Table 5: Summary of a framework for understanding support and support management	84
Table 6: Identification of actors and their roles in based on 5 stages specified in Sauer’s analysis	86
Table 7: Identification of actors and their roles in stage 1 as specified in C Sauer’s analysis.....	89
Table 8: Identification of actors and their roles in stage 2 as specified in C Sauer’s analysis.....	94
Table 9: Identification of actors and their roles in stage 3 as specified in C Sauer’s analysis... 	100
Table 10: Identification of actors and their roles in stage 4 as specified in C Sauer’s analysis. 	102
Table 11: Identification of actors and their roles in stage 5 as specified in C Sauer’s analysis. 	104
Table 12: Identification of actors and their roles between 2002 to 2009 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis.....	107
Table 13: Identification of actors and their roles in 2002 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis	108
Table 14: Identification of actors and their roles in 2003 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis	108
Table 15: Identification of actors and their roles in 2004 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis	109
Table 16: Identification of actors and their roles in 2005 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis	110
Table 17: Identification of actors and their roles in 2006 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis	111
Table 18: Identification of actors and their roles in 2007 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis	112
Table 19: Identification of actors and their roles in 2008 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis	113
Table 20: Identification of actors and their roles in 2009 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis	114
Table 21: Identification of actors and their roles in 2010 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis	115
Table 22: Identification of actors and their roles in 2011 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis	116
Table 23: List of all the criteria adopted for assessing relevance of sources.....	144
Table 24: Comparative application of above criteria against two sources	145

List of Figures:

<i>Figure 1: Provided by McClure (Figure 6. from Naur and Randell 1969:38)</i>	20
<i>Figure 2: Provided by McClure (Figure 1. from Naur and Randell, 1969:42)</i>	22
<i>Figure 3: Relative cost to fix bugs, based on time of detection. Credits to deepsource.io, 2019.</i>	38
<i>Figure 4: Problem causes (“Project Impaired Factors”), Standish Group, 1995.</i>	39
<i>Figure 5: Possible list of stakeholder roles on projects</i>	44
<i>Figure 6: The "Onion" model of stakeholders in a typical system.</i>	45
<i>Figure 7: Why Focus on Requirements</i>	48
<i>Figure 8: What do really users want?</i>	55
<i>Figure 9: Outcome of Informal relationship/influence: change of views/status</i>	74
<i>Figure 10: Traditional organigram</i>	75
<i>Figure 11: Direction of power (the usual assumption behind the organigram)</i>	75
<i>Figure 12: Organigram showing relative power (with multiple arrows)</i>	76
<i>Figure 13: Organigram showing relative power and status/views of each person.</i>	76
<i>Figure 14: Informal influence of D on A</i>	77
<i>Figure 15: Changes of views/status after informal influence of D on A</i>	77
<i>Figure 16: Political relationship, Example 1</i>	78
<i>Figure 17: Political characterisation within a UML Class, Example 2</i>	79
<i>Figure 18: Sauer’s conceptual framework for theorising about support and support management (Sauer, 1993)</i>	83
<i>Figure 19: Case study 1 - Model 1A</i>	90
<i>Figure 20: Case Study 1 - Model 1B</i>	91
<i>Figure 21: Case study 1 - Model 1C</i>	92
<i>Figure 22: Case study 1 - Model 1D</i>	93
<i>Figure 23: Case study 1 - Model 2A</i>	96
<i>Figure 24: Case study 1 - Model 2B</i>	97
<i>Figure 25: case study 1 - Model 2C</i>	98
<i>Figure 26: Case study 1 - Model 2D</i>	99
<i>Figure 27: Case study 1 - Model 3</i>	101
<i>Figure 28: Case Study 1 - Model 4</i>	103
<i>Figure 29: Case study 2 - Model for year 2002</i>	108
<i>Figure 30: Case study 2 - Model for year 2003</i>	108
<i>Figure 31: Case study 2 - Model for year 2004</i>	109
<i>Figure 32: Case study 2 - Model for year 2005</i>	110
<i>Figure 33: Case study 2 - Model for year 2006</i>	111
<i>Figure 34: Case study 2 - Model for year 2007</i>	112
<i>Figure 35: Case study 2 - Model for year 2008</i>	113
<i>Figure 36: Case study 2 - Model for year 2009</i>	114
<i>Figure 37: Case study 2 - Model for year 2010</i>	115
<i>Figure 38: Case study 2 - Model for year 2011</i>	116

Chapter 1: Introduction

1.1 My Underlying Thinking

By means of this writing, I am hereby reporting about the research that had been carrying out as part of my PhD project.

The primary aim of this research, at the very beginning, was to investigate how and why requirements engineer's role has been changing over the years, in the hope of helping professionals and academics to make sense of the present state of affairs and to face (if not to drive) any future evolution of the job (and the discipline) by relying on some sounder conceptual and interpretative tools. Such an idea had been concreted into a number of goals, which in turn have been decomposed into a list of hypotheses and conjectures.

Core to this part of research goals was therefore the development of a comparative and critical history of requirements engineering (RE), which in itself has proven to be an extremely challenging and yet rewarding task, due, on one side, to the lack of a consistent, homogeneous and well-grounded body of historical information, and on the other, to the myriads of factors, inherent to the practice and the surrounding conditions it takes place within, that should be considered responsible for making RE what it is at present.

Also, I could not pursue the development of a history of RE without having previously considered the broader perspective offered by software engineering, mainly because the same term RE is more recent than its actual practice. A systematic approach was adopted and some of the reviewed sources (in particular, Naur and Randell (1969), the report and the proceedings of the NATO conference in 1968) was discussed as part of the early research.

Some factors affecting the development of RE and its practitioners' professional identity, had been highlighted and used across several argumentations.

As my research progressed, its aim was still unchanged and a comparative literature survey and interviews were still the methods by means to gather needed information in order to develop a complex and yet consistent picture of RE, I had read, further investigated and discovered a crucial aspect that has been left substantially overlooked in the field and practice of software requirements engineering. I cannot think of another case in which such a large elephant in the room has been left mostly unacknowledged (although, it is fair to claim that some such as iStar, Yu (2011) sources did indeed acknowledge it, and by doing so they provided my research with the backbone it needed).

If, personally and professionally, this had meant for me a kind of "awakening" in how we interpret and pursue requirements engineering from a more neutral, academic and intellectual point of view I feel confident I was then in a better position to review my previous impression of the state of requirements engineering and view it under a new and more insightful perspective.

I detected, and by this thesis I am hereby arguing, that *politics and power* are crucial aspects not just influencing RE, but actually being part of it, and that the role they play, especially when applied to the software industry, needs to be given greater attention than is currently the case.

It also came to my attention, how along the years, RE have left politics substantially overlooked? Whilst many of the sources that I used in this research are not recent, but it has been contended that the underlying issue was still in need of urgent attention.

Although several authors (e.g., Wirth (2008) and Milne and Maiden (2012) have recognised the conceptual key of “politics” as an important component in any Requirements Engineering (RE) process, however practitioners have still not been given a tool or method to easily detect, represent, control and if possible, leverage politics.

I argue that this issue can be successfully addressed if REs can develop models that include an extra layer of "political" information using a simple new notation. I have therefore developed an Emoji pictograms-based notation which is based on an existing well-known visual language and so could be easily adopted by requirements engineers. This notation helps REs to recognise and document power, politics and the emotional aspects of software requirements-related decision-making in customer organisations.

I believe the time to address the problem REs are facing has been reached, and they need to have a better understanding of the customer organisation to enable them to identify its problems and hopefully solve them within its political context. Given the sensitivity of organisational political issues, I have assumed that diagrams created using the above-mentioned notation are only for the software requirements engineers and their team thus remaining strictly private while avoiding any political involvement with the customer.

As a test of my approach, I have considered two case studies into software projects which have failed, in my opinion due in part to political aspects but whose failure is relevant to software requirements engineering.

I have concluded that my proposed graphical notation, even if not solving the problem, at least might raise awareness in requirements engineering and help them address it, and so would bring them closer to solving the problem.

1.2 The Thesis Structure in Context

In 2018, the 50th anniversary of software engineering was celebrated. The anniversary that was described as “50 years of tremendous by successful promotion of research, education and practice” at the 40th International Conference on Software Engineering (ICSE).

The history started when Margaret Hamilton named the discipline “software engineering” during Apollo missions (Cameron 2018): the “*unbundling*” of software from hardware sales in 1969 by IBM and the role of NATO conferences (1968 and 1969) in development of software engineering were also key in forming the discipline.

Requirements engineering, the first phase in the development process, plays a crucial role in software engineering to understand, define, document and maintain stakeholder’s requirements for the purpose of software development.

Udousoro (2020) includes elicitation, specification, verification, validation and management as some of the activities that can form part of the role to assist requirements engineers, but the initial focus is to gather information about the requirements of a proposed software system or the project in general. There are different techniques and models to help with these activities

but one of the main aims of having a requirements phase in software engineering is to produce a statement of what the software will do, in order to reduce software errors, enable the production of high-quality software within budget, gain stakeholder's satisfaction and in general, to meet the targets of the development activity.

To answer how successful software engineering has been, I have no doubt that no-one could have expected to see this much of dependence on software in our short professional life, in the past half a century. However, despite all the effort and the dependency of our day-to-day life on software, still there are many over-budget or failed software projects around the world; the details of at least some of them are easily accessible on-line.

To start our journey and to find a solution (if any), in chapter 2, and section 2.1 in particular, the detailed background to this research has been set up. It has been explained where the original idea and aspiration for this research comes from, and the necessity for looking to the history of software and requirements engineering.

Section 2.1 divides into sub-sections to present a detailed literature review into the history of software engineering, and the roots of requirements engineering in the process of software engineering including a reminder of some difficulties which have been faced (2.1.1). Whilst the "technical" activities and current status of requirements engineering is a given, non-technical users and different user groups have been looked at as a crucial challenge, because users and other stakeholders (sometime? often? always?) did (and do) not know what they actually want. The traditional assumption that requirements must be "captured" (as if they exist already and fully clarified in someone's mind) is not accepted any longer by practitioners. Plus, a number of possible factors led to the definition and evolution of the requirements engineer's role in the IT and the software industry was discussed too (2.1.2). The origin of requirements engineering, and the role of requirements engineer, globalization, role of social networking and summery of the primary factors (which possibly have contributed to the evolution of the requirements engineer role in the software industry and changes in its definition) are considered in sub-section (2.1.3).

Section 2.2 begins with brief touch again on requirements engineering and the term "Engineering" in particular. The role of Garmisch and Rome conferences (Naur and Randell, 1969) as yet another factor that might possibly have helped requirements engineering to become a recognizable discipline, as well as the "software crisis" and general difficulties in the discipline with relevant references from other academics/practitioners.

The place of politics and emotions in Requirements Engineering are considered in section 2.3, where an argument has been provided to highlight a simple way to model any captured emotions by using emoji pictograms: although some of them are specific to specific cultural settings, the majority of them seem convey meaning rapidly across a shared universal language (Azuma and Ebner, 2008). This could be helpful to requirements engineers operating worldwide, as they could assess and quickly produce models that come with an extra layer of information, namely beyond any international translation difficulty and without the need to introduce any new notation.

So far, most efforts spent to solve problems in requirements engineering have been related to addressing technical factors and components: but one of the aims of this research is to highlight the role of non-technical factors alongside the technical factors in the field. This is important because the issue of politics cannot be solved by technology, and this leads to the content of

sections 2.4 and 2.5, which includes four initial goals for my research that had been explicitly identified, as well as several hypotheses and conjectures for each goal or combination of goals, before they evolved into the goals this research explicitly reports upon, as reported in Chapter 3.

Chapter 3 starts with section 3.1 looking at technical and non-technical factors in more detail, including references to authors agreeing on the important role of non-technical factors in the success and failure of any software development.

In section 3.2, *organisational politics* are highlighted as one of the non-technical factors that has become the focus of this research. Recognition and modelling of organisational politics are suggested to help requirements engineers to identify the source of power and politics, and possible influences, in organisations. The need to model power relationships in organisations - which are not necessarily the same as the formal hierarchy within that organisation - might help the requirements engineers to find those who will influence the decision makers.

Unfortunately, despite the fact that organisational politics has been mentioned and discussed in the past by academics and practitioners, still it has not found its correct place in industry and academia (section 3.3) but most regrettably in professional bodies e.g., International Requirements Engineering Board (IREB) and the British Computer Society (BCS), section 3.4).

To have high-quality software, the stakeholders' requirements should be fully understood. This needs to be started with identifying the stakeholders, and the degree of power each has to influence the development. All relevant stakeholders need to be identified and considered. They could have different roles (internally/externally) and different feelings towards the project. Section 3.5 reveals that knowing the organisation, all the stakeholders, their roles and relationships with the other stakeholders to be able to manage the situation and prioritise their needs, could be keys to success.

The next section (3.6) discusses the role of Champion, the figure from within the organisation to make sure that all the stakeholders involving in the project are on board have been discussed. Section 3.7 and its sub-sections discusses the four research goals in the light of politics.

Section 3.8 with a help of a further literature review reveals how the idea of a politically aware RE relates to other non-technical factors. In the next sub-section (3.9), a need for a better and tighter definition for requirements engineering has been emphasized and discussed. The four methodology methods adopted for this research are discussed in detail in section 3.10

A new notation, the Political Emoji Notation (PEN) is proposed in chapter 4, which will be the next step to model our finding of the importance of the political power affecting any system development. PEN uses the well-known Emoji faces that can be happy, sad or neutral, to represent users' attitude towards the project, connected by lines to show their dependencies/influence and arrowheads to show the direction of power, that is who has influence over whom. The number of lines shows the level of influence. Emojis representing stakeholders with no connection to others exist but are not involved in the process.

Sad (unhappy) stakeholders could be risky if not dangerous to the success of the project but, when they are identified, can be addressed by finding the reason for them not being happy.

Also, this chapter discusses modelling in software development in general followed by the introduction and discussion of the proposed political Emoji Notation (PEN) model which enables requirements engineers and their team to model power and politics without becoming involved with the organisational politics.

Section 4.1 considers its advantages and compare it with other approaches to modelling; UML and i*. Section 4.2 discusses an important aspect of requirements engineering; that people may change their view of the benefits or disadvantages of one or more system requirements, as a result of instructions from more senior members of the organisation or influences outside the scope of the formal organisational structure.

Examples on how the notation would reflect changing attitudes to a proposed system are given. Section 4.3 talks about the impotency of the inclusion of the emotional side of the political dimension and the confidentiality issue around it. 4.4 concludes and terminates chapter 4 by explaining how one of the adopted methodologies for this research, Impact Evaluation, checking for negative/positive impact of any applied process to the outcome, e.g., applying PEN to any old project as demonstrated in the next chapter.

To test the proposed notation, two case studies of failed projects were considered in chapter 5, to investigate and find the footprint of power and politics and revisit them in the light of PEN to see if it would have made any difference. After a brief introduction into the two chosen scenarios/case studies in section 5.1, the next two sections (5.2 and 5.3) evaluate the PEN notation. The first case study (section 5.2) is the Mandata project. The Australian Public Service Commission (APSC) which is a central agency within the Prime Minister and Cabinet portfolio describes the Mandata project in the chapter 6 of “Towards the end of an era” as:

“MANDATA project—a Service-wide computer-based personnel and establishment records system, intended to supersede the lower-level automated Continuous Record of Personnel, introduction of which had occurred in 1960–61, and the further development of which had been outlined in 1966”.

The second case study, the National Program for IT (NPfIT) in the National Health Service (NHS) by the Department of Health in the hope of improving the quality of the services. The project was meant to connect more than 30,000 GPs to nearly 300 hospitals to have access to 50 million patients’ record. The project ran for nearly a decade before being finally cancelled by the government in September 2011, while time, resources and taxpayer’s money were wasted.

The last section of this chapter (5.4) looks to some symptoms to identify when problems might be arising in a project due to political considerations.

Chapter 6 starts in section 6.1 with a reminder of the initial aim of this research and explaining how it was redirected after a shift in emphasis, and how the organisational politics has become the focal point. The next section, 6.2, The four research questions from section 3.7 were revisited and addressed. Section 6.3 talks about the Political Emoji Notation (PEN) and how it was applied to two well-known failed public sector projects to assess how PEN could have been of value to the practitioners during the project.

Chapter 7 draws conclusion to the research in section 7.1, followed by suggestions for future

work in 7.2, Contribution to knowledge in 7.3. and finally, the chapter finishes by Closing thought in 7.4, followed by the list of references and appendices used for this report.

Chapter 2: Software Engineering, Software Requirements Engineering and Requirements Engineer

2.1 Background

The main focus of this research initially was on the current state of the art in the engineering practice and, specifically, on how and why requirements engineer's role has been changing over the years, in the hope of helping professionals and academics to make sense of the present state of affairs and to face (if not to drive) any future evolution of the job (and the discipline) by relying on some sounder conceptual and interpretative tools.

One important component of my research in its early stages was about depicting the history of requirements engineering, its progress over the years, the way in which such progression could be assessed and understood also in relation to the appearance on the market of tools aiming at supporting and improving the requirements engineer's job. It proved not to be possible to pursue the development of a history of RE without having previously considered the broader perspective offered by software engineering, mainly because the term itself of "requirements engineering" is more recent than its actual practice.

As Alfor and Lawson (1979) highlighted, in order to appreciate how the history of requirements engineering developed before the current name of the discipline became established, it is crucial to remember that many other keywords (on top of "requirements") have been (and still are) used to refer to the discipline and its sources.

Also, whenever history is concerned, I could not excuse myself from referring, in first instance, to the proceedings of the NATO conferences (in 1968 and 1969), because of the crucial role they played in the future development of software engineering as a whole and because they included many valuable hints, references and traces that, along the following decades, would have supported the development of a specific requirements-related area of practice and interest.

Software engineering emerged as a concept over 50 years ago at NATO Garmisch conference in 1968 (Naur and Randell, 1969): since then, the role of other factors was acknowledged as undeniably contributing to the forming of the discipline, for example the separation of software from hardware in 1969, as it started happening in the late 50s and was later described by IBM.

If we refer to the period from 1950s to 1968 as the first generation of computerisation, the main battle at that time was to make systems work with the limited hardware available whilst expecting organisation to adapt to those systems. From the second generation onwards, the challenge was to produce more reliable and adaptable systems, better fitted to the organisation.

With innovation and adaptability (as a requirement) came diversification. Many languages, methods and techniques have been introduced in the last 50 years in attempts to achieve this;

Randell (2018) states that

"I am reluctant to accept that it justifies anywhere near 8945 languages, and the very large number of different methods and techniques that have been created."

A similar scenario, notwithstanding so many efforts have been spent for standardising processes and integrating products, still exists.

Adapting the system to its users and organisations still seems the main challenge nowadays, especially when we fail to identify crucial requirements and validate them against stakeholders' needs, and in last instance we fail to fully identify the purpose of a system.

Hofmann and Lehner (2001) state that

“deficient requirements [is] the single biggest cause of software project failure”, adding that “getting requirements right might be the single most important and difficult part of a software project.”

As Nuseibeh and Easterbrook (2000) claim, practitioners seem to agree that

“software systems requirements engineering (RE) is the process of discovering that purpose, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation.”

Whilst the above definition is commonly accepted, it could be argued such acceptance is due to its vagueness. RE is a relatively recent field and yet, notwithstanding its young age, many attempts have been made in providing it with a more detailed definition, accurate enough to be informative of its core interest and –at the same time- agreeable by all (or most) of its practitioners. A decision to investigate its history seemed therefore ineluctable in order to achieve a clearer picture of the state of the art.

In this chapter, I will try to show the findings of my research as they “evolved” before a significant “shift of focus” towards politics. My research journey started many years ago by investigating the history of software engineering until present.

My main driver, at that time, was Cockburn (2008), who rightly wrote: *“Those who do not learn from history are doomed to repeat it. Let’s stop repeating painful history by following this good, old advice”*.

- **2.1.1 The roots of RE in the (historical) process of Software Development**
“Prevent software from becoming known as softwaste” (Wirth, 2008)

In order to better understand how practice and theory in requirements engineering as applied to software systems and more in general to IT, have evolved up to the current state of affairs, a parallel excursion in the history of software engineering was undertaken.

My literature review started with Charles Babbage, so called “the father of computing” from the early 19th century, whose design and two programmable machines were mostly aiming at solving mathematical problems. The idea that calculating devices could have been “programmed” came from Ada Lovelace, commonly recognised as history’s first computer programmer. When it was envisioned, that programming could eventually “merge” with, and control, a physical mechanical device, that was the birth of some proper system analysis: the same Ada Lovelace used to label herself as an “Analyst” (Toole, 2010).

It is with Charles Babbage and Ada Lovelace that computers gained their own specific identity as “computing machines”: they were inherently general-purpose machines, and therefore different from other devices, such as calculators or other mechanisms (clocks, automatons, toys,

etc.), which computational behaviour was somehow inflexibly embedded within the hardware they came with.

It was a fascinating story for me to learn, although with several gaps, about how modern computer systems have evolved: from mechanical machines to electro-mechanical ones up to electronic computers. The first electronic computer we know about is the ENIAC, developed in 1946 from within the United States Army for calculating artillery firing tables. Since then, computing and computers moved towards society and a commercial computer industry started worldwide in the 1950s: this marked the beginning of a decade of significant improvements in computing technology. However, whilst some might believe that the term “software” was coined by John W. Tukey in 1958 in an article in the American Mathematical Monthly in January of that year, not only there was still no sign of any “software industry”, but also a differentiation between hardware and software had not been displayed until the release of S/360 computer and OS/360, which was marketed by IBM in 1964 as “the world’s first sophisticated operating system”.

Johnson (1998) reported that, well before personal computers, the term software was not in use until the very end of the 1950s, and software was not separated from the hardware, as it was quite common for the software to come free with the hardware. This is obviously not happening today, as customers could buy any software, they need off the shelf (so to say, or from the internet, more likely) for their own personal or business needs. In the first days of computing, though, bespoke software would mostly operate on a single computer in a specific organisation, and it was not possible to use it on any other machinery. This was so until the idea of a family of computers has been introduced by IBM and other actors (like MITS, who launched Altair 8800, and then Apple and Commodore) in the 70s and continued with the concept of Personal Computer (PC) in the 80s, which gave freedom to users.

Interestingly enough, although IBM is known for starting the software industry, that seems not to be the case.

Johnson (1998) writes that

“the entrepreneurs who started their software companies in the 1960s say that the industry was already well-established by June 1969”. In order to support her argument, Johnson states that Larry Welks, from International Computer Programs in Indianapolis, Ind., produced a marketing catalogue listing all available software and their vendors before IBM started charging for software in 1967, which then led to a big profitable business. Nevertheless, Johnson admits that “IBM’s unbundling helped to legitimize the concept of paying for software and was great boom to the growing software industry”.

In those days, the software was the cheapest part of a system: as we all know, now it is likely to be the most expensive part, as if the physical components are becoming less and less important if compared with the conceptual components of the overall solution computers are expected to provide.

No one could imagine what costs or profits would have been derived by separating the software from the hardware. Nievergelt (1968) believed that any prediction about how a market in which selling software would evolve up to a “software explosion” was a very difficult one to make. Since the production of the first commercial computers in the 1950s (mainly for universities and large organisations), the on-going effort to build faster, more affordable computers have

continued. Progress in available technology and the arising perception and expectation that computers could provide their users with powerful tools for solving more and more complex problems as a result have increasingly challenged software developers to design and handle large complex systems.

Although developing software has started becoming more and more a structured activity to be performed within industrial settings in the late 50s and early 60s, and the engineering side of producing software was becoming evident by then, the profession of “software engineer” (or at least the need for such a profession) became somehow officially recognised and “certified” only after the first NATO conference on “Software Engineering” in 1968 in Germany.

At the 40th International Conference on Software Engineering (ICSE), Cameron (2018) reported that it was Margaret Hamilton to name the discipline “software engineering” during Apollo missions. However, Brian Randell, a British computer scientist who was then working with IBM on the OS/360 and was heavily involved in the 1968/69 conferences on Software Engineering as an active member of the editorial team responsible for producing a report of the findings of the first conference (NATO, 1968), remembers that such a term of “software engineering” and the idea of NATO conferences on software engineering in Garmisch, Germany on 7th to 11th October 1968, all originally came from Professor Fritz L. Bauer, a German computer scientist, and from the Study Group that had been set up by the NATO Science Committee in 1967, in the aim of being “deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering” (“Background of Conference”, NATO, 1968).

The conference, held with the participation of about 50 people, all of whom were professionally concerned with software design and development and the seriousness of the existing software problems (including management, reliability, etc.), had a follow-on in 1969 in Rome, Italy.

However, Randell (1996) was not entirely satisfied with the report writing process of the 1969 conference. When compared with its predecessor, he expressed his dissatisfaction by stating that

“unlike the first conference, at which it was fully accepted that the term software engineering expressed a need rather than a reality, in Rome there was already a slight tendency to talk as if the subject already existed. And it became clear during the conference that the organizers had a hidden agenda, namely, that of persuading NATO to fund the setting up of an International Software Engineering Institute”.

In August 1996 at Dagstuhl, Germany, Randell (1996) appreciated the opportunity offered for him to talk about the NATO conferences under a “Memories of the NATO Software Engineering Conference”, which enabled him also to talk about a paper (or rather a “short satire”) called “Masterpiece Engineering”, written by a Tom Simpson of IBM, that did not get into the NATO Software Engineering Conference in first instance because the editorial team “were in the event ‘persuaded’ by the conference organisers to excise this text from the report due to its sarcastic reference to a ‘Masterpiece Engineering Institute’”.

It should be mentioned that the same paper from Simpson had been also published once before in 1974 by Robert William Bemer (another member of the conference) on “Honeywell Computer Journal”.

All details aside, Randell (1996) believes that:

“It was little surprise to any of the participants in the Rome conference that no attempt was made to continue the NATO conference series, but the software engineering bandwagon began to roll as many people started to use the term to describe their work, to my mind often with very little justification.”

It is not so obvious whom Randell (1996) is really blaming for this, but when he says,

“it was little surprise to any of the participants in the Rome Conference”,

it seems likely that he is talking about the organisers rather than just the participants. He did not go into much detail about the actual discussions of the 1969 Conference in this last paper of his. Still, and without getting too shadowed by the dissatisfaction of the author for the 1969 conference report writing or by other “political” reasons, it was by then that software engineering started including some of the features that later on were to be identified and recognised as peculiar to requirements analysis and engineering.

By moving from the arrangements of the two conferences to (the history of) software itself, and back to the development of OS/360, which was labelled as the largest software project for “the most successful computer” in the history of computing (Philipson, 2004), it became important for me to understand how software engineering was expected to deal with requirements within an engineering perspective: late deliveries, not meeting schedules and specifications on large and complex software projects, being over budget along with so many other problems brought to the NATO conference for decision over the future of software development, all these issues were for the first time publicly and openly discussed in relation to the so-called “software crisis”, and the inadequacy of the software development techniques adopted during those days. Haigh (2010), with regards to the software crisis, added that “indeed, it is probably the most widely and extensively discussed event in that history. Software was late, over budget, lacked features, worked inefficiently, and was unreliable. Something to be called “Software Engineering” was proposed as the solution to the crisis.”

Randell (1996) also addressed some of the pre-conference problems (that are well-known in the software industry) such as software pricing, copyright issues and the cost of developing large system as they were revealed and highlighted: all these were issues that affected OS/360, whose success was far from being pre-ordained. Even many engineers within the company argued against it. The machine was rushed into production, and IBM could not handle the demand, its internal accounting and inventory control systems buckling under the strain.

IBM’s chairman at the time, Thomas J. Watson Jr, recounts the story (Watson and Petre, 1990: 349):

“By some miracle hundreds of medium-sized 360s were delivered on time in 1965. But [...] behind the scenes I could see we were losing ground. The quality and performance [...] were below the standards we’d set, and we’d been skipping some of the most rigorous tests [...] everything looked black, black, black. Everybody was pessimistic about the program [...] we were delivering the new machines without the crucial software; customers were forced to use temporary programs much more rudimentary than what we’d promised [...] with billions of dollars of machines already in our backlog, we were telling people they’d have to wait two or three years for computers they needed [...] I panicked.”

Unbundling software by IBM was revolutionary too: software was to be sold separately. More and more companies were able to become part of this new industry, and this came with a huge demand for many requirements engineering tasks and jobs to be performed.

Some time has passed since participants at the NATO conferences started using the term “software engineering”: many of the tasks “engineers” are performing nowadays are those who were traditionally referred as “managerial”, and the technical side of software engineering (together with the same common perception of computers, which are now getting lean and “mobile” as opposed to the huge “boxes” occupying whole rooms and dungeons in usually grey buildings) has moved towards a more user-friendly idea of IT, somehow transforming even the technicality of some tasks (like software development) into a non-technical job, focused more on people than on machines.

Such a paradigm revolution has been mirrored by the academic training of computer scientists and software engineers: in a typical university course in the 60s or 70s, students studying “informatics” were mostly mathematicians with an interest in computers.

Rice and Rosen (1994) explained that

“the undergraduate program evolved initially from very sparse courses offerings in programming to a computer science option in the mathematics department to a separate B.S. degree approved in 1967.”

• 2.1.2 Requirements Engineering: evolution and current status

Along the history of software engineering, requirements engineering (RE) techniques, methods and tools have evolved in parallel with (and benefitted from) a general progress supported and made available by IT and society.

As Alexander (1997) wrote in his essay entitled “A Historical Perspective on Requirements”:
“Several major trends in technology have driven this progress in understanding the place of requirements in development:

- *falling price and increased accessibility of computer-based systems*
- *growing interactivity, bringing a widening range of users with rising expectations*
- *increasing memory and program size, bringing problems of complexity*
- *rising size and cost of system failures despite ever-better development tools*

Any one of these trends could have had a powerful impact on development methodology. Together they have forced system and requirements engineering to transform themselves into full engineering disciplines”.

Routledge (2004) (quoted by Philipson, 2004) noted that, during the first NATO conference in 1968, the term “engineering” marked the beginning of a commonly recognised good practice approach in software engineering built around the adoption of a structured design methodology, and similarly structured techniques and formal design methodologies had been identified and started being applied to the complexity of producing software.

In the NATO conferences, the engineering of the software development process even started echoing a larger “industrialisation” process.

To prevent or overcome the challenges so called ‘problem of scale’, David (quoted in Naur and Randell 1969: 39) believed that these challenges derive because of the increasing size of the

system under development and complexity as a result. By dividing the system into smaller and more manageable chunks and estimation of time spent on the project per annum.

In the same report, Perlis (quoted in Naur and Randell 1969: 39), claimed that “*we must learn how to build software systems with hundreds, possibly thousands of people*”, which would come with extra tasks and challenges that, if not properly performed (e.g., keeping documentation up to date), could then lead to the above-mentioned problems of scale.

More in general, an industrial engineering approach to software development, made up of measurements, metrics and design procedures started to be expected in those years, and even now such an expectation is far from being fully met.

In RE, one the main challenges associated with the above-mentioned expectation is to support final decision makers about the feasibility and sustainability of the development project before (too much) money and effort are committed to it.

Not surprisingly, over the years and as a result of the progress of technology, software increasingly became larger and larger in scale and more complex programs have been produced as a result since.

Already in 1968, practitioners and developers had become aware of an exponential and correlated growth over the years since 1956 of both lines of code and software requirements (McClure, Figure 1).

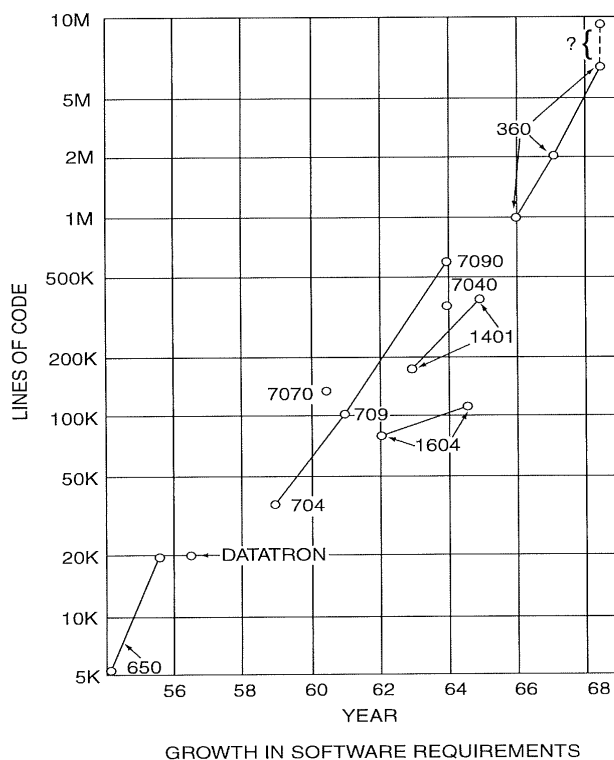


Figure 1: Provided by McClure (Figure 6. from Naur and Randell 1969:38)

Many reasons would have led to such a growth in complexity. More and more often, stakeholders and customers were no longer individual users, but rather large organisations coming with multiple and diverse users, whose different views and needs had to be accommodated by software engineers in their software requirements design and development activities. Among the several stakeholders, the ratio between developers and users has been constantly increasing from potentially one to one (1:1) or even many-to-one (m:1, i.e., many developers for one user!) to one (or many) developers to potentially billions of users (1:m and m:m), as it is the case for our internet-based systems nowadays.

As have seen before, whilst in the first years software developers and end-users were both familiar with the operations and the details of both the goal of the system and how the system worked (in many cases, the end-users were indeed the same persons as the developers: but even if not, they could still be referred to as “technically-minded” users, such as scientists or engineers), later on (and to a greater extent so in our time) the set of stakeholders began to include a mixture of technical and non-technical end-users who needed to collaborate and work on (or towards) the same system without sharing goals, knowledge, expertise, etc. etc.: this resulted in an increasing demand for an engineering approach to requirements analysis and management.

As became evident quite early in the process, determining needs and expectations of non-technical users and different user groups was increasingly a crucial challenge to face, last but not least because users and other stakeholders (sometime? often? always?) did (and do) not know what they actually needed or wanted beyond a “magic wand” (supposedly expected to solve all their problems), without fully appreciating the extent of complexity and potentially the contradictory implications a specific system or solution could come with, whilst impacting not just on their own work or goals, but on those of many other different stakeholders too.

In parallel with (or rather determined by) the pervasive and cheaper computing technology, society itself evolved towards a new and still confused organisational model: work-related environments and traditionally well-bounded professional settings started merging with people’s personal lives. It is commonly accepted that this started when corporations began introducing “personal computers” in their working environments (Quillard et al., 1983). Emailing system could be seen a good example (and heritage) of such a “merging”: emails are part of many people’s day to day activity and often the same email address is used indifferently for work-related and personal reasons.

Requirements engineers (or software engineers who were specialising in handling the requirements life-cycle) needed to focus on several issues related to the challenge of dealing with a higher number of users and user groups: traditional requirements elicitation techniques (such as individual interviews, brainstorming work-groups, questionnaires) were no longer enough for capturing requirements, either because it was physically impossible to do so with all involved users and stakeholders, or, more subtly, because each user and/or user group had different needs or, even worst, was expecting the system to solve similar problems in different contexts and scenarios.

It is possible to appreciate how developing a system flexible enough to adapt or fit its problem-solving power to non-identical situations became a priority to be considered, on top of the other factors that affected requirements analysis and specification, such as those related to the challenge of breaking increasingly complex problems and systems into more manageable “chunks”, for design, implementation and testing/validation purposes. A big question here

would be whether all this can be seen as a RE problem altogether: this issue on the boundaries of RE recurs quite often in this research as it parallels some more fundamental questions about the nature and the definition of the discipline.

Reassuringly, though, any attempt to answer those questions and challenges related to the development of software systems would still imply a need for RE as a practice, to identify, pursue and embed any strategy and technique that would support a suitable (and feasible) management of the development process under time pressure, whilst employing more and more people to work together and trying to predict and optimise costs and efforts required: again, this is not a new issue, as Figure 2 shows.

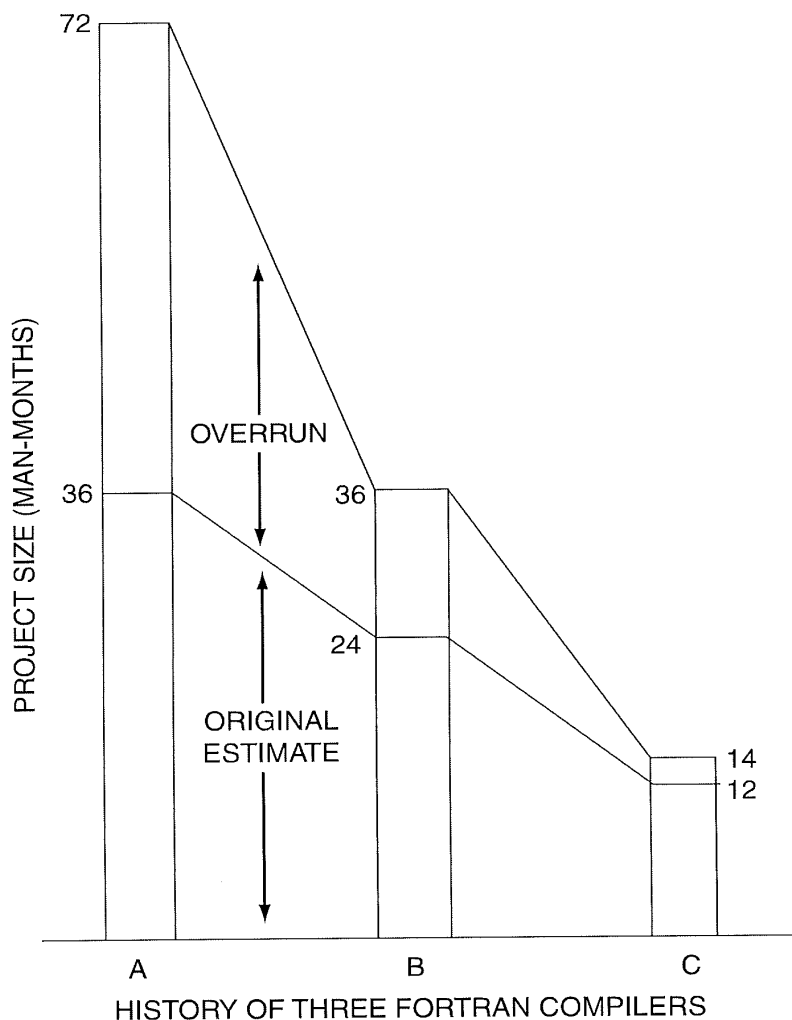


Figure 2: Provided by McClure (Figure 1. from Naur and Randell, 1969:42)

So, to summarise, as software became larger in scale, RE started becoming the “way” by means of which:

- to identify user/user groups,
- to determine user(s) needs,
- to find and hopefully solve conflicts between users,

- to support compromise and hopefully manage the design of software solutions fitting similar problems in non-identical situations and for more and more users and stakeholders, and
- to give designers and managers the conceptual tools for breaking complex problems into more manageable chunks (without necessarily addressing decomposition and code-slicing problems, which usually are better addressed at a later development stage) whilst managing a complex production process, made up of people, systems and quality procedures working in parallel, potentially in a distributed development environment.

The need to support people working in parallel (for example, having different teams of developers working on different modules of the same system), which has been mentioned above, has also become in itself a cause of concern, especially nowadays, when development teams are not necessarily located in a single site or even in the same country. RE has been hugely affected by the possibilities granted by available innovation and technology, and networking teams of developers and stakeholders from all over the world has meant the capability of understanding and accommodating diverse and different representations of requirements within one single and yet larger project. In a way, this has been one of the natural outcomes of the global village prophesied by McLuhan (1962), and it would come with the ability (and the responsibility) of developing global systems across a diverse world.

Over the years, demand for computer and software systems to sell in different markets increased, and this affected the very nature of RE, which was establishing itself predominantly as a technical and formal discipline and practice until the beginning of 1990s, when it started becoming more sensitive to the socio-technical nature of software development.

Again, such a shift is easier to detect in software engineering and the computing industry. From electro-mechanical and the first electronic computers in the 1940s to commercial computers, from a barely visible distinction between hardware and software to the development of the first operating system and programming languages, Fortran and Cobol, from the structured design approaches and the simple methodologies adopted in the 1970s to agile: computing has slowly evolved into and towards the overwhelming socio-economical component of IT.

People have started using computers regardless of their age, gender and education, mostly to take advantage of the social side and opportunities that IT is offering nowadays. Likely, this has led to an unforeseen circumstance: the more people use computers, the broader is the basis of users who are ignorant of how computer systems work and who lack even the minimum appreciation of the work behind the development of software systems.

A negative side-effect of this for requirements and software engineers is that the higher the expectations, the less people were (and are) easily satisfied with reasonably budgeted products: hence the ever-increasing demand of satisfying everyone whilst trying to develop software that could best fit most of our users in terms of usability, accessibility and in general friendly guidance.

It has to be stated that with these challenges came opportunities too: since the late 60s, companies started charging for software, and the unbundling of software (i.e., software that could be sold separately in smaller parts) was a great (if not the greatest) lift to the software industry.

On top of the “obvious” consideration due to functional requirements, non-functional requirements have also become an area that has attracted more and more attention, and their taking into proper account became part of developer’s day-to-day good practice and a major drive in defining RE. Software companies realised that ease of use and reliability, security of customer data and other non-functional requirements were important and valuable assets not just to themselves, but to vendors and clients as well: with this grew legal pressure and liability to ensure that data protection, frauds and hacking were issues to be fully addressed.

Developers learned that, in order to be successful in a highly competitive market, work needed to be more efficiently managed under time pressure and result in higher quality software products, within costs which were still acceptable and satisfactory enough to customers to let them decide to buy it (out of a wide range of alternative products by many competitors), and all this burden slightly moved upon the shoulders of the new-born RE professional practice and discipline.

At the same time, while customers and end-users started becoming more and more demanding, whether willingly or unwillingly developers had to accept (or even to look forward to) having customers increasingly interfering with the development process itself. Not only dealing with non-technical stakeholders, but also exploiting their valuable contributions and encouraging new ways to communicate, still is one of the main and most known duties requirements engineers have to pursue in their daily job.

But when the “social” situation of managing such a complex set of tasks reached a critical threshold, practitioners of RE cried for help: new and better tools, rules and techniques were needed, upon which to rely on.

Many factors induced RE to turn towards technology to support and redefine its own body of practice.

Jarke and Pohl (1994) mention the following issues: “*Coping with flowing changes*”, “*A move from market-driven traditional approaches to customer driven requirement engineering*” and “*Dealing with software design (from how to what)*”

It is interesting to note that, whilst in 1967, (Jirauch 1967) believed that “*the designer must be an experienced programmer with a strong hardware background*”, only after a few years Boehm (1975) thought that system requirements and software requirements clearly marked separate stages prior to the design phase, implicitly assuming that the two set of competencies might well belong to two different professionals, and he moved design and testing closer to software requirements. This is also when he explained that “*the investment in discovery of finding errors at the very beginning and solving these problems are more affordable to comparing with a time spending several times more on fixing errors*”.

Later on, the same Boehm (1976:3) suggested that the line between the analysis and design phases could be very blurry, therefore advocating a new name for the “Analysis and Initial Design” stage. By stating this, he admitted “*the necessity of considering a broad enough interpretation of the word ‘design’ to cover the extremely important activity of software requirements engineering*”.

On top of the above reflections, it seems common sense today, especially once new approaches in software engineering have been established, like agile methods and test-driven development, to admit as desirable (at least) that the designer (and the tester) would be working with the

analyst (obviously, when they were not the same person!) from the very beginning of any development project: hence the intrinsic difficulty in considering requirements engineering and design two completely separated set of activities. Also, it might be worthy to remember how often moving from programming to system analysis is seen as career progression.

All these factors, whilst accumulating and mutually affecting each other, slowly lead to a more dynamic, or “liquid”, definition emerging of RE. One of the first attempts (and one to be less likely to apply to real-world scenarios) was made in 1976, when Boehm in their paper (1976) defined Software Requirements Engineering as “*the discipline for developing a complete, consistent, unambiguous specification—which can serve as a basis for common agreement among all parties concerned—describing what the software product will do (but not how it will do it; this is to be done in the design specification)*”.

Since then, requirement engineering (RE), as a field, has been further rebounded by many authors and practitioners, by means of papers, standards, talks and manifestos. With such a profusion of words, awareness that RE could be not just the most important, but also likely the most difficult step in producing software arose.

Another definition comes from Cheng and Atlee (2007), who believed that RE

“is about defining precisely the problem that the software is to solve (i.e., defining what the software is to do), whereas other SE activities are about defining and refining a proposed software solution”.

The same authors also stated that “*the resulting requirements artefacts have to be understood and usable by domain experts and other stakeholders, who may not be knowledgeable about computing.*”

- **2.1.3 The Requirements Engineer’s role**

As stated above, other keywords than “requirements engineering” (or “requirements” altogether) have been used in the past to refer to RE. Also, not always RE-related specific findings and achievements have been documented properly: all this has made it difficult for me to find the exact starting point for the profession.

In my investigation and literature survey, therefore, many other keywords have been used (extending up to “system analyst”) in order to discover how the role of requirements engineers have changed over the years and the factors affecting it in the hope of learning some lessons to help us plan for the future.

It proved to be a good start to review a definition of the requirements engineer’s role from ModernAnalysts.com and its community forum. In the “Roles” page (ModernAnalyst.com, 2013),

“Requirements Engineer is a term which is often used interchangeably with the IT Business Analyst though many people see this role as being limited to requirements gathering and documentation. The reality is that there are no industry standards for the scope of the requirements engineer. From our experience, this term refers to a role that sits somewhere in between the IT business analyst and systems analyst. The requirements engineer is charged with working with the project stakeholders and end users to elicit, understand, analyse, and document the requirements for a system in order to solve a given business problem. Other common titles for this role are:

Requirements Analyst, Functional Architect, Business Systems Analyst, and Business Analyst (generic term), etc.”

In some circumstances, becoming (or being called or perceived as) a “requirements engineer” could be seen as the outcome of a long and slow transformation process of one’s own job within a certain organisation, given the occurrence of new projects, which respective needs were to be better met from someone who could “focus” as their main (if not sole) task on eliciting, analysing and managing requirements.

Nuseibeh and Easterbrook (2000) quoted Zave (1979) for

“the clearest definitions of RE”: “Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families.”

As previously hinted at, a number of possible causes led to the definition and evolution of the requirements engineer’s role in the IT and the software industry: technology-based new opportunities, more sophisticated organisational models, world-wide socio-cultural changes (e.g., PCs, Internet), a new customer-oriented commercial philosophy and the emerging and increasingly important role of non-technical “stakeholders”.

It is important to underline that requirements engineers had always needed to calibrate (and to manage) their own professional profile against the adopted development process model: sometimes in an indirect manner, others in a more explicit way. Even before the understanding we have nowadays of process models as clearly phased development process, the role of requirements engineers (as well as other roles) could become identifiable and identified once software development was established as a team-based exercise, and this could happen only after it was possible, for each member of a team, to be allocated tasks, goals and responsibilities.

Before that, when developing software systems was mostly a one-person-only activity, suitability of candidates for a given job and quality or expertise of practitioners were considered “as a whole”, and no particular attention was paid on individual skills. This could be seen as a pre-industrial era of software history, and those involved with the “creative work” (Edmonds, 1974) behind the genesis of code-based systems were more likely to be commonly perceived as scientists, artists or artisans rather than engineers, notwithstanding a few open minds and pioneers in the field had already established the real industrial nature of software production.

So, moving towards a phased linear process model, such as the Waterfall, it should have become a necessity to allocate software engineers’ responsibilities on assessing requirements to the first stages of software development: the assumption being that requirements were to be identified, specified and validated before the actual building stages.

Over the years, as demand for larger systems kept increasing, increasing efforts in building faster and more affordable computers and computer-based systems resulted in a general improvement of technology, from process automation systems to medical equipment, from air traffic control to defence systems, from personal records management systems to banking systems.

As it could be expected, the larger in scale software became, the more complex and challenging was handling its production.

Kolence (Naur and Randell, 1969:71), inspiringly, acknowledged that

“I do not like the use of the word ‘crises. It’s a very emotional word. The basic problem is that certain classes of systems are placing demands on us which are beyond our capabilities and our theories and methods of design and production at this time. There are many areas where there is no such thing as a crisis: sort routines, payroll applications, for example. It is large systems that are encountering great difficulties. We should not expect the production of such systems to be easy”. Another author, Ross (Naur and Randell 1969:71), added that “it makes no difference if my legs, arms, brain and digestive tract are in fine working condition if I am at the moment suffering from a heart attack. I am still very much in a crisis”.

Whilst technology evolved (for example, when in the mid-1960s, databases started being conceived, until and since the revolution of relational databases in the 70s, which further impacted software development), more technical skills and competences were expected to be suitably mastered by those in charge for analysing and determining which system had to be developed. The same applies for developing embedded systems: to survive in and successfully face competition with other companies, new development strategies were needed, and this affected any role definition for requirements engineers. They had to be able to act much quicker, to reduce waste of time, resources, etc., and to be innovative (which was usually interpreted as to take full advantage of available technologies), in order to produce better products whilst bringing down the price of the final product.

With globalisation, supported by IT and smart workload distribution solutions, teams started working together within the same development project without sharing the same space: either they collaborated under the same company umbrella or as a partnership between contractors and subcontractors. For the past twenty-some years, multinationals like IBM have been reducing their production and operation costs by opening branches in those countries where they could find the right skills in the right environment at a lower wage. IBM was the first organisation to transform itself into a Globally Integrated Enterprise (GIE). Other companies have adopted a sub-contracting or outsourcing strategy, i.e. to employ smaller companies to do all or part of the development for them at the lower cost, by using offshore services. Either way, as usual, with opportunities come challenges: whilst globalised approaches to development could bring more profits to companies in the shorter term, some nasty side-effects could still occur and get amplified, especially as product maintenance and evolution are concerned.

In other sectors, similar solutions and strategies have been long adopted: for example, we all are aware of customer call-centres delocalised in a number of countries. Whilst this has meant usually some savings in the short term, customers and people have eventually and in many cases outsourcing companies had to think twice about any similar decision made. More in general, communication between people is obviously possible and sometimes even enhanced by the means of IT and the Internet: but lack of face-to-face contact and cultural and linguistic differences do provide barriers to running projects smoothly at a distance, and requirements engineers are nowadays expected to be able to cope with these criticalities, hopefully by avoiding invoicing their employers with travel and time costs, which are sometimes a reason for outsourcing, but which then still appear in the account

In such situations, organisation of people, procedures and resources becomes at the same time of paramount importance and another issue to be properly addressed. In the last ten years, what was a typical, although unfortunately quite expensive, solution (i.e., having a local representative to be able to communicate and collaborate 24/7 with offshore subcontractors) has been replaced by an efficient and low-cost answer: social networking.

The way social networks have evolved has been a surprise to many social networking plays a major role in preparing a good foundation for communications among its networking members regardless of their location as long as they share the same interests and, in a professional project, the same goals. Obviously, many issues, especially in terms of security of information and information exchange within the customer/us relationship, are still open: but it has become largely agreed that, in order to sustain a successful business in a global world, companies should have a structured framework and operative protocol in place, rely on suitable IT and share clear rules and standards for every part of development, which is essential for this type of business model.

Furthermore, all these conditions should be met whilst involving and making of the non-technical user/stakeholder the gravitational centre of the development process, and whilst embracing the challenge to cope with a drive for continuous change and evolution.

Last but not least, mainstream agile process models are not plan-driven: admittedly, their success is due to their ability to respond quickly and effectively to changes driven by customers: it is undoubtable they contributed massively to an urge for re-defining the professional identity of practitioners in RE.

I summarise here the primary factors, as I have identified and discussed so far, that have contributed to the definition and evolution of the requirements engineer role in the software industry, by providing particular components of a broader profile of the practitioner.

1. Developing software for many users

Someone who is expected to cope with inconsistent, incomplete and sometimes contradictory requirements elicited for and by an increasingly high number of users and stakeholders. The mass, as such, is already a contributing factor. Past is the golden age of dealing with requirements specified for one individual user¹ only (a much simpler task)

2. Dealing with different users/user groups

Someone who is expected to deal with diverse users and user groups, for (and/or on behalf of) whom it would be increasingly difficult to identify requirements and resolve conflicts. Elicitation techniques (e.g., online surveys, UX data mining, “games”) have been developed to allow capturing of requirements even when it is not physically possible to do so on an individual basis: but their applicability would still need a patient and expert eye to ensure that representativeness of chosen samples or the interpretation of found data would fit and suit overall system goals and expectations.

3. Different users’ needs

Someone who is able to identify accurately and analyse in detail as many users’ needs as possible, even when (as quite often happens) users do not know what they actually want, which implies for that someone to be able to adopt elicitation techniques, such as throw-away

¹ Such as a senior manager with “dictatorial” powers on staff and systems

prototypes, which might help users in making their minds. Again, this could be another factor to make the role more challenging.

4. Dealing with technical and non-technical end-users

Someone who is able to build a communication bridge between those who are familiar with the operations and goals the system is expected to support and those whose competences are mostly related to how to develop that system.

5. Non-identical situations

Someone who is able to identify, to read and to respond to even barely noticeable differences in similar problems and/or contexts by producing suitable requirements for a system still suitably fitting those non identical usage scenarios David in (Naur and Randell, 1969:39) explained that “one [factor] of increasing importance is the number of different, non-identical situations which the software must fit. Such demands complicate the tasks of software design and implementation, since an individually programmed system for each case is impractical”.

6. Increasing time and financial pressures

Someone who is able to manage human and time resources accordingly with whatever budgets constraints and expectations are in place, by advising on how to tackle problems of large scale and complexity whilst predicting cost and work, by breaking the problem into more manageable chunks, by reckoning which skills and competences are needed to complete the work.

7. Manage more colleagues

Someone who is able to manage and motivate developers, users and other stakeholders, by identifying and valuing skills and enabling people to contribute to their maximum potential. From these last two factors, two questions arise, which my research will try to answer: what is the relationship/difference between requirements engineers and project managers? And how such a relationship/difference changed over time?

8. Distribution all over the world

Someone who is able to make development projects work around the clock, making people from all over the world working effectively and in synchrony, despite their being displaced far away in space, time, different cultures, languages, different degrees and extents of experiences and expectations.

As noted by Nuseibeh and Easterbrook (2000) in their famous “Roadmap” paper, the requirement engineer was and still is usually expected, although with differences that will need to be further discussed, to perform the so-called “context and groundwork activities” (such as understanding the market or assessing risks) for preparation purposes in any software development project. These activities, which were somehow just “preliminaries” of the core set of tasks a requirements engineer was then expected to perform (alongside elicitation, analysis and specification of requirements), have increasingly gained importance, because of the growing interaction between context conditions, customers and development process models.

2.2 The emergence of RE as a recognisable discipline

In 1993 a first RE conference (the International Symposium on RE) took place in California, and another (the International Conference on RE, or ICRE) in Sorrento, Italy, followed by the establishment of the Requirements Engineering Journal in 1996. The merging of the Symposium and ICRE, announced in 2000, was finalised in 2002 with the International

Requirements Engineering Conference (RE), which was then rebranded as the IEEE International Requirements Engineering Conference (RE'03) in 2003: a series of conferences that continues successfully.

Whilst from an academic perspective it has been possible to identify a start date, the same is not as easy from the view of the practitioner.

As Easterbrook (2004) wrote, *“the name “requirements engineering” may seem a little awkward. Both words carry some unfortunate connotations:*

- *‘Requirements’ suggests that there is someone out there doing the ‘requiring’ – a specific customer who knows what she wants*

‘Engineering’ suggests that RE is an engineering discipline in its own right, whereas it is really a fragment of a larger process of engineering software-intensive systems.”

By questioning whether the current discipline could be actually considered as “engineering” (and not rather a hybrid field with a strong “political” component), I share Mahoney’s (2004) doubts:

“is it about the engineering of software? If so, by what criteria or model of engineering?”, but tend to disagree with him when he cites Shaw (1990) in the same paper suggesting that *“Software engineering is not yet a true engineering discipline, but it has the potential to become one.”* Here again we see the temptation of assuming that all factors (including the non-technical ones and the political dimension) could somehow be absorbed or neutralised by an engineering approach, as supported by those who think that “engineers make things work. They apply theories, methods and tools where these are appropriate” (Sommerville, 2007).

As we have seen above, Boehm’s approach to the field (in an attempt to redefine engineering in order to include a non-technical dimension, without “cannibalising” it or denying it altogether) is perhaps a more suitable starting point for discussion. But a healthy dose of cautiousness is needed before agreeing with Nuseibeh and Easterbrook (2000), when they claim that

“the term engineering in RE serves as a reminder that RE is an important part of an engineering process, being the part concerned with anchoring development activities to a real-world problem, so that the appropriateness and cost-effectiveness of the solution can then be analysed. It also refers to the idea that specifications themselves need to be engineered, and RE represents a series of engineering decisions that lead from recognition of a problem to be solved to a detailed specification of that problem.”

2.3 Politics and emotions in Requirements Engineering

All the non-technical aspects of Software Requirements Engineering are critical to consider during software requirements engineering to ensure the project will be successful, but it is much easier to talk about the other non-technical aspects of software requirements engineering like time, finance and cost compared with power and politics.

It is typical for any project to make predictions about the time it will take to finish, create a schedule, and evaluate progress throughout the entire process. The individuals involved in the project must utilize this timeline to strategically plan and oversee the necessary resources in order to meet their objectives regarding time and expenses. Similarly, the budget must also be estimated for each stage of the project, along with the overall expenses, to ensure that the project remains within the predetermined time and cost boundaries.

Atkins (1999) cites two other authors and says “*Olsen (1971) [...] suggested cost, time and quality as the success criteria bundled into the description. Wright (1997) reduces that list and taking the view of a customer, suggests only two parameters are of importance, time and budget.*”

Time and budget are also easier to quantify to compare with power and politics. Generally, other non-technical factors have been better addressed and explored than power and politics. For example, as part of good project management, which Suliman and Kadoda (2017) describe as

“Software Project Management is a core topic in software engineering courses because it teaches how software projects planned, implemented, controlled, monitored, and evaluated.”,

there are techniques to estimate the schedule (time) or budget (cost). Suliman and Kadoda (2017) and Matson et al., (2016) are just two examples among many that propose different techniques for cost estimation. Power and politics can affect everything and also are much softer.

Politics and power are very seldom an exercise in rationality (Writh, 2008) and even less in morality Milne and Maiden (2012): short-sighted self-interest, narcissism, sympathies, and the whole spectrum of human irrational (and sometimes unethical) behaviours are unfortunately more likely to affect politics than any rational analysis or ideal model.

So, whilst in theory politics should be concerned with the fair pursuing of the "common good" of communities and organisations, an emphasis should therefore be placed on the "we all" perspective, in practice it is the “self” perspective, which individuals tend to adopt and give the higher priorities and consideration, creating a sort of epistemological conflict that resonates in any domain of human activity.

However, whilst it is safe to assume it is difficult (if not impossible) to know in detail and likely of no utility to judge whatever unobservable "political" motivations by which stakeholders are driven by alongside the requirements job, there is room for arguing that it is possible to focus on the observable political component of people behaviour, which carries a very visible component humans are usually unable to hid or control: emotions. Emotions (happiness, frustration, anger, fear, love, greed, etc.) can be revealed across many levels of human interactions, from the intentional words used and the consequential decisions made, to body language and up to chemical physiological (Kövecses, 2000).

Whilst writers, psychologists and poets can afford to describe emotions in detail, sometimes by using metaphors and/or accurate descriptions of body and mind states, requirements engineers (need to) value speed and simplicity. Modelling is an attempt to translate accurate (although sometimes subjective) descriptions, which can be developed

by using natural language, into simplified diagrammatic representations of the core abstraction of a problem, a system, a relationship.

A simple way to model any captured emotions is by using emoji pictograms: although some of them are specific to a culture, the vast majority seem of them convey meaning rapidly across a shared universal language (Azuma and Ebner, 2008), by means of which requirements engineers operating world-wide could assess and quickly produce models that come with an extra layer of information, namely the political dimension, representing relationships and other relevant emotional information in an easy way, beyond any international translation difficulty and without the need to introduce any new notation. Chapter 3 will describe my approach and the derived modelling technique more in detail.

2.4 Where this research comes from

Requirements engineering (RE) is a trans-disciplinary field of computing-related practice and applied research. In comparison with other well-known subjects to which the above attribute would apply in an even broader and more challenging sense (for instance, bioinformatics), the boundaries of RE are reasonably contained in quite a homogeneous, technical setting: computer-related engineering.

The term “engineering”, though, has expanded its meaning such in a way that also in ordinary language people end up using it to signify those activities aiming at “engineering” and controlling complex processes, which are well beyond and outside any technical scope or application: for example, “knowledge”. In other contexts, practices that could be better seen as engineering management activities are usually referred to as “engineering” *tout court*: it is as if, eventually, “engineering” has become a term which means more than the sum of all its possible original connotations.

Such a semantic widening seems well represented and applicable to RE and to any of its applicative sub-domains, including the one closer to my own background and field of interest: software requirements engineering (SRE), or RE applied to software design and development. This research project has been to some extent inspired by such a cloud of ambiguity surrounding the field I have been working with for several years, both as an academic and as a practitioner: to understand what RE has become along the years, the role of non-technical factors like power and politics and whether and how such evolution has been determined (and paralleled) by any concurrent development in IT and software engineering.

The very term of “requirements engineering”, as we have noticed earlier, is in itself comparatively recent (Alfor and Lawson, 1979) and, in order to appreciate how its history developed before the current name of the discipline became established, many other keywords (on top of “requirements”) have been (and still are) used to search for relevant information sources.

Later on, in this section, the research aim will be subdivided into a more detailed set of goals and questions: but it is worthy to note that providing an understanding of the change processes affecting RE would retroactively provide even software engineering with a pair of further lenses by means of which to read into its own history and current state of affairs.

Wirth (2008) wrote:

“It is unfortunate that people dealing with computers often have little interest in the history of their subject. As a result, many concepts and ideas are propagated and advertised as being new, which existed decades ago, perhaps under a different terminology. I believe it worthwhile to occasionally spend some time to consider the past and to investigate how terms and concepts originated.”

In the early stage of this research, a first level of complexity was given by an attempt at understanding the necessity (if any) for which such an evolution occurred the way it did. This would imply understanding more in depth, when possible, the rationale (the “why”) for RE to have evolved (and to keep evolving) given a number of factors that have determined which direction computing and software engineering have undertaken so far (e.g., software development process models), as I have highlighted above.

This thesis will therefore briefly refer to how the RE “job” has evolved along time, against a framework of hypotheses and conjectures, it would be shedding some light over the hectic times we are currently living in this industry and hopefully it will help practitioners at identifying and reflecting on future trends according to which RE is evolving, and that has to include a serious refocusing exercise on “politics”.

For my research, I have read, talked with a distinguished academic and practitioner who wished to remain anonymous, further investigated and identified a crucial component that, has been left substantially overlooked in the field and practice of software requirements engineering. I cannot immediately think of another case in which such a large elephant has been left mostly unacknowledged in the room (although, it is fair to claim that some sources have indeed acknowledged it, e.g., Milne and Maiden (2012) and by doing so they provided this research with the backbone it needed).

If, personally and professionally, this has meant a kind of “awakening” in how to interpret and pursue RE, from a more neutral, academic and intellectual point of view I feel confident I have been now in a better position to revise all the previously identified hypotheses, and to view them from a completely new and more insightful perspective.

As mentioned earlier on, one of the aims of this research was about understanding how and why the requirements engineer’s role has been changing over the years, in the hope of helping professionals and academics to make sense of the present state of affairs and to face (if not to drive) any future evolution of the job (and the discipline) by relying on sounder conceptual and interpretative tools and models.

As common expectations would have dictated, such research had originally focused on what logical and technological concepts and achievements in the broader practice of software engineering could have affected (and still affect) the requirements engineer’s role, and nobody denies they have played an important part. However, as will be discussed later on, it was surprising for me to realise that the “technical” component, together with other factors (such as time and memory loss), can be seen as a secondary source of change and in itself the effect of some quieter, deeper force driving what it will be argued seems underlying the “evolution” of RE practice across different cultures and scenarios.

Further down the line of this research, *politics* and *power* have been identified as crucial components of requirements engineering (RE) and it has been argued that the role they play, especially when applied to the software industry, needs to be given greater attention than is currently the case. It has been highlighted how, for several reasons and along the years, practitioners and researchers in RE have left politics substantially overlooked. Whilst many of the sources that have been used here are not recent, I have concluded that the underlying issue is still in need of urgent attention.

Over the last decades, evolving concepts, practices and technologies in the software industry have greatly affected the requirements engineer's role. The "technical" component of such an evolution (supported by broader access to data and information, agile methods, new and more sophisticated tools and techniques, etc.), accompanied by other factors like time and memory loss, is not the only source of change in Requirements Engineering (RE) though, and I agree with Wavell (1982) that a quieter, deeper force seems more powerfully driving the underlying evolution of Requirements Engineering (RE) practice over time: political power.

Power and politics are not a new topic of reflection in software requirements engineering. Bergman et al., (2002), Sadraei et al., (2007), Nelson (2008) has all been inspired by political scientists (e.g., Dahl, 1957), whose definitions tend to converge into the idea that politics is the study of power as it happens and it mainly focuses on the "*process of bargaining and negotiation that is used to overcome conflicts and differences of opinion*" (Nelson, 2008).

It is noticeable how non-technical aspects other than politics and power, such as commercial awareness and viability, finance and project management, have been better received by the practitioners as well as the academic community in software RE. I believe that this is either because they are considered to be more easily translatable into non-functional requirements, or just because it is nowadays accepted and expected that a requirements engineer should have both business knowledge and strong 'soft skills' together with sound IT-related technical skills Gallivan et al., (2004) in order to be able (and enabled) to negotiate with all stakeholders, even with the ones at the top of the pyramid (and negotiation is obviously part of the 'politics that have been referred to).

It still may not seem to be 'politically' too wise to use the word 'politics' in Software RE: politics has become such a difficult issue to handle, as it clashed against the traditional assumption that engineering is and should be, for its vocation and constitution, neutral to the political dimension of human relationships, so that practitioners were never given, nor they asked for, a sound pragmatic answer (or set of answers) on how to deal with politics alongside software development.

In the last instance, it might have been assumed by all parties that an ability to deal with (and sometimes to simply ignore) politics should be considered as a mostly desirable (but totally unstated) part of what we usually refer to as "professional experience" Holmström et al., (2011). This has been mirrored by the fact that the full set of (conceptual) tools available to requirements engineers (such modelling techniques, notations, pragmatic heuristics, protocols and guidelines, etc.) usually does not include anything which they could realistically adopt, outside the speculative environment of academia, for dealing with politics.

So, how to break out of the impasse, given software development is deeply embedded in human factors? Whilst I believe that software construction processes can be, should be and often are well-engineered, software developers and any other stakeholder in the development process are still creative human beings, who need to come to terms with unpredictable, sometimes turbulent, environments, made up of organisational and individual agendas, attitudes, personal interests, sympathies and conflicts, emotions, etc.

Because software engineering has not traditionally fully embraced the political (and indeed the human) dimension of its own nature, it is of little surprise that notations and tools in software engineering, as they are currently available to practitioners, are not suitable to represent politics in any useful (and usable) way. Also, notations derived by other fields, like organisational charts, seem quite biased and unable to capture the actual political dynamics occurring within and under the official structure of any organisation. I believe that this issue could be successfully addressed and resolved by ensuring that, the organisations map against the system that expected to develop, include power and politics in their "too human" and even emotional dimension.

The goal of software engineers is to produce high-quality software. To achieve this, must be ensured to understand customers' needs not fail to meet their requirements/expectations, and they must know what to expect. As Alexander and Maiden (2005) says, "*The System is made for Man, not Man for the System.*"

By understanding customer organisations better, we will be able to identify their problems and to address them accordingly. It has not been uncommon for us to expect customers to incline towards our (software developers') ways of thinking about their organisations and systems, whilst ignoring the internal dynamics of customer organisations. It could be suggested that software requirements can only be agreed on all sides if we understand the way that customer's organisation makes decisions.

I believe that for a long time, non-technical issues have not been addressed, and/or that some crucial factors that might possibly improve RE practice have not yet been effectively addressed. To resolve this issue, a mechanism is needed to capture and model how an organisation actually operates.

A notation described in section 4.1 below has been developed which recognises and documents power, politics and the emotional aspects of software requirements-related decision-making in customer organisations. Siadati et al., (2017) have suggested that

"a simple way to do so is to use emoji pictograms: most of them are part of a universal language, which requirements engineers could easily adopt and exploit to assess and produce models that include an extra layer of "political" information in existing organograms, without the need to actually introduce a radically new notation.

The proposed notation then has been tested on different stages of two failed projects to demonstrate the benefit of the use of this notation."

2.5 Initial Research Goals and Questions

This aim of this research has been decomposed and reassembled in such a way that several measurable goals (and research questions) were derivable. Four goals have been defined and

considered and for each goal (or combination of goals), several hypotheses and conjectures are derivable.

Here is the list of questions (and their underlying hypotheses and conjectures) that had been initially identified:

- 0.1 (Re-) writing a critical and comparative history of RE: where is RE from?**
- 0.2 Identifying a solid and yet flexible definition of RE: what is RE really about?**
- 0.3 Profiling of the professional practitioner: who does the job?**
- 0.4 How RE affects and is affected by which socio-technological factors?**

As a simple definition of hypotheses and conjectures, I referred to Popper (1963) and to Pólya (1977), to whom a conjecture is an unproven (and possibly undecidable) statement (until it could be proven and therefore transformed into a theorem), whereas a hypothesis should be a provable (and testable) statement.

A first list of hypotheses (H) and conjectures (C) are shown here:

H1: it should be possible to name at least one dimension/criterion (and likely more) throughout which to identify and explain relationships between the evolution of RE and the history of computing and software engineering (e.g., development process models)

H1.1: it should be possible to name some features

H1.2: it should be possible to name some processes in RE that parallel (and/or are paralleled by) analogous features and processes in computing and software engineering

C1.3: it should be possible to give evidence that RE, as a set of practices, existed before its actual name, because of the existence of the above features and processes

H2: it should be possible to name at least one dimension/criterion (and likely more) by which to correlate specific RE techniques and some specific approaches and paradigms in software engineering (e.g., object orientation)

C2.1: it should be possible to identify a “core” in RE that has been constant throughout all the changes

C2.2: it should be possible to identify at least one “core need” (and possibly more) outside RE that would lead/drive RE towards some specific historical interpretations, emphasis, perspectives and/or approaches as adopted by leading practices and practitioners

H3: it should be possible to produce evolving profiles (and/or jobs specifications) of practitioners in RE against at least one dimension/criterion (and likely more)

C3.1: it should be possible to give evidence that any given profile (and/or job specification) is correlated to certain socio-technical features in the environment, in the expected product, of the type of client, of the adopted methodology.

H3.2: it should be possible to give evidence that at least one socio-technical factor is currently affecting the way RE is pursued by its practitioners.

A draft table matching hypotheses and conjectures against this first set of research goals is provided in the Table below.

Table 1: Initial hypotheses and conjectures VS research goals

	Goal 2.1	Goal 2.2	Goal 2.3	Goal 2.4
H1	✓			
H1.1	✓	✓		
H1.2	✓	✓		
C1.3	✓	✓		
H2		✓		
C2.1		✓	✓	
C2.2		✓	✓	
H3			✓	
C3.1			✓	✓
H3.2				✓

All the above goals and conjunctures have been reviewed and eventually evolved into a more focused set of research goals and question, as explained in Chapter 3.

Chapter 3: Problem, Goals and Questions

3.1 Technical and non-technical (or socio-technical) factors

Geethalakshmi and Shanmugam (2008), together with many other software engineers, developers and/or authors, have pointed out that the success and failure of any software development project depends not only on technical factors, but also on non-technical or socio-technical factors and/or components. Socio-technical factors have the same amount of influence, if not more, than the technical factors on the success or failure of software development projects.

Despite the concern of Hull et al., (2002) that

“the most common reasons for project failures are not technical”, and even if for many years new techniques (addressing also socio-technical issues) have been suggested in RE, Fricker et al., (2015) state that “many of these techniques did not become industrial practice because they were not practicable or ineffective when used in real-world projects.”

Admittedly, even when new techniques and tools have been introduced and indeed adopted to address alignment between technical and socio-technical problems, in too many cases we are still failing to deliver successful projects or, even more worryingly, to increase the success rate.

Siadati et al., (2019b) observe that, as the project development life cycle starts from system requirements specification, any effect of non-technical or socio-technical factors, whether obvious or hidden, should be actively sought and considered already at this stage to minimise the chance of (propagation and /or amplification of) failure at any later development stage.

Apart from the threats to requirements validity potentially affecting the project as a whole, it seems reasonable to assume that the overall costs for fixing individual problems related to unsolved socio-technical issues at requirements stage would be in line (as potential contributing factors) to the well-known exponential models describing the cost of software bugs: as an example, see report made available from NIST (2002), as reproduced in the following Figure (credits to Sanket, deepsource.io, 2019).

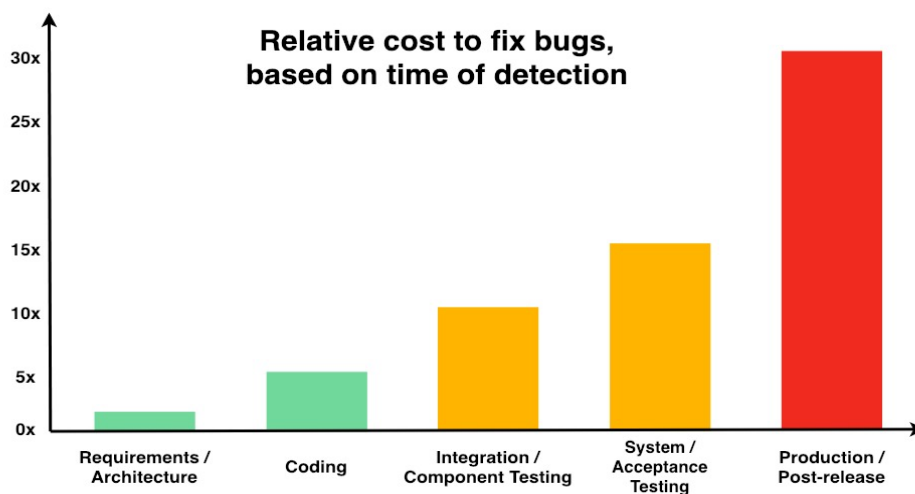


Figure 3: Relative cost to fix bugs, based on time of detection. Credits to deepsource.io, 2019.

Research conducted by the Standish Group (1995; see **Figure 4**), highlighted in detail the main known problem causes affecting the overall quality of RE outcomes. We are not aware of any more recent research arguing that state of the art has changed dramatically since.

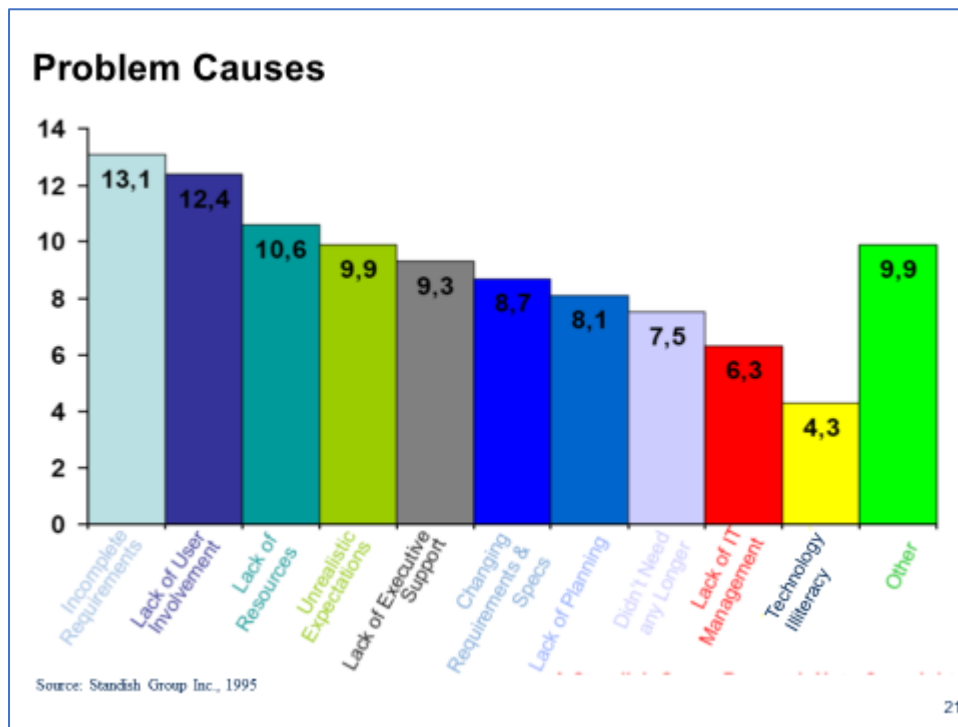


Figure 4: Problem causes (“Project Impaired Factors”), Standish Group, 1995.

As can be seen here, problems relating to or including a major “technical” component (which I identify in the “Lack of IT Management” and “Technology Illiteracy”) account overall for about 10% of identified problems. Even if “Incomplete Requirements” and “Other” are added to the previous two and assumed to be “technical” issues, we barely achieve one third of all problem causes.

The remaining two thirds of problems are of a non-technical or socio-technical nature. This should give rise to concerns that call for a self-critical reflection in requirements engineers: might practitioners in RE sometime overlook something crucial?

In a way, what made it difficult for me to understand how far we were from a full appreciation of the “non-technical component” of this dual technical/non-technical nature of RE was the fact that researchers often only “talk” (and no doubt we do so quite extensively) about this almost mythical non-technical nature.

RE has developed into a discipline where it is now commonly accepted that human, organisational and extra-technical factors play a crucial role, but support for this remains limited.

Wiegers (2000) is one of the few authors who explicitly claims that “*requirements engineering is primarily a communication, not technical, activity*”. According to him, issues related to such a communication activity can begin early on, especially if project participants do not share the same understanding of exactly “what” requirements are.

Either way, even if we have no problems in assuming that the technical component is indeed playing a substantial role in RE, main problems seem still to come from the non-technical part of it, and crucial keys to leveraging this seem to be not properly supported yet. Is it because of a limited self-perception of what the requirements engineer is and the role s/he plays; whether this is a merely technical role or rather a more managerial one, due to an assumption (or fear) that what is beyond the technical dimension is “per se” uncontrollable and so assumed to be of no interest to the requirements engineer? Is it possible that, by doing so, the RE practitioners end up defining themselves as a parallel to the musicians playing on a sinking Titanic?

Answering this dilemma question from a higher philosophical or cultural perspective is beyond the scope of this thesis but helping the discipline to re-adjust its professional identity and trying to support it is not.

3.2 Organisational politics

Organisational politics has been identified as one non-technical or socio-technical factor that has existed for as long as organisations themselves have existed. In Organisational Behaviour (2010), Brandon and Seldman (2004) and Hochwarter et al., (2000) stated that

“organizational politics are informal, unofficial, intentional/unintentional and sometimes behind-the-scenes efforts to sell ideas, influence an organization, increase power, or achieve other targeted objectives”.

It might therefore be beneficial to recognise its importance and document it. An example of this approach is the modelling notation that will be presented in this thesis.

It has been highlighted that little work has been done to date to assist requirements engineers navigate organisational politics to gain acceptance for sets of systems requirements. Milne and Maiden (2012) write that

“although notable work has been undertaken on the importance of social factors in RE, there has been relatively little direct consideration of the influence of power and politics”.

Modelling the actual power relationships in an organisation, as against those formal relationships identified from a traditional organogram, will help the requirement engineer identify those influences which not necessary always comes from the person above. It is possible to have a scenario in which a key influencer is not a powerful person in the formal hierarchy, when influences go beyond the formal to include the informal influences both within and outside formal structures. Knowledge of these situations should assist a requirements engineer to understand how to achieve a solution acceptable to those most able to influence requirement decisions.

In particular, it is vital for a requirements engineer to have this information available when they take a job over from a colleague who is already aware of it, and may be taking into account the need to convince informal as well as formal power-holders of some changes, as needed.

Pinto (1996) believes that

“Power and Politics in Project Management offers you something that is immediately useful in your day-to-day business dealings a better and more sensitive set of warning signals for spotting and reacting appropriately to unseemly political ploys. Pinto presents a combination of theory and practice, laying a foundation of important guiding principles that put project politics in proper context. Once you understand how

pervasive political behaviour is, you will learn to take steps to minimise its potentially negative effects on your projects”.

3.3 Industry and Academia

There is another non-technical factor in which politics would play a role, and this is the traditionally difficult relationship between industry and academia. This results in a gap that both sides complain about. Although Boehm (2007) suggests that

“keeping courses and courseware continually refreshed and up-to-date” would be of help, the reality is somehow less easy to frame and handle. For example, Berry and Lawrence (1998), when focus on the research-practice gap, say that “slowly, we are bridging the gap between requirements engineering research and practice. The gap is still large, but we have a few more practice-validated methods and tools in our pockets, and the bridge building continues”.

On the other hand, Fricker et al., (2015), as cited above, states that many research outcomes did not make into industry because of their lack of practicality or plain ineffectiveness: overall, it seems typical of cases in which it is legitimate to wonder whether some power relationships have consolidated into ideological positions that have their supporters in different fields and are grounded on arguments of a political nature. A more nuanced view of the theory/practice gap is taken by Sadraei et al., (2007), who cite Kaindl et al., (2002) to highlight that

“researchers have continued to develop new techniques and methods, but practitioners have experienced difficulty in applying them [...] the challenge is thus to fill the gap between the research [...] and [...] practitioners.”

Closer to Fricker’s criticism is Bubenko’s (1995), when he writes that

“many of our academic researchers in SE and RE continue to tackle ‘interesting and researchable’ RE problems, often without knowledge of relevant issues and problems in practical life. Our academic education in SE and RE is, in many places, not quite adequate for practical work in the field. There is a need to look at software and requirements engineering education, indeed, as a question of producing engineers instead of computer scientists”.

Alongside the lines of this debate, Wirth (2008) claims that

“we teach, learn, and perform only what is immediately profitable, what is requested by students. In plain words: we focus on what sells”.

However, I feel that Wirth’s analysis is incomplete, as is that of Shaw (2000) when she suggests that

“education for software developers should prepare students differently for different roles, infuse a stronger engineering attitude in curricula, help students stay current in the face of rapid change, and establish credentials that accurately reflect ability”.

Neither explicitly mentions the role of politics as one of the main drivers for producing what makes things “profitable”, as well as “different roles”, “rapid change” and the content of their graduates’ “credentials”.

There is room for negotiating between the most extreme positions here, and this should take into account needs and opportunities by both sides, in an attempt to improve RE in the real world by ensuring that any changes proposed by researchers to the discipline (with new notations, processes etc.) must be usable by practitioners whilst only investing a minimum of training and conceptual readjustment on their part. In an ideal world, one of the criteria of

success of my research would be the adoption of its outcomes in the RE community and any observable improvement this could lead to in RE as an activity.

3.4 Professional Bodies

Another political issue to be addressed in this investigation is that of the role of professional bodies in RE, and the ‘professionalisation’ of the practitioner. Shaw (2000) argues for the professionalization of software engineering (and by extension of software RE) by licensing its practitioners; a trend towards explicit qualification for a role which is increasingly happening in other fields.

Any answer given to this issue is in itself a political statement, depending on where the emphasis has been placed in determining the relevant body of knowledge to be tested in candidates to ‘qualify’ them as practitioners. As it will be observed below (see section 3.7.4) when comparing the two certifications from IREB and BCS, training professionals within a certain given perspective brings with it an implicit political agenda arising from the attitudes of the membership of a particular professional body (cf. Kuhn, 1970). Given that all parties to a negotiation come with their own explicit and/or implicit beliefs about political and power relationships, background and training will in part determine how these qualified practitioners will then interact with stakeholder groups.

Bergman et al., (2002) state that

“the idea that requirements engineering is a political process is already well established, although it is often overlooked in the prescriptive literature”.

They also suggest that failing to acknowledge the political dimension would lead to overall failure, especially in those

“large-scale development [projects that have] remained a high-risk proposition despite huge advances in computing and telecommunications technologies”,

therefore highlighting (again) how the technological component is not the main problem.

From my readings, it is evident that RE professional bodies as “political” entities are not including politics in their training agendas. The questions asked by Shaw (1998) about the need for professionalisation of the software engineer (*“is there a need? yes; are we ready? no”*) are in my view equally applicable to the political aspects in RE, and here they demand different answers: *“is there a need for addressing politics in RE? yes; are we ready? not yet”*.

It is important to recognise that the requirements engineering job is not just “yet another” technical job. I agree with Yang and Liang (2013), as cited by Hoh et al., (2001), when Yang says that

“improving requirements modelling methods, supporting techniques and tools is not enough to deal with the increasing complexity in the RE process, especially in requirements negotiation”.

It is time to acknowledge that a specification document is not only a technical specification, that a representation is not only the outcome of a modelling exercise, and – above all – that any agreement, including the agreement of a set of software requirements, comes with a social and political dimension that must reflect the real human world, whatever the underlying technical achievement.

3.5 The Requirements Engineer and identifying politics in an Organisation

Identifying relevant politics in an organisation is important as other factors in the success or failure of any information system, but it should be recognised and managed in the favour of project. A first step will be identifying the stakeholders. But who are the stakeholders?

One might refer to the stakeholder or user as one single person or group who can provide us with all the necessary requirements, which are to be looked at as the basis for the successful design and development of an information system. However, there is normally more than one user or group of users, and this would usually happen in a broader (e.g., organisational, social, etc.) context in which some have more power than others. Some stakeholders might not be obvious either, and yet all their requirements should still be considered and eventually met by the system. Let us look more closely at stakeholders and users now and try to understand them better.

Knowing and involving all relevant stakeholders is very important: any organisation might (and very likely would) have a number of stakeholders who are reciprocally linked and somehow bounded by (or otherwise interacting with) the organisation itself. Lyytinen and Hirscheim (1987) describe the inability of an IS system to meet specific stakeholder groups expectations as a gap between an existing situation and a desired situation for members of a particular group of stakeholders.

This is further highlighted by Donaldson and Jenkins (2000), for whom system failure analysis grows in parallel with high public expectancy of a certain computer technology to be developed, up to a point in which the fashion or popularity of that system can obscure its basic objectives, whilst varying stakeholder interests create different perceptions of the system.

Groups of users could have different roles and power within an organisation as well as different feeling and intention towards the project. They could be happy, unhappy or neutral. Stakeholders could be in the favour of or against a proposed project: getting to know the organisation and all its stakeholders, with their roles, needs and the relationships among them could be a key to success. One could even wonder whether modelling political relationships overall affecting a certain project would not prove to be more powerful and rewarding than modelling the very products or outcomes the project is “officially” about: in reality, no matter what approach could be adopted, no matter how extremely lean or agile this could be, no one in their health mind would attempt at avoiding a fair understanding of the expected goals and products, which the project aims at. But it seems clearer and clearer that, in many projects, neither that no one bothered to understand (and even less to model) the “non-technical” or “socio-technical” political context in which that project would live its whole life-cycle, nor that, if the “politics” problem could be solved by technology and its many techniques, we should not still have so many unsuccessful projects, of which the London Ambulance Service and BBC abandon £100m digital projects are only two examples.

Although some big management system providers like SAP, IBM, etc. now have learned their lesson and always try, as part of their normal daily practice, to involve most of relevant users and stakeholders and listen to what they want from a certain system in a broader context, not necessarily their consultants and RE practitioners always know (or are allowed to involve and consider) all possible users and stakeholders, such as trade unions, regulators, customer associations, etc.

Without taking into account the relationships among all users involved within a project, we are not able to manage the situation in an exhaustive, comprehensive manner, because, as Ian Alexander and Beus-Dukic (2009) say, “*requirements ultimately begin and end with people – stakeholders*”, and this means that unhappy or negative users or stakeholders could be quite harmful (if not dangerous) to the project.

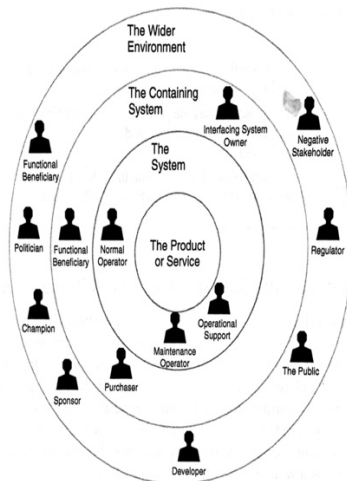
Intercepting (and interpreting) “happiness” (or lack of it) among stakeholders is exactly how we are proposing to address the “political” problem, as we will demonstrate and argue about in the following sections of this thesis: at this stage, it is crucial for us to highlight how identification of stakeholders is potentially in itself a project!

The following figures (Fig. 5), Figure shows a potential list of stakeholders produced for a project by Alexander and Beus-Dukic (2009): all of us could even further add extra roles and stakeholders, keeping in mind specific domains, markets, needs, etc.

- | |
|--|
| <p>Operational roles</p> <ul style="list-style-type: none"> • Normal operations • Maintenance • Support • Helpdesk • Training • Planning, scheduling • Control, management, administration <p>Beneficiaries</p> <ul style="list-style-type: none"> • Functional beneficiary • Social beneficiary, etc (if indirect benefits exist) • Financial beneficiary • Political beneficiary (not only professional politicians) <p>Interfacing roles</p> <ul style="list-style-type: none"> • Owner of interfacing system (sharing data, etc) • Neighbouring business (mutual benefit) • Interoperating service (sharing facilities/equipment) <p>Surrogate roles (representing or working on behalf of others)</p> <ul style="list-style-type: none"> • Champion • Purchaser (roles here vary widely and often overlap) <ul style="list-style-type: none"> - Internal customer - Procurement - Project sponsor - Marketing (representing the consumer) - Product manager • Developer (many roles possible here) <ul style="list-style-type: none"> - Requirements analyst - Designer - User interface designer - Programmer - Tester (very useful in requirements work) - Technical writer • Manufacturer/subcontractor/supplier • Expert/consultant (many more roles possible here) <ul style="list-style-type: none"> - Human factors/ergonomics - Safety engineer - Security engineer - Simulation/modelling expert - Legal opinion - Translator, cultural advisor, etc • Regulator <ul style="list-style-type: none"> - Parliament, statute law - Government departments, regulations - Statutory regulators - Standards bodies (national and international) - Voluntary/industry regulators |
|--|

Figure 5: Possible list of stakeholder roles on projects

Alexander and Beus-Dukic (2009) also suggests the famous ‘Onion’ model of layers of stakeholders, whenever we talk about stakeholder in a typical system (Fig. 6).



Hybrid roles

- User (= Normal Operator + Functional Beneficiary)
- Consumer (= Mass-market Purchaser + User)
- Service Provider (= Operational Support + Maintenance + Helpdesk, and sometimes Developer, Subcontractor, Manufacturer)
- Risk and Revenue-Sharing Partner (= Developer + Manufacturer + Financial Beneficiary), etc

Negative/hostile roles

- Vandal, graffiti artist
- Thief
- Hacker, virus writer
- Competitor
- Industrial espionage (via malware, etc)
- Fraudster
- Disgruntled employee
- Trades union
- Political opponent
- The public, residents' association, etc
- Activist, environmental pressure group, etc
- Military enemy, terrorist

Figure 6: The "Onion" model of stakeholders in a typical system.

When looking at the roles listed in Figure 3, it should be safe to admit that not all these roles normally are involved/considered by requirements engineer when gathering information and that some would likely never appear in the technical documentation. For example, a trade union can be actively invited, listened to and have influence in a project, but it is unlikely it would ever appear “as such” on any UML diagram. By not considering any user or user group, not knowing their needs and especially if they are “politically powerful”, we would end up creating more and more “unhappy stakeholders”: a problematic circumstance to handle, as we will discuss later when introducing our proposed emoji-based modelling approach.

Consider the London Ambulance Service system briefly: its “development history” is often referred to as an example of failure in IT, with a first failed attempt between 1987 and 1990, followed by a second, thunderous crashing in 1992, made up to two major episodes, one in a few hours after being made operative and the second, in about nine days, reverting the overall operations of the London Ambulance Service “back to where it was prior to the 1992 system” (Fitzgerald and Russo (2005)). It became newspapers headline as a result of over 20 people died as a result.

Page and Boyd (1993), after a public inquiry report on the failure, claim that

“the London Ambulance Service (LAS) management put price before quality and committed to an over ambitious project timetable. This was evidenced by the selection of a supplier who has no experience in building ambulance dispatch systems but had significantly underbid a more established supplier. This was made worse by the management putting the supplier under immense pressure to deliver the system quickly”.

If we were to think of those stakeholders cited in the above few lines, we could easily imagine many “unhappy faces”, when for example due to major reform within the LAS, more than 210 managers lost their job, or when the control room staff had their training 10 months before the start date, and eventually “found the system to be too complicated and did not trust the motives behind implementing a computerized system” (Ukessay (2018)). The system was reported to have 81 bugs just before going live and trade unions were not involved. All these and many more issue show that the system also failed politically and socially, within and outside the actual organisation it was supposed to help. A summary of the report can be also accessed

online, courtesy of Berry and Lawrence (1998). “*The supplier was not solely to blame*”, it was suggested by Boehm (2007):

“Management and routine maintenance of the trust’s IT systems were found to be insufficient, with a lack of oversight at board level. Since the incident, LAS has appointed a chief information officer and a non-executive director with extensive experience in IT has joined the board. A comprehensive plan is now in place that includes improvements to governance and performance management of IT.”

Despite all the effort and investment to upgrade, the LAS system failed again on the New Year’s Eve on 2017. Bano et al., (2018) believe that having multiple teams of stakeholders with different levels of power in decision-making, the politics is inevitable and inescapable.

Without paying careful attention, the political aspects of user involvement in software development can significantly contribute to result in unsuccessful projects.

3.6 Support/Champion

Some stakeholders can carry explicit political (or otherwise decisional) power: this could be given by the actual role they play within or upon an entire organisation or on a specific team or board, by the ability to attract consensus and/or to grant financial support or even by other, less immediately detectable factors and interests. But only one type of stakeholders is invested with a special, “authoritative” power, by which they could inspire individuals, redirect focus on existing or new objectives, inform and drive projects with in-depth domain-specific knowledge, and this independently from the actual place they might occupy in an organisational chart: the so-called champions.

Champions could well be the recipient of other categories of power, but it is in their specific capability as champions that they are enabled to drive the project and ensure all people involved are fully focus and tuned towards the achievement of certain objectives. In this way, a champion can exercise quietly a very powerful power, without being somehow hampered by the burden of carrying any official “political” responsibility attached to it. Entrepreneurs like Mark Zuckerberg, Elon Musk, Bill Gates, Steve Jobs, Larry Page and Sergey Brin, just to name a few in the IT world, have managed to develop a “champion” profile within their organisations, so to almost hide the power they carry as owners of the business.

In many other cases, a champion is someone who has been identified as the enthusiastic, committed “expert” acting as the main contact point for those who are expected to support the project and as a mediating voice towards those whose work would allow the project to be developed and, in last instance, successfully implemented: a simple extra “appointment” to the list of stakeholders, smart enough to secure benefits for the projects they are called to lead without necessarily modifying or disturbing other balances within an organisation. It is important to note that a certain stakeholder can well be a positive influencer, but not necessarily a champion: only the champion, by denying any “official” political power, can de facto playing a crucial political role.

But as Young and Jordan (2008) state, there is

“evidence that top management support is the most important critical success factor for project success and is not simply one of many factors. The finding is justified in the context of the project management literature and the IS factor

research on project success. There are implications for practice because it appears that the conventional technical and project management advice has less impact on project success than previously thought. Boards and top managers may have to personally accept that they have more influence on whether a project succeeds or fails”.

Emphasis on organisations and senior managers’ responsibilities is also placed by other authors, who suggest “that IS projects fail more because of organisational than technical issues” (Lucas, 1975; Sauer, 1999). It has been denounced that the majority of IS researchers and practitioners, in an attempt to improve efficiency and project success rate, still “persist with technological or engineering conceptions of the problem” (Currie et al., 1999) and, by doing so, repeat the same mistakes every year (Collins, 1997; Young, 2006).

3.7 Research questions and goals (plus why notation)

Notwithstanding several authors have recognised the conceptual key of "politics" as an important component in any Requirements Engineering (RE) process, practitioners still lack a pragmatic answer on how to deal with the political dimension: such an ability has become a mostly desirable but undeveloped part of what we usually and vaguely refer to as "professional experience". Nor were practitioners given any suitable tool or method to easily detect, represent and, if possible, leverage politics.

Main goal of my research is to provide practitioner with a conceptual technique enabling them to easily manage the political dimension during their requirements job, on the basis of a critical understanding about what and how politics is and can be represented within software development projects. In order to do so it has become paramount to review and attempt an answer to the following questions.

- **3.7.1 Why and how to focus on RE: politics as the conceptual key**

Requirements engineering is one of the first work phases/steps (if not the first) where to define goals and the outcomes of any software development project, and it has been argued (e.g., Finkelstein, 1994; Maccari, 1999; Gebauer et al., 2008) that this stage is essential for any other following one.

The main actor at this stage of software development is the requirements engineer, who – as part of their job – needs to learn about clients, their organisation and their needs, and to map them against their goals. Any proposed solution, which relevant parties (or stakeholders) expect to specify, design and develop, needs to be fully analysed and validated against a number of constraints (finance, technical feasibility, time, etc.). The above exercise is twofold, as it implies both highly technical skills and competences, and an ability to negotiate across relevant parties: failing to comply with both these technical and “diplomatic” tasks would increase the risk of project failure.

A traditional response in the past has been to increase the efforts in pursuing and managing RE-related activities, and we should all now be more aware than ever that these activities are far from being optimally performed (see Figure 7). Yet, they still compound of the most critical mass in ensuring a project is successful.

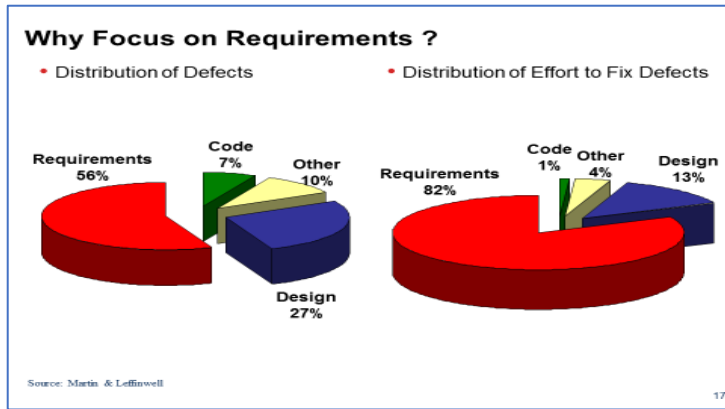


Figure 7: Why Focus on Requirements

Along the decades, several approaches and methods have emerged to give clients and practitioners easy answers to complex problems: whilst it would sound ungenerous to classify agile methods as one of those answers, it is widely accepted that focus on short interactions and user involvement have been representing, more and more, a simple and effective good practice in modern software development.

The conceptual framework within which I am hereby proposing to better understand the nature and the consequence of the above practice is what I refer to as politics (or the political dimension) in RE. Politics in RE can be defined by several forms and meanings, but all relate, in last instance, to the roles people play within any software development project from its early stages. It has been a long process for me to eventually recognise, focus and commit my research on politics in RE: somehow I had to wait for a deconstruction process to be completed in my own personal and professional understanding of RE, seriously reviewing years and years of education, research and practice in the field based on the assumption that a “technology-oriented and driven” approach (as expected by software engineers and technology enthusiasts) was, in last instance, what really matters in RE. And of course, like most of academics and practitioners, I really was aware of the role any socio-technical component played in software development and IT-based projects. And yet the political dimension was just something blurred existing in the background, something one would consider only accidentally, if and when, and not as a real, primary need. I believe I simply could not acknowledge the importance of politics without letting the word “need” sink deeply in my mind from the crucial contribution from Alexander and Maiden (2005), when he wrote that *“projects need to know the roles involved, and the viewpoints of the stakeholders playing those roles”*.

I would argue that the term “deconstruction” (although not used as Derrida would have), which I adopted to describe the process I had to undertake in order to fully realise the importance of the political dimension in RE, is not inappropriate: like for entire generations of students from IT and CS, who then started practicing as software engineers, requirements engineers and then as academics teaching those subjects, with an emphasis on the technical component of the job, such an achievement had been somehow obstructed in many ways by the dominant literature.

I report two examples here, which I consider symptomatic of the dominant technophile approach to RE. The first case is from Sommerville and Sawyer (1997), who consider the cost to identify too many viewpoints when they write that *“requirements engineers must balance the advantages of wider coverage offered by additional viewpoints against the costs of analysis and the problems of information management”*. Given the problems shown in Figure 5 above,

it is useful to compare costs with costs, namely those costs necessary to gain and manage additional viewpoints with those emerging from not doing so: failing to make sense and value different viewpoints would seriously undermine the same kernel of the requirements engineer's job, which is – in the last instance – to understand the purpose of a system and the activities it will be used for.

A second case is given by Easterbrook (2004), who acknowledges the intrinsic challengingness of such a job and still somehow seems lean to suggest it could be only a matter of applying a number of “techniques”, when he writes that

“such human activities may be complex, because they generally involve many different types of people, with conflicting interests. In such situations it can be hard to decide exactly which problem should be tackled, and it can be hard to reach agreement among the stakeholders. Requirements Engineering techniques offer ways of dealing with complexity, by systematically breaking down complex problems into simpler ones, so that we can understand them better”.

I think Easterbrook did not mean any “reductionism”² here: but his words might have offered in the past an easy alibi not to see the “non-technical aspect” through to the bitter end and an even easier selling point to requirements engineer's job, if nothing else because of some shamanic-like ways society tends to look at science, technology and technicality nowadays: as yet another magic manner to heal and solve all problems (for a more comprehensive and eye-opening debate on this, see Stivers and Stirk (2001)). This is clearly a trick advertiser keep using to sell us toothpaste, shampoos and baby nappies: to dress actors as scientists!

So, notwithstanding the above, a specific component of this non-technical aspect has become the key conceptual finding of this research, and because the talk is mainly about people roles and activities, there are two terms which properly represent it from different and yet converging perspectives: politics and power. Focusing on RE suddenly means we cannot avoid any longer the burden of considering and understanding politics and power and how they affect both RE practice and the practitioner's self-awareness.

If software requirements engineers and analysts are to really take advantage of any insight they could gain on politics within organisations (for example, who *de facto* decides which priorities come first when a conflict arises between two departments in the same company), a more appropriate (but unfortunately less easy) approach is demanded (and would demand) more efforts in capturing political dynamics and their source(s) behind the official organisational structure.

Such an approach could be translated into a number of “non-technical” tasks, such as:

1. Mapping organisational politics, i.e., the set of power relations occurring across (and despite) the official organisation chart;

² I am paraphrasing the definition of reductionism provided by the Merriam-Webster's Encyclopaedia of World Religions (Doniger, 1999), according to which “theoretical entities of particular sciences, such as biology or psychology, are definable in terms of those of some more basic science, such as physics”, as if the complexity of managing all aspects of requirements engineering (including the so called “human factor”) could be reduced to some modelling or procedural exercise of any engineering nature.

2. Mapping the political dimension of finance, i.e., what drives decisions behind the identification and allocation of time, budget, skills and resources;
3. Mapping the politics of the technological dimension, by enquiring who makes decisions about what to develop, by using which development technologies, and why.

And yet, somehow, politics as such can be seen as a fourth, more fundamental dimension informing the above three tasks.

Even Bubenko, someone who has investigated in detail the non-technical component of the engineer's job, seems reluctant to mention explicitly the political dimension. In one of his most famous papers (Bubenko, 1995), he states that *"RE includes managerial, organisational, economic, social, as well as technical issues and problems"*, and later he writes that *"Existing methods are not adequate for explicit capturing, and representing, in a structured way, "business and organisational knowledge" to be subsequently used to drive the information system development phases"*, which probably includes what we mean by organisational politics. But the keyword is missing, and so another opportunity to consider in further depth this aspect could have been lost.

As another example, Nuseibeh and Easterbrook (2000) indicates that *"RE needs to be sensitive to how people perceive and understand the world around them, how they interact, and how the sociology of the workplace affects their actions"*, which is clear evidence of what I meant before: the elephant is there in the room, but somehow, for some reason, requirements engineers often seem not to be able or willing enough to point at it.

And then (in the highly arguable assumption there is anything before power) there is power to be taken into account.

Dahl (1957) writes that

"if so many people at so many different times have felt the need to attach the label 'power', or something like it, to some Thing they believe they observed, one is tempted to suppose that the Thing must exist; and not only exist but exist in the form capable of being studied more or less systematically. [...] That some people have more power than others is one of the most palpable facts of human existence [...] power is a relation [...] among people [...] The main problem, however, is not to determine the existence of power but to make comparisons.[...] But what, precisely, do we mean? Evidently, we need to define the concepts 'more power than', 'less power than', and 'equal power'".

The definition of power as an influence and politics as a process of demonstration of power is agreeable, as indicated by Milne and Maiden (2012) who believe that *"if power is defined in terms of a potential force, then politics can be regarded simply as the study of power in action or alternatively in terms of a 'process of bargaining and negotiation that is used to overcome conflicts and differences of opinion'. Power and politics have not been adequately considered"*.

As a matter of fact, they are indeed quite far away from being adequately considered: even when the importance of politics and power is rightly stated, the temptation to reduce it to an engineering problem or just to over-simplify things is strong, whilst performing what is, in last instance, an incorrect political analysis.

Nelson (2008), for example, in his “The Politics of Agile” explains politics as follows:

“Politics is the process by which groups of people make decisions. In business, the process for building products requires hundreds of decisions every week. The challenge lies in who is involved, when do the decisions need to be made, how are decisions made, and who has the final authority. What is the real problem? Ultimately, we all want successful products that help us all make lots of money”.

This is quite clearly a case of naivety and –possibly– of weak political analysis, as it is assumed that everyone has the same agenda.

Only a relatively few authors have explicitly addressed the issue.

Maiden (2012), whilst pointing to the topic of politics, warns that

“the concept of ‘Politics’ is a phenomenon that we perceive to affect our work. However, our failure to define it puts us at a disadvantage, because it keeps us from developing strategies to respond to it in our requirements work”.

Maiden continues that “*modelling power relations between stakeholders should become a norm*”. This is a statement which I would subscribe to, but only after bearing in mind that talking about politics in any organisations can be quite a frightening exercise and RE practitioners might not be given the right skills and tools to do so appropriately and effectively.

Milne and Maiden (2012) cite Kentar (1979) who admitted that “*it is easier to talk about money and much easier to talk about sex – than it is to talk about power*”. This kind of taboo has affected and seems evident in the RE literature, where “*although notable work has been undertaken on the importance of social factors in RE, there has been relatively little direct consideration of the influence of power and politics*” (Milne and Maiden, 2012).

- **3.7.2 Where is RE from? Politics and power underlying human activities**

Whilst many research projects rely on some specific theoretical conditions and criticalities which inevitably limit the task of suggesting and further investigating new and yet very circumstantial hypotheses, others of more general nature, including the current study, need to be grounded on a broader set of circumstances, which are affected by and somehow embrace the full history of their field of interest.

This is certainly the case for software requirements engineering, in which the concept of history has always been a constant nerve to touch, starting from the very beginning of the so-called “computer era” (Wirth, 2008), on which consideration several authors have spent their efforts. Nievergelt (1968) claimed that

“it is necessary to look into the past because we have not yet had time to assess it and draw final conclusions from past experience. Yesterday’s ideas, out of fashion today, may well come back to prominence tomorrow”.

Boehm (2006) thought that “George Santayana’s statement, “*those who cannot remember the past are condemned to repeat it*”, is only half true. The past also includes successful histories. If you have not been made aware of them, you’re often condemned not to repeat their successes. Extensive studies of many software projects such as the Standish Reports offer convincing evidence that many projects fail to repeat the past successes”.

And yet, it is interesting to underline how difficult it is to define “the past” and its trends, no matter how recent these might be, and, even if we were able to track benchmark episodes, how easy it is then to fall down into an over-simplifying, naïve interpretation of facts.

A typical example of this could be that, after the first electronic computer, the ENIAC, was developed in 1946 by the United States Army for calculating artillery firing tables, computing and computers have since moved towards society and a commercial computer industry started worldwide in the 1950s. How did this happen?

Was there growing commercial demand?

Or maybe there was pressure from the cold war to improve quality of IT systems for the military and the government?

Was computing moving towards society, or was maybe a general shift happening in society, which somehow affected and determined how both the army and the computing industry became more open to society?

In other words, were not the political factors that could be overwhelming the accidental occurrence of progress in science and technology? And this is just the surface:

were we able to further dig, what kind of “politics” happened in practice?

Who started moving the mechanism?

Who exercised their power to let this happen the way it happened?

And the same kind of retrospective critical evaluation could be applied to academia.

When Wirth (2008) used to talk about the “computer era”, would not his argumentation be again based upon a set of naïve assumptions that simply neglected how politics and power have been the most influential drivers in the shift occurring in computing and IT, from academic labs and researchers to ordinary public and customers?

And what role would the political arena and actual politics and power (including the micro-decision people in charge at different levels made) played in dictating how, in the last instance, software would have needed to be understood, designed and managed the way it did outside the academic fences?

It could be underlined here that those overall political dynamics go well beyond any kind of “conspiracy theory”: it is hard to believe that it was never assumed that someone in particular (or some group of people) behaved as a “big brother” who plotted and controlled whatever historical trends could be identified in software and system engineering: likely, this would have been the unpredictable emergence of a huge, complex political game, in which all relevant parties (from the very big actors on the market to large masses of individual consumers, each coming with their own agendas and their little load of power to spend) eventually contributed to the success or the failure of that project, to the rising or the decline of that process model, to the influence to be acknowledged to that approach, to the usage or the dismissal of that technology.

But when this political game occurs at a lower level and within smaller environments and organisations, as it usually is assumed to be the case for “ordinary” requirements engineering jobs and projects, should not practitioners pay attention and try to understand its dynamics, in order to ensure that any functional (and non-functional) software requirement would in the last instance be truly “functional” (or at least not dysfunctional) to that organisation, within that certain job or project?

There is a long-standing issue, known since the Sixties as the “software crisis”, a term definitely used during the first NATO Software Engineering Conference in 1968 (but likely coined before) to mean how increasingly difficult it was to ensure the achievement and the deployment of suitable solutions to increasingly complex problems by relying on more and more powerful computers and IT solutions within evolving organisations and socio-technical structures. Haigh (2010) reports that attendees, *“having never met before, were shocked to realise that identical troubles plagued many different kinds of software and agreed that a ‘software crisis was raging”*.

Wirth (2008) believes that in the first conference sponsored by NATO in 1968 *“the difficulties and pitfalls of designing complex systems were frankly discussed”*. Maybe this is the case: but from a study of the 1968 Garmisch conference’s proceedings, power and politics did not emerge as topics of the highest priority. This did not prevent the NATO conferences (in 1968 and 1969) and their proceedings from making a crucial contribution to the development of software engineering as a whole, evidently: but then, would this not make it arguable that somehow, they come with their responsibility for the current status of neglecting politics and power in our ordinary practice of software requirements engineering?

Whether or not that is the case, some authors reported unsolved issues with RE from its very beginning. Van Lamsweerde (2000), for example, considering what happened since 1975, wrote that *“software requirements have been repeatedly recognised during the past 25 years to be a real problem”*. Possibly, this meant the opportunity for RE to be recognised as a specific body of knowledge and practice, which needed to have its own dedicated spaces of discussion and reflection.

Mead (2013) confirms that *“in the early 1990s it was recognised that requirements engineering was becoming a distinct discipline. The initial idea for a requirements engineering conference occurred within the software engineering community”*. In the meantime, as I already noted in section 2.1.1, Alexander (1997) had noticed that several major trends in technology have driven this progress in understanding the place of requirements in development:

- *falling prices and increased availability of computer-based systems*
- *growing interactivity, bringing a widening range of users with rising expectations*
- *increasing memory and program size, bringing problems of complexity*
- *rising size and cost of system failures despite ever-better development tools*

More generally, it is now time for practitioners and academics not only to realise that politics and power need to be considered and properly addressed, if we are to make full sense of what Easterbrook (2004) claimed, when he wrote that *“the real goal of an engineering process is to improve human activities in some way, rather than to build some technological artefact”* (and, by saying so, contradicting Sommerville), but also for them to consider whether they should not completely redefine RE itself.

- **3.7.3 What is RE really about? Politics and power in action: the subtle art of negotiation**

Many authors have attempted a definition of requirements engineering or at least of its application field but it is appropriate to revisit Cheng and Atlee (2007) who believes that

“requirements engineering is about defining precisely the problem that the software is to solve (i.e., defining what the software is to do), whereas other SE activities are about defining and refining a proposed software solution”.

Whilst most of us might substantially agree with the above definition, it could be argued that, by doing so, it could also be agreed to include within the semantics of the word “engineering” something which has nothing to do with the technical component that is usually associated with it: because “the” problem software is to solve within organisations and complex environments is in the last instance not (only) an engineering problem, at least not in the same way in which the same organisations would likely look at other assets they might want to “engineer”: buildings, products, money and any other “physical” entity they might like to attain and/or accumulate.

By means of software systems, organisations are trying to set up not just “physical” repositories of information and knowledge: by channelling information by means of software systems, organisations try to establish and/or control networks of power, to alter balances of forces, to ensure that knowledge and information can be easily translated into political currency for better managing dynamics between internal and external stakeholders.

It might be appropriate to quote Berry and Lawrence (1998), when they cited Brooks (1995) that *“the hardest single part of building a software system is deciding precisely what to build”* and continue with the definition by Boehm (1981), as cited by Sutcliffe (2002), that requirements are *“designing the right thing”* as opposed to software engineering, which is *“designing the thing right”*. Sutcliffe in the same book explains nicely how requirements are *“inseparable from us”*, whenever the design of software systems take place, across the four traditional phases the RE process is usually divided into: requirements discovery, requirements analysis, requirements negotiation and requirements specification.

As already mentioned, Wiegers (2000) is one of the few authors who is radical about his claim that *“requirements engineering is primarily a communication, not technical, activity”*, and that *“communication problems can begin early on if project participants have different ideas of exactly what requirements are”*.

An interesting and slightly different viewpoint is offered by Fricker et al., (2015), who focus on success measurements for RE against quality criteria, which include quite explicit references to what is hinted as politics and power; the strategy, ability and willingness to make changes, willingness to defend solutions, relationships to users and between stakeholders.

Table 2: Success measurements for RE including and/or referring to politics (derived from Fricker et al, 2015)

Quality of RE service	Quality of RE products
Business-technical alignment: fit with strategy, ability and willingness to make business changes, and management support.	Quality of cost-benefits analysis: completeness and coverage of cost-benefit analysis, new benefits created by the new solution, and sufficient accuracy of cost estimates.
Stakeholder acceptance: awareness of business changes, extend of consensus, willingness to defend solution, and relationship to users.	Argumentation of impact: diagnosis of existing solution, traceability of supported processes to problem to be solved and to system goals, and traceability of strengths and weaknesses of new solution to replaced solution.

The indirect definition they provide of RE is derived by “*the quality of RE service, [which] refers to the effects a requirements engineer wants to achieve, and the quality of requirements engineering products, [which] refers to the work results delivered by the requirements engineer*”.

Mentioning and including users in any definition of or related to RE is always tricky, because of the extent of uncertainty and ambiguity they usually bring with them. Complaining about users would be as meaningless for a requirements engineer as for doctors to blame their patients for making their job so challenging. But whilst doctors can nowadays rely on clinical and biochemical tests and scans to better diagnose their patients' conditions, requirements engineers still have to rely 100% on what users and stakeholders tell them and on observing what they do. And to be fair, whilst patients try to refer what they “have” or are experiencing in terms of symptoms (which supposedly could be quite a straightforward task), users and/or customers have a more difficult and abstract goal to achieve in telling what they “would want”.

The funny and paradoxical comic by Scott Adams in the following figure (Fig. 8) depicts this scenario much better than what it is possible to do in one thousand words.

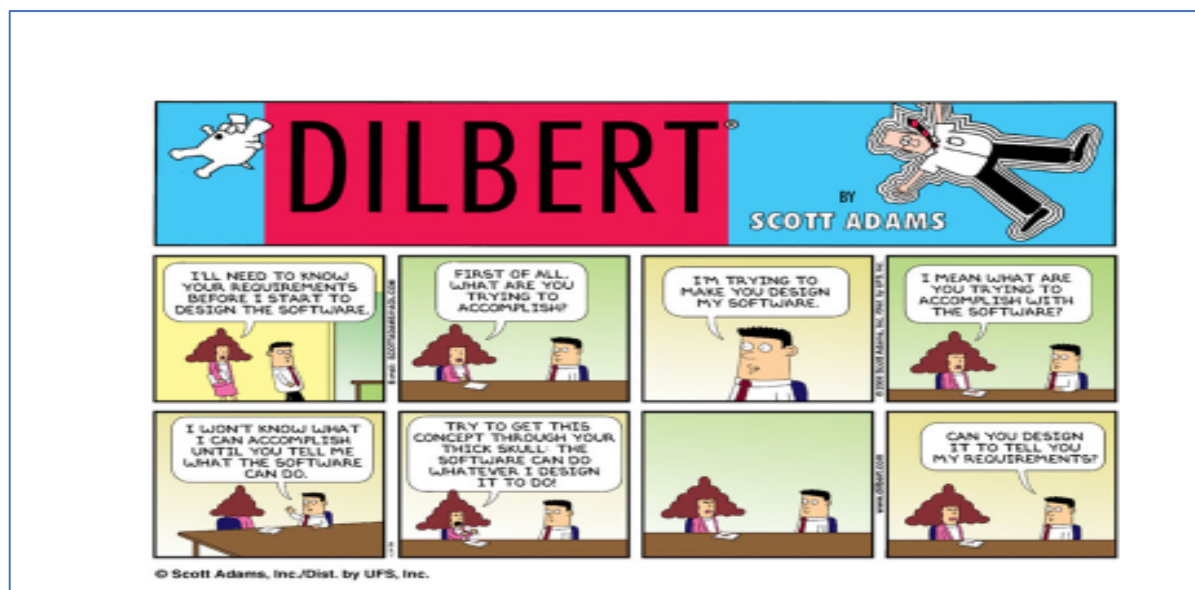


Figure 8: What do really users want?

Whilst Berry and Lawrence (1998) think that *“requirements are inherently incomplete mainly because they are never fully understood and because they typically change as a result of system deployment”*, Alexander and Maiden (2005) correctly points out that the *“system is made for man, not man for the system”*, and that focusing on the product (or its requirements, as if they would be “objective” entities existing outside organisational reality), rather than on the user, is *“a hangover from the past”*: one would add especially when this occurs with no awareness of the political conditions and circumstances which form the context within which these requirements get elicited.

Requirements engineers for a long time have indulged themselves with the idea of “capturing” requirements, before accepting that requirements are elicited, not captured.

As Easterbrook (2004) explains,

“‘requirements’ suggests that there is someone out there doing the ‘requiring’ – a specific customer who know what she wants. In practice, there is rarely a single customer, but rather a diverse set of people who will be affected in one way or another by the system. These people may have varied and conflicting goals. Their goals may not be explicit or may be hard to articulate. They may not know what they want or what is possible. Under these circumstances, asking them what they ‘require’ is not likely to be fruitful”.

It has to be noted that that users are a type of stakeholders, not the only one. For example, we might have some users (staff) with little power who are going to be using the system, or governments and unions with possibly no direct involvement, and still with a very important and powerful role to play in a certain project.

The idea of having to manage conflicting goals is critical to RE. In the past, as it was confirmed by the report published as part of Software Engineering problems in the first NATO conference, this was rather interpreted (if not simplified) as *“the lack of understanding in System requirements on the part of customers and designers”*.

Managing conflicts and the uncertainty deriving from them seems the same old problem RE practitioners are still unable to face, if – as Sutcliffe (2002) seems to suggest – *“we have not improved our practice much since 1910 and mistakes in RE still cause problems”*.

The extent of such problems is quite clear. Sommerville and Sawyer (1997) cited and agreed with Gao (1979), Gibbs (1994) and Barlas (1996) when they stated that *“it has been recognised for many years that problems with specifications are probably the principal reason for project failure where systems are delivered late, do not meet the real needs of their users, and perform in an unsatisfactory way”*.

Easterbrook (2004) also accepts that requirements (and programs!) *“fail to work in the way we expect, they are unreliable, and sometimes dangerous, and they may create more problems than they solve”*.

It is fair to state that, notwithstanding many technophile temptations, no author has actually and explicitly underestimated the importance of stakeholders and the role they play in RE. It has to be acknowledged that such a shift in the focus from the product to the user, however difficult to accept for many who put “engineering” first, found many supporters among well-known

practitioners in this field, ready to pay extra attention to the human-centred design, and the challenges coming with it.

Bubenko (1995) believed that a crucial methodological challenge to address in RE is

“to improve user-developer communication, much more than to expand the use of formal methods. Current models are not communicative, and descriptions cannot be fully understood by users and stakeholders. Current modelling methods are borrowed from SE, and do not document the reasoning and the rationale behind solutions suggested. Users often sign-off requirements specifications without fully understanding them”.

The very same Sommerville (op. cit.), who has been one of the strongest musketeers of the “engineering” approach, admitted that *“there is no such thing as a typical user”*, and then listed a number of them (e.g., doctors, nurses, patients, administrators, maintenance engineers).

He suggested that “managing the users” is the solution, for example by limiting their viewpoints (*“if too many viewpoints are identified, it is difficult to manage the large amount of information generated and prioritise the requirements”*): although in very specific cases, this could be understandable, in general I would strongly disagree with such an approach. But why to disagree? Would not it be right for us to limit the viewpoints if we want to eventually obtain our final engineering product and the technology, which would eventually support it?

Easterbrook (2004) gives us the answer:

“the purpose of a banking system is not to be found in the technology used to build such systems, but in the business activities of banks and day-to-day needs of their customers. The purpose of an online flight reservation system is not to be found in the details of the web technology used to implement it, but in the desire by passengers for more convenient ways of booking their travel, and the desire of airlines to provide competitive and profitable services”.

In other words, we cannot avoid the challenge of understanding in detail all users (or at least enough of them), their field/domain and their needs if we want to produce a working system that is of any value to as many of them as possible.

On this ground, I would therefore support Easterbrook when he claims that

“software on its own cannot be said to be of ‘high quality’ or of ‘low quality’, because quality is an attribute of the relationship between software and the purpose for which it is used. This means that requirements engineering plays a critical role in understanding and assessment of software quality”.

Other authors seem to over-simplify the circumstances a requirements exercise occurs. Bergman et al., (2002), for example, state that

“two key assumptions frame traditional RE. One is that requirements exist ‘out there’ in the minds of stakeholders (user, customers, clients), and they can be elicited through various mechanism and refined into complete and consistent specification. The second is that the key stakeholders operate in a state of goal congruence, in which there is widespread and coherent agreement on the goals of the organisation”.

Both assumptions above, and especially the second one, would somehow seem to justify those requirements engineers who feel they are forced to guess (or “induce”?) users’ needs and to

produce some imaginary requirements so that the developed system can eventually be perfectly designed against a complete set of requirements, as the original ones provided by users or based on our knowledge of the domain can be incomplete: but even if and when the set of requirements all stakeholders worked on and agreed with seems complete, there is always a chance that some important requirements will be ignored and cause troubles (or even clashes) later on during development or after deployment.

For this reason, getting to know users and their needs is increasingly becoming a very significant and essential step in RE practice, and “to imagine” them is now seen as a poor choice, the same way in which it would be a poor choice “not to imagine” them as parts of the system we are going to engineer.

Requirements engineers need to know their users the same way in which we supposedly know the technology we are going to adopt and exploit to eventually create our system. And it is a good starting point to acknowledge that, in most cases, we do not really know who the actual clients/users are.

Alexander (2003), in his article called “*Stakeholders – Who is Your System For?*”, provides food for thought when he writes that “*there is not one person who you can unequivocally call 'the client'*”.

Engineering projects like the UK’s Network Rail West Coast Main Line Upgrade nowadays run into billions of pounds, take many years to complete, and involve many people, many of whom who could be 'clients', 'users', or 'customers'. In fact, since public money is involved, are not we all customers in some sense?

The question is, therefore, who can legitimately claim to hold a stake in a project's territory; or, to put it the other way round, who should engineers seeking to define the requirements for a project consult about their viewpoints?

There are now two options we can consider: either to decide that, with regard to the example above, it is not possible to find all the users and try to negotiate with them and therefore to accept the risk this “black hole” could itself be the cause of a serious accident as a result of missing requirement(s), or to try to find all users and stakeholders involved.

Leaning towards the first option is Sillitti and Succi (2005), who believes that “*in agile method, the problem of multiple stakeholders is solved by reducing their numbers to one, a single person that represents all the stakeholders involved in the project*”.

This is a wonderfully naïve quote because it denies or simply removes “the” main issue a requirements engineer needs to address, i.e. how to negotiate the conflict of goals:

Does the presence of one individual really address all the political issues in an organisation?

Who is this person?

What is his/her agenda? Who selected them as the user representative and why did they do this? And what are the agendas of other stakeholders involved in the selection?

This quote, if it is typical of agile supporters' opinion, would open up a rich vein of critique.

The primary purpose (and a major problem) of requirements negotiation is to identify and resolve conflicts among stakeholders, not to deny them. It was of great comfort to hear from my supervision team that *“you can negotiate once you have a party, and if you do not have a party then you cannot negotiate!”*

Notwithstanding all this, organisations evolve and whilst different types of hierarchy can coexist within the same organisation, hierarchical models as such are usually still in use (see Schwarz, 2006) and a person or a group of people located “somewhere” at the top of this multi-dimensional pyramid typically has/have more power than those located at the bottom.

Within these hierarchical models, though, does always power inform organisations through executive boards, chiefs, line managers, supervisors, team leaders, etc., with a direction that clearly mimics the force of gravity, from the top to the bottom?

Whilst this kind of structure could be seen as a sort of “survival of the fittest” feature and can be an advantage in an organisation as everyone is clear about whom to get commands from, does it always perform linearly from the top to the bottom?

It could be argued that power is not always representable as simply as this, and in a more confusing situation there are many negotiation processes to be considered, apart from the obvious one between those who pay the money and those who actually do the job, where the art of negotiation is clearly needed and practiced.

Finkelstein (1994) confirms that assessing organisational settings is part of

“the necessary preconditions for effective Requirements Engineering [...]. The process of stakeholder analysis involves identifying those individuals or roles who should have a voice in the requirements engineering process”.

These may be clients, users and other beneficiaries, they may also be people involved in subsequent design, implementation, maintenance of the system. Stakeholder analysis involves understanding their responsibilities, capacities and the organisational relations between them. This analysis serves as a map for subsequent information gathering and a means of interpreting the information provided and its status'. It is a pity they did not produce a notation to set out this knowledge and to capture political power relationships.

Back to negotiation. Even the technophile Sommerville and Sawyer (1997) accepts that

“stakeholders negotiate to agree on a set of requirements for the system. Generally, there are a greater number of desirable requirements than can be implemented so decisions have to be made at this stage about leaving out requirements and modifying requirements to result in a lower-cost system.”

As reported before, Fricker et al., (2015) seem to have spotted the elephant in the room when they stated that *“management support is critical to align the software with the strategic goals of the organisation”*, and therefore appreciated that, in order to be able to find the right person to negotiate with, we have to get to know the client's organisation. In the following table, I report the set of negotiation techniques the last-mentioned authors have identified: it is worthy to underline how they explicitly list “power analysis”.

Table 3: List of negotiation techniques (Fricker et al. 2015)

Technique	Description
Conflict management	Discovering and resolving conflicts among stakeholders and between stakeholders and development team
Handshaking	The review and discussion of implementation proposals to align the planned implementation of the software system with stated and unstated stakeholder needs
Negotiation analysis	Analysing possible negotiation outcomes and selecting a value-creating, fair agreement
Power analysis	Analysing power and influence of stakeholders and planning how to interact with them
Prioritising	Ranking the requirements to obtain an order of how they shall be addressed by the project work
Strategy alignment	Aligning requirements with company strategy, for example through explicit traceability
Variant analysis	Analysing and selecting alternative features or ways of solving a problem
Win-win negotiation	Structured, possibly tool-supported approach to identification of options for agreement and selection of the appropriate option

Structured, possibly tool-supported approach to identification of options for agreement and selection of the appropriate option.

One of the main goals for this research was to identify how to model the politics within an organisation in order to help requirements engineers to better perform and support negotiation processes. Requirements ultimately begin and end with people – stakeholders.

- **3.7.4 Who does the RE job? The politically vested analyst and engineer**

But then, who is this “mythical” practitioner called “requirements engineer”? Until a few years ago, it was supposed (or expected) to be *“an individual [...] capable of utilising knowledge to synthesise effective solutions”* (Davis and Hickey, 2002). And knowledge, according to the same authors, meant (1) knowledge of the problem domain, (2) knowledge of existing solutions, and (3) knowledge of processes, methods and tools of the practice of requirements engineering. *“Only recently –they state- a fourth area of knowledge has been identified: knowledge of how to decide which processes, methods and tools make most sense as a function of certain aspects of the problem domain”*.

In the same paper, Davis and Hickey (2002) state that practitioners tend to work within a standardised code of practice, which barely tackle the deeper and more problematic aspects of the above fourth area of knowledge. More generally, some practitioners tend to be quite “conservative” in pursuing innovative practice and recommendations as they become available, and not coherent in their quest for quality and insight: the title of a paper Davis wrote in 1996 perfectly summarises the criticalities that have been highlighted so far: “Practitioner, heal thyself!”

And whilst someone legitimately wonders whether “we practice what we preach” in RE (Davis and Hickey, 2002), my provocative answer could be: yes, unfortunately we do.

I have grounded such a depressing answer in two major professional bodies in Western Europe which currently confer a recognised status to requirements engineers, and the training handbooks they adopt: the German IREB (International Requirements Engineering Board) and the British Computer Society, here in the UK, providing certified status to business analysts, who need to study modules such as “Requirements Engineering” to achieve their certification awards.

The Germans use “Requirements Engineering Fundamentals” by Pohl and Rupp 2nd Edition 2015 (from now on referred to as REF), whilst here in the UK our practitioners have been trained and assessed on “Business Analysis” by Paul and Yeates Third Edition 2014 (which hereafter will be referred to as BA).

According to REF, *“the requirements engineer as a project role is often at the centre of attention. She is usually the only one who has direct contact with the stakeholders and has both the ability and the responsibility to become as familiar as possible with the domain and to understand it as well as possible. [...] To be able to fulfil all of her tasks, the requirements engineer needs much more than process knowledge”*, and amongst the many skills it lists (e.g., analytic thinking, empathy, communication skills, moderation skills, etc.) there is also a special mention for “conflict resolution skills”. Whilst all other skills (and the process knowledge, obviously) are the (usually secondary or tertiary) object of traditional training and education courses in software and/or requirements engineering, it is unclear whether resolving conflicts is learnt, and if so, how this is achieved.

BA, on the other side states that the analyst, whilst pursuing their “commercial awareness”, must be careful not to emphasise the “selling” of a resolution as their aim should be the actual process of resolving, not the sale of the service/solution.

Both books, in the last instance, fail to spend many words on politics and power, either because they claim faith on “structured processes” as the Germans seem doing (possibly because of their well-known engineering tradition?) or because politics is in last instance perceived as (and/or reduced to) “marketing”, as Britons (the legitimate father of liberal thought, and historically some of the proudest musketeers of market economy?) seems to suggest.

Requirements engineers trained on these books alone will have barely heard of politics and power to be considered alongside their practice and would simply be unaware (and likely unable) to cope with any associated challenge from this education. It is a cultural and mental boundary, I am hereby arguing, that somehow needs be broken: an imprisonment in Plato’s cave from where to escape.

3.8 How a politically aware RE relates to other non-technical factors?

How do I intend to transform what might seem more like a mission than a research programme into a set of achievable research tasks that will meet the goal of helping REs in industry with their practice? I already anticipated that this would not be an easy journey, because, as can be seen from the authors that have been mentioned above, politics has been raised by very distinguished academics and practitioners, who were however left in doubt on how to transform and implement it into RE practice.

Milne and Maiden (2012) admits that they were not the first to talk about power and politics and to attempt a formalisation of their influence: to confirm that statement, they cite sources from the 1980s. In fact, they also refer to a much older source by Dahl (1957), which also I mentioned earlier in this thesis and that includes a proposed mathematical notation to represent power relationships between people.

Still, and quite correctly, Milne and Maiden (2012) expresses his surprise about

“the absence of concrete requirements engineering guidelines and techniques with which to understand the structure of relationships between project stakeholders (Power) and the decision-making process informed by these relationships (Politics)”.

Despite the technical advances in the intervening decades since Dahl published his paper on “Behavioural Science”, it is still becoming further evident that, since other non-technical aspects (finance, human factors, cultural backgrounds, etc.) can be causally correlated to political and/or power factors, any proposed model and its underlying notation would need to be sufficiently sophisticated to *“reflect the social characteristics of [a] complex system”* (Milne and Maiden, 2012). Bergman et al., (2002) should also be taken into account, who according to Milne and Maiden (2012) were among the ones who presented *“a comprehensive argument for reconceptualising RE as a political process”*. As a result of such expected complexity, it will not be a surprise if the model Milne and Maiden (2012) were hinting at would be less engineering-like and more of a political nature.

3.9 A need for tighter definitions? Or a need for better definitions?

Whilst it could be concluded that the discussion so far has given enough evidence for the need to unveil the elephant in the room and to grant suitable consideration to power and politics in RE, one big question still demands attention: how could this consideration of power and politics be transformed into usable tools (conceptual or otherwise), approaches and techniques, so that practitioners could get familiar with them and start using them whilst pursuing overall good practice in their daily job?

Milne and Maiden (2012) contribute to a working definition for both terms that goes in the direction of a possible answer of some usage to engineers and other “techie”:

“power can be viewed as a potential possessed by an actor, A, who can only be properly understood in terms of A’s relationships with others.

So, given a relationship between A and B, if A has power over B then one or more of the following are possible:

- *A can get B to do something that B would not otherwise do, or prevent B from doing something that B would otherwise do;*
- *A can define reality in such a way that B will act in accordance with it.”*

Simply put, one-dimensional power involves A telling B what to do and B complying. Two-dimensional power is where A is able to define an agenda, which B will act in accordance with without needing to be asked. For example, within an organisation this could be seen in a department where A has the power to determine both who attends decision-making meetings, and also to set the agenda for such meetings. Lukes (2005) added a further dimension to this understanding of power. This third dimension can be characterised as the ability to determine values, norms and ideologies.

Experts in software engineering would immediately recognise the above as easily applicable to their field. Over twenty years ago, such a scenario seemed quite evident to Andriole (1998) too, who was one of the first to claim that “*requirements management is a political process, not a technical one*” as he highlighted the risk for naïve programming teams to think that, by writing good-quality code, they were satisfying customer requirements and not instead providing their employer with air cover, promptly available alibis and a more general ability to control expectations, should the inevitable happen and a need arises to focus or diffuse blame.

And whilst there might be genuine and “not-that-political” reasons to pursue a political analysis, for example when, “*with the increasing complexity and uncertainty in requirements engineering (RE), the impact of power relationships between stakeholders become critical to the success of requirements engineering process, especially in requirements negotiation to resolve conflicting requirements*”. (Yang and Liang, 2013)

3.10 Methodology

Having described the landscape for this research, how am I to conduct it?

Cockburn (2008) suggests not to “*believe in any single process or methodology because each works only in a particular and limitation context*”.

While this applies particularly to software engineering, I have adopted and followed his advice throughout the development of all my research and have therefore attempted at identifying and considering several research methods in order to cover as extensively as possible my research topic.

- **Method 1 - Literature Review and historical analysis: a methodological approach**

“*There is little point in reading history unless we are willing to learn from it.*” Wirth (2008)

The first methodology I have adopted consists of a quest for relevant literature and other available sources (trying to prioritise those of greater interest and more accessible to practitioners), following a relatively flexible approach that I would describe as a hybrid of two well established methodologies: systematic literature review, or SLR (Armitage and Keeble-Allen, 2008), and DESMET (Kitchenham et al., 1996).

In accordance with the SLR component, I systematically retrieved and read a number of academic abstracts and papers from a number of research databases (ScienceDirect, Scopus, IEEE Xplore, ACM Digital Library, Springer) or through Google Scholar, but also other non-academic resources (white papers, public reports, publications from relevant professional bodies, blogs, etc.) as they resulted from my searches on the internet. This has been of absolute value to me: any material has then been evaluated individually against a set of pre-identified criteria in order to select eventually a set of suitable documents and sources for the purpose of this study (for further details please refer to Appendix C).

The second, DESMET-like component I have adopted derives from the evaluation methodology developed in the context of a DTI-backed project (Kitchenham et al., op. cit.) aiming at providing software engineers with a structured method for evaluating software tools and methods given certain constraints. Briefly, the method consists in evaluating a given software tool (or a method) against certain criteria whilst playing different roles: the tool vendor, the software engineer, the academic researcher, or a member of an ESSI consortium

undertaking an evaluation of a new method in an industrial context. This method resembles the job of the requirements engineer, trying to ascertain and satisfy all requirements from and to a number of different stakeholders.

In my hybrid literature review, the roles I considered were the author of the source/paper being read, my goals as a PhD student, my supervisors' advice and the RE practitioner at a certain time in history.

In order to use both the techniques, I needed to identify some criteria of relevance against my research goals. These criteria are listed in Table 5 (see Appendix C).

Table 6, always in Appendix C, illustrates, as an example, how these criteria were used for assessing the relevance of two historical selected papers. The same technique/methodology has been used to evaluate further sources and useful information upon which to develop history of RE.

The lightweight systematic review approach I adopted seems to be well known in medicine, biology and social sciences (CRD, 2009), and it is driven by the research goals: the method aims at collecting and comparing many high-quality sources against a number of criteria, in order to eventually focus on those that were better meeting those criteria. In my case, the literature review started by considering the list of criteria that have been adopted at the time and an example of its application against two sources that proved to be crucial as a starting point in this research: the NATO 1968 report and conference proceedings and Alexander's "History of Requirements Engineering" (1997).

The actual searches have been performed by using several keywords in different combinations, not necessarily including the "requirements" term, also because before 1969 its usage had been quite sporadic. The two NATO conferences (1968 in Garmisch the former, 1969 in Rome the latter) have somehow represented the most valuable source in digging for the history of software engineering.

Reports and proceedings from those two conferences had been scrutinised and used consistently in the initial part of this research work. Regular meetings were held with the supervisors to discuss the findings in those proceedings, and to develop a clearer picture of the state of the art in computing and software engineering at that time.

Other papers, both more and less recent than the NATO conferences were also identified. Their abstracts were first briefly reviewed and further sources identified from their cited references. As a result, over 200 papers have been reviewed. Eventually only 130 on those that were offering a relevant contribution to this research goals have been considered.

- **Method 2 - Interviews and personal communications**

The second method adopted for this research was interviews and personal communications with those who have been played a crucial role in the history of software and requirements engineering. At the time, relevant ethical approval had been applied for and obtained. By means of these interviews and personal communications, I gained further understanding of not only the past, but also of those present events, processes and practices that are currently affecting the professional identity of requirements engineers.

This dual method had been applied for several months, and yet answers to research questions seemed to be missing a crucial component. Only after what became the “benchmark” event, a long interview with an internationally well-known practitioner in RE (who wishes to remain anonymous), a conceptual key, that of politics and power, was eventually explicitly identified as the crucial factor to be considered, and yet a layer of pre-existing “distracting” information, mostly from the literature survey, initially prevented a full appreciation of how insightful this key could turn out to be.

As often happens, only when a certain number of conditions were met, suddenly a new light or perspective appeared, under which to look back at all the data that had been accumulated, and things immediately got somehow clearer: rather than a case of serendipity, a little and yet crucial “paradigm shift”, to use Kuhn’s (1970) words, occurred in my research.

At the beginning, both a literature survey and interviews were used in order to develop a complex and yet consistent picture of RE. But both methods proved to be sometimes ineffective and time-consuming, as they included:

1. repeated reviews of each interview, which had been recorded and transcribed,
2. discussions with the supervision team and other practitioners,
3. the usage of sometimes even “extreme” metaphors (one was based on the duck pond game!) to better clarify issues, and
4. the comparative analysis of the two manuals from the German and British professional bodies in the RE field mentioned above, that are currently used for certification purposes worldwide

After the above paradigm shift happened, it was possible to classify the previously identified literature with a new status of significance. Since the shift occurred, new approaches to investigate and an opportunity to model how the conceptual key of politics and power could be controlled and used by practitioners have also been identified and considered.

- **Method 3 – Development of a simple, intuitive, additional notation to represent and model political relationships (more on following Chapter 4)**

In order to ensure any solution could be of immediate value to practitioners (who usually come with a background in IT and software development), an unavoidable requirement would have been to bank on assumed previous skills. Software and requirements engineers are used to model problems and systems by means of modelling techniques, mostly based on diagrammatic, possibly intuitive notations.

Understanding, capturing and modelling politics should be an activity requirements engineers should be enabled to do by adding very little burden to their already busy daily job, and ideally “on the back of an envelope”, as a simple add-on they might want to pursue informally.

Keeping all the above in mind, with my supervisors we envisioned a simple way to represent politics and political relationships, by using emoji pictograms: most of them are part of a universal language, which practitioners could easily adopt and exploit to assess and produce models that include an extra layer of “political” information, without the need to actually introduce any new notation.

An introduction to the emoji-based notation (which I have called it Political Emoji Notation, or PEN) and a sample of emoji-aware UML and organisational charts are proposed in the following chapter. The main idea behind this notation is not to provide a precise, unequivocal and syntactically coherent tool, but rather a simple platform to take notes, potentially support (informal and confidential) communication and share reflections on how to deal with politics.

- **Method 4 – Impact Evaluation on identified case studies**

Another methodology that has been identified and used for the proposed Political Emoji Notation for accessing the result of the two case studies; this is called Impact Evaluation (IE), and it is mostly based on Goodman's (1947) counterfactual analysis.

Impact evaluation has been mostly adopted in social and political sciences and especially in those operative environments (such as third sector agencies and NGOs) where policymaking is a core activity. This methodology requires decision-makers (here, it would be experienced practitioners in RE) to review any of their past decisions and/or projects and consider what would have happened if (or if not) some extra factors (such as the one have been identified as playing a crucial role in RE, and any related tool or methodology) would have been taken into account (or applied with a different emphasis), were they available.

For a review on this methodology, including its criticalities, I will refer later more in detail to a number of seminal works: the document produced by the Development Co-operation Directorate (or DAC) of the OECD entitled "Outline of principles of impact evaluation" (OECD, 2006), and a paper written by Aguirregabiria (2012), on using counterfactual experiments in models with multiple equilibria, which it is arguable would typically represent RE professional scenarios.

Another source is the website of the International Initiative for Impact Evaluation, "*an international grant-making NGO promoting evidence-informed development policies and programmes*" (3ie, 2016).

In this research, I would run Impact Evaluation against two relevant software systems development project failures (as case studies suitably documented in literature), by applying the proposed emoji-based modelling exercise to these case studies and analyse the potential impact PEN might have had if practitioners could have had it available at that time. The two case studies are described in detail in sections 5.2 and 5.3 below.

It is useful to highlight that impact evaluation could be used also to address retrospectively projects directly from the actual practitioners involved, and this could be the object of some future work.

Chapter 4: Modelling

Modelling (and modelling techniques) can be used to identify, describe and highlight the important features in software development and to separate them from the unnecessary aspects and details of a given system, especially in large and complex projects. Alternatively, when models are used for focusing on and analysing specific issues, they filter out what is reasonable to consider irrelevant or less impactful. Any development exercise would typically employ several types of models, each model type emphasising a different view.

Sommerville (2016) believes that

“System modelling is the process of developing abstract models of a system, with each model representing a different view or perspective of that system. [...] It is important to understand that a system model is not a complete representation of a system.”

There are many different notations or approaches available for modelling stakeholders’ and system requirements: their value is mainly in supporting requirements engineers to understand a certain problem and to communicate about it by minimising those ambiguities and partialities, which natural language would not help preventing otherwise.

In the same book, the author explains that

“Model are used during the requirements engineering process to help derive the detailed requirements for a system, during the design process to describe the system to engineers implementing the system, and after implementation to document the system’s structure and operation.”

Modelling techniques based on pictures and diagrams, such as those grouped under the UML umbrella (for example, use case diagrams), are commonly adopted by practitioners, and whilst non-diagrammatic requirements modelling techniques are also used, they tend to have one crucial advantage over words and human language, as philosophers of language and cognitive psychologists have well explained: they externalise the meaning they convey, differently from the written (or spoken) word, which – as a symbol – refers to a meaning we hold in our mind.

Chen et al., (2012) claim that

“visual requirements models are one of the most effective ways to identify software requirements. They help the analyst to ensure that all stakeholders including subject matter experts, business stakeholders, executives, and technical teams understand the proposed solution. [...] Visualization keeps stakeholders interested and engaged, which is key to finding gaps in the requirements. Most importantly, visualization creates a picture of the solution that helps stakeholders understand what the solution will and will not deliver”.

It is a common experience that designing a future system usually requires modelling it by using different notations.

Chen et al., (2012) also state that

“to make a requirements process ‘fly’, the first step is to understand that there is more than one kind of requirements model. A shopping list of requirements is invaluable in a contract, but on its own, it’s desperately difficult to check for correctness and completeness, and it does not offer any suggestions on how to discover requirements,

either. Different requirements models are needed to assist with discovering, checking, and analysing the requirements. The ‘shopping list’ is an output, not the one-and-only input.”

4.1 Modelling politics in software requirements engineering

There are many different notations or approaches available to engineers and developers for modelling both stakeholders’ and system requirements and, in last instance, to gain a deeper understanding of the underlying problem(s). Models have always been a valuable instrument in the toolbox of requirements engineers, for their ability to detect and depict, at different levels of abstraction, not only the diverse components of a certain system and how they interact within that system, but also the perspectives through which stakeholders and users can look at that system, when taking into account their tasks, goals and expectations.

The same principle, I would argue, could be applied to capture requirements and support the understanding of the political dimension (or dimensions) surrounding any large-scale software project: modelling can help capturing valuable and useful information and agendas of political nature by highlighting emotional status, power relationships and potential conflicts of interest among stakeholders during the whole software requirements life cycle, from elicitation to specification and beyond.

For the above purpose, I have developed a specific notational technique, which I have named Political Emoji Notation, or PEN. PEN aims at showing “political” relationships among stakeholders, and how these affect each stakeholder’s emotional response and attitude towards the project.

The main drive behind PEN is simplicity, so to encourage practitioners to use it, and yet it comes with an important disclaimer: the intention of PEN-based modelling would always be to model power relationships within or across organisations during any system development project, but by no means of getting practitioners involved in politics.

Simplicity in PEN would be supported by its basic notational elements (emojis, arrow lines and very little else) and the fact it would be totally unconcerned about a precise, objective or comprehensive representation of the political dimension affecting a given project, its stakeholders and the political issues they carry.

The number of lines would show the level of influence and the direction of power is shown too. Unconnected or absent stakeholders means they would be not involved at that specific project stage or just irrelevant.

PEN would still encourage the requirements engineer to discover all the relevant (and sometimes less obvious) groups of stakeholders (e.g., trade unions) and to find out the source(s) of power and the influence direction at the same time, which may be crucial when the future system will have to be designed.



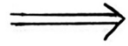






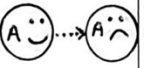
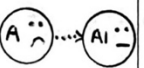
Shortly, PEN can present a clearer picture and recognise a situation in a straightforward way, and hopefully provide more understanding to support the future system.

It is easily arguable that some requirements practitioners might indeed notice and become aware of politics in organisations and projects, but this might not be enough, and we would still need

a simple notation to clarify the thinking about it, to capture the complex (and subtle) influences among stakeholders and document in an intuitive, graphical way both informal and formal power relationships.

The elements forming the Political Emoji Notations (PEN) are shown and described in the following table:

Table 4: A notation for graphical modelling of politics in requirements engineering

Name	Notation	Comment
Entity		Identification will be presented by circled entity (with name, title or other identification)
Formal Power Relationship		Single lines will be used as connectors and represents power within the organisation. More lines show more power.
Power Direction		Lines with arrow shows the direction of influence/power which can only be either one sided.
Power to Block		Lines with a cross in the middle represent power to block which means that person is not reachable, but he/she might be reachable through another person. For instance, a manager will not allow a requirements engineer to talk to his employees as he might find out about the influence/power within the organisation.
Note: Status of stakeholders will be shown by emoji faces which can be defined as some internal or external entity that interacts with the system. The purpose of using emoji faces including facial expression.		
Happy Stakeholder		'Happy' stakeholder (satisfied with current system requirements)
Sad Stakeholder		'Sad' stakeholder (dissatisfied with current system requirements)
Neutral Stakeholder		Neutral stakeholder
Informal Relationship/influence		Informal relationship/influence
Stakeholder with unknown emotional status		Not sure about his/her emotional status
Change of Emotion, attitude in the role or change of mind		Change of face from emotion with the same name, title or identification connected by a Dotted line)
Change of personnel, department, organisation, etc.		Change of face from emotion also with the name, title or identification followed by a number, e.g. A to A1 or A1 to A2 connected by a Dotted line)

The suggested graphical notations, PEN, would not aim at solving all issues politics may raise in real-world software development projects: but even if it just raises awareness, this will already provide practitioners with some advantages.

I would like to underline the above notation seems particularly good for inexperienced practitioners when they need to identify who is the key person to talk to with regards to a specific problem or when a new requirements engineer replaces the old one and needs to understand what is going on in the project in no time.

Given the sensitivity of political issues, it is strongly recommended that diagrams prepared using the PEN notations is only to be used confidentially by the requirements engineer and his/her team, thus remaining strictly private.

Below, benefits and advantages of PEN are listed in more detail, without any particular order.

Benefits and Advantages of PEN

- PEN is a graphical/visual technique and not a text-based notation, and visualisation makes stakeholders more interested
- Pen is for the measurement of the state of the political dimension.
- PEN shows relationships among stakeholders and their nature between actors and/or stakeholders (e.g., formal VS informal relationship)
- Visualisation helps to keep inexperienced practitioners interested and engaged
- PEN raises questions
- PEN highlights and models politics without getting the practitioner involved with politics
- PEN drawn diagrams are strictly private/confidential to the requirements engineer and his/her team, due to the sensibility of political issues
- PEN helps to find the keyperson, the underlying decision maker and/or the influencer to talk to as it might not be so obvious
- Even if PEN does not solve the problem, it would make us closer to solving it
- PEN uses emoji, a universal and easy way to represent emotions by means of sketched facial expressions
- PEN allows the new practitioner to join the project mid-way to understand what is going on in the project
- PEN shows emotions, e.g.: Happy, Sad, Neutral
- PEN is suitable especially for the early phases of system modelling, as it allows to investigate a “sad face” and do something about it before it is too late
- In PEN, same actors can be represented differently in different scenarios, e.g. A and B are both sad for requirement area number 1, but A is sad, and B is happy for the requirements area number 2
- PEN support showing dependencies
- PEN shows level of dependencies and power
- PEN is bottom up
- PEN is dynamic and can manage changes
- PEN is easy to understand, model and change
- PEN is agile, because it represents easily what would otherwise require too many words
- PEN is capable of showing all stakeholders even outside the organisation, like trade unions, and also is capable of showing the level of power they come with, for example, trade union as users’ representative.
- PEN is designed to be easily usable, in very informal manners
- PEN can help control conflicting agendas as a tool for negotiation purposes
- PEN can support socio-technical modelling
- PEN notation allows to be easily capturing and represent political aspects of a software requirements process

- PEN only includes people/roles, not artefacts
- PEN is designed to be intended for practitioner use only
- Pen does not need a software tool for people to use it (actually, a “formal” use of it could even be discouraged!), as hand-drawn diagrams would demonstrate later on!

Comparing PEN with an existing notation

PEN is not the first notation intended to help software requirements engineers to capture a “political” situation by modelling actors’ behaviour and intentions within a certain project.

The i* (pronounced “eye star”) modelling technique was developed to model stakeholder’s relationships on the basis of their “intentional properties (goals, beliefs, abilities, commitments)” and how they use these properties to “assess their strategic positioning in a social context” (Yu, 2011).

Dalpiaz et al., (2016) noted that

“the i language was presented in the mid-nineties [...], was quickly adopted by the research community in fields such as requirements engineering and business modelling [...and] has been applied in many areas, e.g., healthcare, security analysis, eCommerce.”*

i* aims at a much more formal modelling exercise than PEN, on the mathematical foundational grounding of the so-called principal-agent problem, as it emerged in the 1970s, which aims at modelling how different agents make decisions and/or take actions in their own best (and potentially conflicting) interest. It goes beyond the scope of this research to get into further details of i*, even when several authors have explored or discussed the potential of this modelling language in the context of goal-oriented requirements engineering (Yu and Mylopoulos 1998).

However, there have been criticisms of its suitability for use by practitioners. For example, Dalpiaz et al., (2016) also mention that *“the most critical [problem] is the difficulty to spread the framework outside the experts’ community”*. The same authors argue that i* is difficult to learn because of the “intricacies” of the language, to teach because there is not a shared body of knowledge and to use because technology does not support it.

There is also a more fundamental problem with i*: *“actors links are binary, linking a single actor to a single other actor”*, in the assumption political relationships can be deterministic and mutually independent. This assumption means that any derived model in i* could end up in the paradoxical situation of been over-complicated from a formal viewpoint (making it impossible to be of any real use to practitioners in their daily job) and yet potentially inadequate to represent real political relationships captured at a certain time among a certain group of stakeholders, given these might not follow predictable dynamics.

PEN, on the contrary, places its main value in ensuring practitioners can use it to the best of their abilities, “on the fly”, in an attempt to capture some particular relationships within a specific scenario, without being too concerned about the certainty its representations are objective, complete and coherent alongside the whole project life cycle.

Comparison of i* (and UML) with PEN

A comparison between Political Emoji Notation (PEN) and i* shows the following similarities:

- They are both based on belief that capturing and understanding political aspects of a software requirements process can be useful in a real-world software development environment
 - They both assume it is worth expending the effort needed to achieve an understanding of the relevant elements in a political environment, including relevant actors (but ours only includes people/roles, not artefacts), relationships between these actors, nature of these relationships
 - enough of the above aspects can be expressed using a graphical notation, and both i* and PEN show dependencies and relationships among stakeholders
 - they were both intended for practitioner use (but i* has somehow proven to be not useful for practitioners in their daily job)
 - they both lack a coherent body of knowledge (but admittedly i* is better documented than PEN (so far!))
 - does not need a software tool for people to use
- However, there are differences between PEN and both i* and UML worth noting:
- i* is full formal language and its complexity might deter practitioners from using it in their ordinary job, during the elicitation process or at any other stage of the requirements life cycle
 - i* is objectively more complex than PEN, as it aims at a syntactically formal coherence, well beyond PEN's scope
 - Pen covers politics without aiming at a full integration with other modelling languages used during the requirements engineering process, whilst i* aims at integrating with UML
 - i* assumes all requirements and stakeholders within a certain project exist and act coherently, as if regulated by a set of universal, immutable laws. PEN does not assume any coherent set of rules or behaviours: they can change within the same project, the same actor, the same requirements, and even more when some different scenarios are considered: e.g., A and B are both sad for requirement area number 1, but A is sad and B is happy for the requirements area number 2
 - differently from i*, PEN consider and represent emotions (e.g.: Happy, Sad) and levels of dependencies
 - PEN is bottom up whilst i* is top down
 - PEN drawn diagrams are strictly private/confidential to the requirements engineer and his/her team, due to the sensibility of political issues
 - PEN seems to be more dynamic and can better capture unpredictable changes
 - PEN is easier to model and understand
 - PEN seems to be more Agile (it “embraces changes”) and less concerned with the assumption actors would supposedly always act consistently alongside a certain project
 - PEN enables engineers to consider and model actors even outside the actual organisation and/or project boundaries they are focusing on (e.g. trade unions)
 - PEN can control conflicts by allowing practitioners to suggest “irrational” or inconsistent negotiations, which would not be possible in i*
 - PEN is capable of showing the level of power for each stakeholder, on the basis of its ability to represent cumulatively common interests, for example trade unions or lobbies

can be seen as powerful representatives of otherwise less powerful individual stakeholders.

- i* covers actor oriented and goal modelling for business modelling, in the assumption these stay coherent during the whole life a certain project and change more or less deterministically because consistently obeying certain rules or “laws”, whilst PEN do not make assumptions on actors or their goals
- i* answers who and why but not answer what, whilst PEN could also answer what

Ratio analysis

PEN has a number of advantages as this has been revealed deeply in section 4.1, using the small digital icons, Emoji, which is a graphical representation adds emotional context to written content or drawn diagram, in our case to the PEN notation.

However, PEN could be more beneficial if the notation output represents numerically.

Representing notations numerically enable the relevant requirements engineering team members to objectively compare the findings and conclude for the future of the project. Also, saves time, helps in making senses, reducing risk and making analysing the larger models easier.

In addition, planning and managing power and politics will be easier and as a result making the decision for the future of the project is also easier.

Ratios is one of the data measurements scales that are used to compare values, could be used to compare the differences, in this case, to quantifying power and politics, happy, sad and the neutral faces to compare with the total number of faces in each model.

On the other hand, simply the number of changes during the life of projects which could be positive or negative changes.

For highlighting the objectivity of the PEN notation, the Ratio has been chosen to show the number of Happy, Sad, Neutral.

Advantages of Emoji Notation in software requirements engineering, compared to other modelling notations

There are several benefits associated with using emojis as a notation in software requirements engineering, compared to other modelling notations. Simplicity is one of the key advantages as it reduces the time required for users to learn and become proficient with the notation. When a notation is designed with ease of use in mind, it becomes more intuitive and requires less effort, thus reducing the learning curve.

This, in turn, leads to faster adoption rates, increased user satisfaction, and a quicker understanding of the system's requirements. Additionally, by reducing the learning time, valuable resources such as time and money can be saved, which would have otherwise been spent on

training and support for the software requirements engineering team. Simplicity is often regarded as a fundamental principle of good design, resulting in more efficient notation usage, fewer errors, and improved user experience.

Emojis, being widely used and understood by people from different countries, offer a visual cue that simplifies communication and allows for the expression of emotions without the need for detailed explanations. This feature becomes especially valuable when working on projects involving team members located in different countries. The use of emojis as a visual notation also helps to avoid misunderstandings and misinterpretations.

Unified Modelling Language (UML) is widely used in software engineering. UML employs graphical notations to model various aspects of a system and often illustrates relationships between the attributes. However, it should be noted that our intention is not to replace existing notations but rather to complement them with an additional notation that specifically documents the impact of power and politics on the project. PEN serves as add-on in this regard.

Furthermore, it is important to mention that, at the time of thesis submission, no other notations related to power and politics were discovered, except for i*. As discussed in section 4.1, of the thesis. i* does not capture the same essence and is not suitable for highlighting and modeling politics.

4.2 Discussion

An important aspect of requirements engineering is that people may change their view of the benefits or disadvantages of one or more system requirements as a result of instructions from more senior members of the organisation, or for reasons or influences outside the scope of the formal organisational structure. A couple of examples of how the notation would reflect changing attitudes to a proposed system have been shown below: A's emotions change from "happy" to "sad", or from "neutral" to "happy".

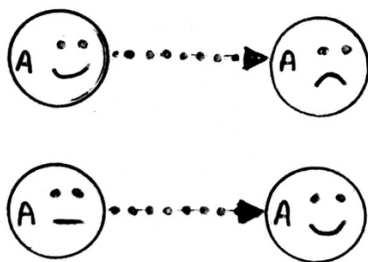


Figure 9: Outcome of Informal relationship/influence: change of views/status

Figure below shows a simple organisational structure (or an organigram), such as might be found when documenting a hierarchical organisation. This is what is immediately visible from aspects such as organisation charts, job titles, formal roles, size of offices and quality of office carpets and so on.

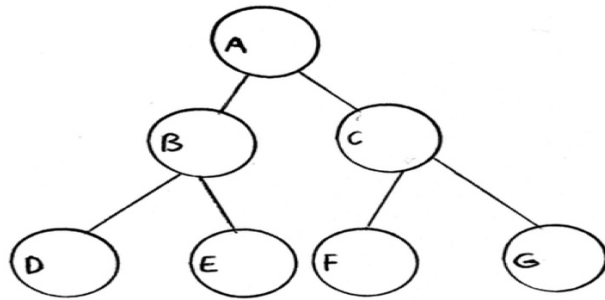


Figure 10: Traditional organigram

It is generally the case that those nearer to the top of this pyramid have more power than those nearer to the bottom. To show the assumption of the direction of power, arrowheads were added to the lines as shown in the following Figure.

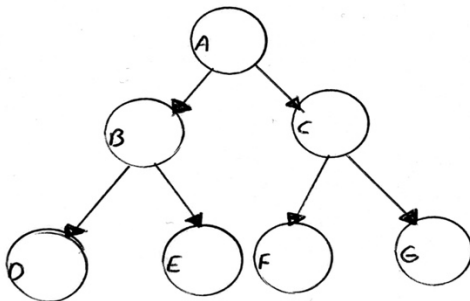


Figure 11: Direction of power (the usual assumption behind the organigram)

The above diagram also represents who reports to whom in a company, and who can formally direct subordinates. However, it does not capture the degree to which a senior member of the structure can direct their subordinates – their relative power. To represent this, more lines have been added to the more powerful relationships, as shown in the following Figure, in which A has more power over B than they have over C, and C has more power over F than they have over G, and even more than A has over B. This may be due to the job roles, some of which are more directed than others; compare, for example, the relative power positions of a finance manager over a finance clerk whose work is routine and follows a set of rules laid down by their seniors, with the power of a design director over product designers whose creativity and independence may be valued rather than discouraged.

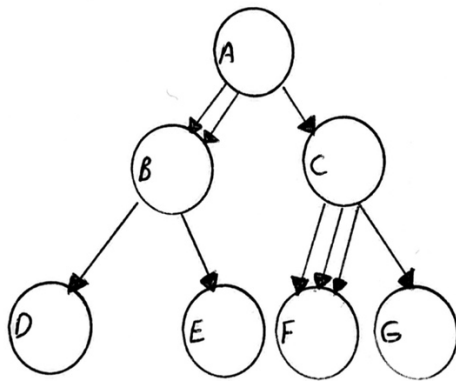


Figure 12: Organigram showing relative power (with multiple arrows)

After “official” and real power relationships, the next most important political factor requirements engineers should consider are the “views” (or opinions, together with their consequent emotional status) both users and decision-makers might have on the system requirements as currently specified. The PEN notation can then show how happy or unhappy each participant is. This can be done in our diagram by applying emojis to represent the status of stakeholders.

The following Figure adds information on the opinions of each person in the hierarchy as to the system requirements. This example show that all are currently happy with these, with the exception of one low-level person (D). Ostensibly this might indicate that there will be no major problems in obtaining support from people with sufficient authority and gain eventually their agreement for the requirements.

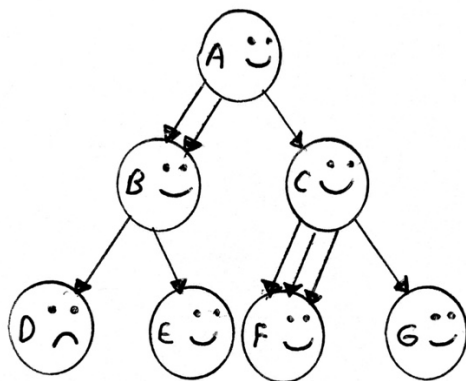


Figure 13: Organigram showing relative power and status/views of each person.

However, not all the relevant relationships and influences are reflected in the formal structures as documented so far. For the requirements engineer, in order to fully understand the complete political environment within which they are working, it might prove to be essential to show in a diagram also the informal, organisational relationships.

The following Figure shows an example of an informal relationship between two entities which can change the whole situation. Here for some reason the apparently lower-level member of the hierarchy has an influence in decision making beyond that which would be expected from

their apparent lack of formal power. This might be due for example, to a back channel within the organisation, or a personal relationship outside work between the two people, such as a common sport or private relationship. Documenting this is important for the requirements engineer to gain a clear picture of which influences (and how) can affect decision making processes; also, it would show the importance of maintaining the privacy of this model from people who might not be aware of the relationship being somehow captured.

In Figure 14, D has a relationship, whose specific nature is not specified here, with A. Whereas Figure 13 suggests that D’s unhappiness with two requirements can be ignored because of their comparatively low position in the formal hierarchy, an informal influence on A might cause A to change their mind to a greater or lesser extent, and thus cause A to act or decide in a manner which would be unexpected in the absence of this information.

Other internal or external people, or roles or entities that interact with the system and will have influence over decisions on its requirements, can also be represented in this notation using the dotted “informal influence” lines shown below.

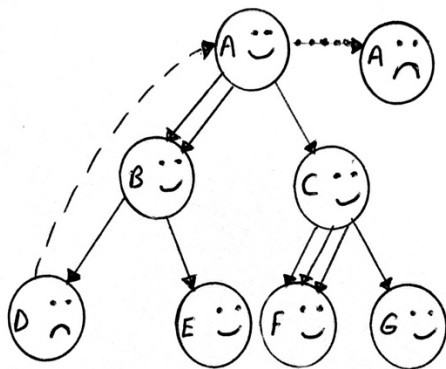


Figure 14: Informal influence of D on A

In the above figure, it becomes clearer how the informal relationship and power/influence of D over A has turned A from happy to unhappy mode. Give that A has more control to B than they have over C, the mode of B may also be changed from happy to unhappy whilst C remains happy. This situation is reflected in Fig 15 below.

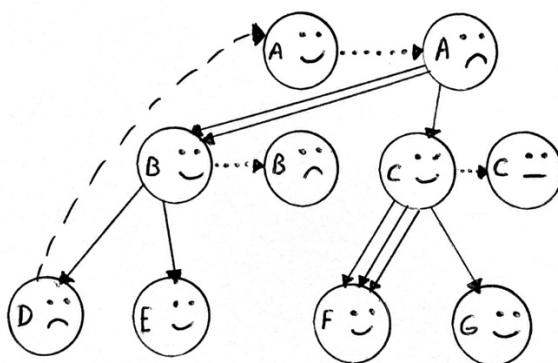


Figure 15: Changes of views/status after informal influence of D on A

To summarise, we ended up with a model which enables the requirements engineer to capture how the main decision-maker (A) was initially happy with the work of the requirements engineer, but then became unhappy (A) under the informal influence of D: a relatively complex scenario, potentially triggering a need to change the requirements. Should this possibility have been known earlier, the requirements engineer might have talked and listened more to D who has proven to be an informal and yet significant influencer.

4.3 Including the emotional side of the political dimension in modelling

As noted above, current software requirements modelling techniques based on pictures and diagrams still come with an important flaw: they have been designed without considering the importance of capturing and analysing the political dimension of the context (or the “broader” system) they are to be engineered for.

Whilst analysts and designers focus on the structural components of the system to be developed, they are led to proceed as if the system they are designing could be totally abstracted from reality and the political component it embeds, including power relationships, sympathy, common interests, "clan belongingness" and other similar (and emotionally characterised) affecting factors, all of which has been proposed to simply denote as "political relationships".

Two examples of how existing notations could be made PEN-aware are given here, as a platform to support communication and share reflections on how to deal with politics within traditional technology-oriented modelling exercises.

Example 1. A certain process (say representable as a level 2 DFD) is discussed between a manager (who is the "official" decision maker) and two workers: the requirements engineer has spotted that worker 1 is positively influencing the manager, and the other worker seems unhappy.

However, the requirements engineer has been told that the second worker is going to be unhappy with the project and likely to actively obstruct the work: the manager has de facto already decided not to be influenced (or bothered) by worker 2.

As a little insert in the DFD (say at the top left corner of the page), the situation could be represented as a quick sketch, as in Figure 16.

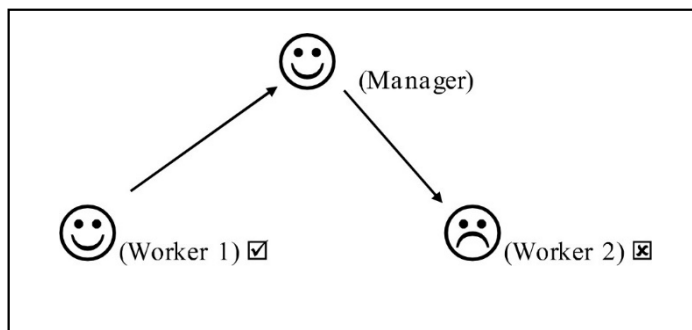


Figure 16: Political relationship, Example 1

This information is politically useful (and usable) for many reasons:

- It is particularly good for inexperienced practitioners as they need to identify the key person in the organisation for solving possible problems as it might not be obvious.
- If the current requirements engineer leaves the project, any incoming replacement practitioner could immediately learn which client decisions are authoritative
- Whilst obviously the manager is the "official" decision maker, the political relationship says something about who the "underlying" decision maker is, and who actually influences whom
- If the manager is unavailable for acting as -say- the product owner, practitioner knows whom s/he could talk to. There is no need to represent an arrow pointing from the manager to worker 1: it is safe to assume that the manager does somehow exercise some power on worker 1. But it is more informative the arrow from worker 1 to manager. Such a simple extra notation is not immune from some sort of ambiguity: arrow from manager to worker 2 could be seen either as the ordinary influential relationship between the manager and the worker s/he coordinates, or it could be seen as an "active" power exercise "against" worker 2.

The argument here is that both circumstances can be similarly dealt with by the same representation: whatever requirements worker 2 has an interest in, it would be worthwhile to refer to the manager.

This example also shows an advantage of the informal PEN notation; that freehand sketches can be added to existing paper documents without the need for technological support.

Example 2. In other scenarios, there might be the need to consider the political dimension associated to the design of more atomic elements of a system: in such a case, political relationships can be highlighted (even partially, by simply representing one of the parties compounding the relationship) within the actual diagrammatic representation of that element, as per the following Figure, which “says” some methods of a certain UML class are discussed.

A method x() is about reporting some complex information: a behaviour expensive to implement. Let us assume the political relationship from the previous example stands, as this class affects the same manager and the same two workers.

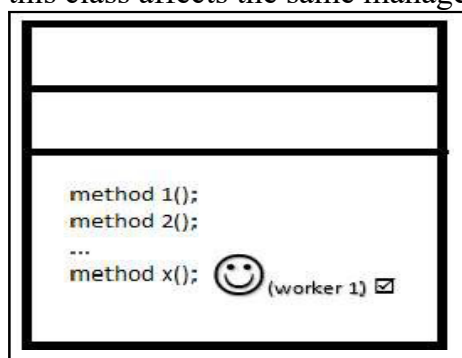


Figure 17: Political characterisation within a UML Class, Example 2

In this case, the political relationship is useful not just for allowing designers to immediately recognise whom to ask for more details about that method, but also which "political" priority to give and which readjustment has to be sought should any conflict arise between say implementation decisions, etc.

The intention, it is important to note here, would be to model politics without getting involved with politics: it is not up to requirements engineers to involve themselves with less influential users or stakeholders, like trade union could do, say for the sake of equality or fairness. This model is not for RE to have an involvement or a saying in political dynamics as they can occur within organisations or projects.

Two general points need to be taken into account: confidentiality of the political relationship and representation of more complex relationships than those provided here. Confidentiality: no stakeholder should be made aware of the notation, as it could easily generate a butterfly effect in the organisation or the project. It has been proposed that political notation is only for the benefit of the requirements engineer.

Complexity: when a political relationship becomes more complex than the above (intentionally simple) examples, inevitably a more structured representation is needed. A typical case is when there is a need to represent stakeholders who are exercising their political influence behind (and sometimes even contradicting) the official organisational chart. In this case, little amendments (like reverting the directions of a relationship) to the model can achieve the expected outcome, as for the example in Figure 16.

It is important to underline that the political dimension does not need to be represented with the same extent of accuracy, unambiguity and systematic usage we usually demand from traditional engineering modelling techniques. The very same occasional presence of political notation could be seen as an important hint to the practitioner, who is then alerted and can pay special attention to individual requirements and/or stakeholder.

Also, practitioners need be aware that any description of the political dimension depends on how the personal interaction they might engage with stakeholders on a certain requirement, and therefore inherently carrying a subjective evaluation, which not necessarily is to be confirmed by another practitioner.

As previously discussed in section 4, when considering PEN diagrams in terms of confidentiality, it is important for the requirements engineering team only to have access to this potentially sensitive information on the political context in which they are working while themselves avoiding any political involvement. This involvement could impact the relationship between the organisation and the developer team. In extreme situations, public knowledge of stakeholder conflicts may even influence financially important attributes of the client organization such as its share price, and potentially lead to legal actions against the developers.

4.4 Impact evaluation based on case-studies

As mentioned in the previous chapter, within the methodology section, impact evaluation is one method I shall be using to evaluate the value of PEN-derived models as they will be applied to case studies in Chapter 5.

The purpose of impact evaluation is to check (even if only speculatively) the negative or positive impact of an applied process to its outcomes. The history of impact evaluation in some countries goes back for many decades (Oakley, 2000), since when it has been adopted to evaluate the impact of social or economic policies on certain groups of people, regions or organisations of any sort, and the methodology has not been immune from several criticisms. Confirmation bias could be count among the worst risks of impact evaluation, meaning that this evaluation method could lead the evaluator to mistakenly confirm what they believe or

claim it could be the real impact about a certain process given some circumstances that cannot be controlled in a proper experimental setting.

Walsh (2018) cited Gugerty and Karlan (2018) who believes that

“bad impact evaluations can also provide misleading or just plain wrong results, leading to poor future decisions. Effective programs may be overlooked, and ineffective programs wrongly funded. In addition to such social costs, poor impact evaluations have important opportunity costs as well. Resources spent on a bad impact evaluation could have been devoted instead to implementation or to needed subsidies or programs.”

Although the disadvantages of impact evaluation have been noted, several reasons led me to its adoption for this project.

Leeuw and Vaessen (2009) believe that

“in field like crime and justice, education and social welfare, impact evaluations (IEs) have over the last decade become more and more important”.

The same authors highlight the important reasons for doing impact evaluations:

IEs provide evidence on “what works and what does not” (under what circumstances) and how large is the impact. [...] Measuring impacts and relating the changes in dependent variables to developmental policies and programs is not something that can be done “from an armchair”. [...] Academically well regarded. [...] Impact evaluation can gather evidence on the sustainability of effects of interventions; [...] IEs produce information that is relevant from an accountability perspective; [...] Individual and organizational learning can be stimulated by doing impact evaluations.

Gertler et al., (2016) provided a clear guidance on how to use impact evaluations. They claim that

“impact evaluations can be divided into two categories: prospective and retrospective. Prospective evaluations are developed at the same time as the program is being designed and are built into program implementation. Baseline data are collected before the program is implemented for both the group receiving the intervention (known as the treatment group) and the group used for comparison that is not receiving the intervention (known as the comparison group). Retrospective evaluations assess program impact after the program has been implemented, looking for treatment and comparison groups ex post.”

Even if the same authors, in their book, argue that results from the prospective evaluation are usually more solid and valuable than those from the retrospective evaluation, retrospective impact evaluation has been chosen for the purposes of this project. A few reasons led to this decision:

- It has not been possible to get involved in big projects and/or large organisations (e.g., BT and their projects), for the purpose of setting up a quasi-experimental approach grounded on a prospective evaluation

- As the application of the PEN notation to individual projects is to be considered confidential to RE and his/her team, neither it would have been possible to use prospective impact evaluation without the risk to divulge confidential (and potentially disruptive) information within affected organisations, nor it would have been possible to use those data to be published afterwards for privacy or even reputational reasons
- When requested, permission of publishing reports from formal interviews and informal conversations with experts, practitioners and even academics was granted only on the assumption they would be reported anonymously. This would have severely affected a research study based on a prospective impact evaluation without the possibility to cite the actual project and/or organisation it refers to.
- Retrospective evaluation was feasible because applied to well-known case-studies publicly available in literature.
- Scope of this research is to provide some insight within which to generate and start a validation process for a new conceptual (and yet reasonably useful and usable) modelling tool, aiming at supporting practitioners in their daily jobs, which conclusion would need the involvement of more than one researcher.

In the next chapter, therefore, I will apply retrospectively impact evaluation to a couple of well-known (and well-documented) case-studies: the MANDATA project (Case study 1) and the National Program for IT (NPfIT) in the National Health Service (Case study 2).

It is interesting to note that both case-studies refer to projects from public sectors (one in Australia, the second in the UK), which is why -it is arguable- documentation is publicly available: in the private sector, normally, failed projects tend not to be that documented and publicly available.

Chapter 5: Applying the Notation

5.1 Introduction

As it has been discussed in sections 3.10 and 4.4, impact evaluation (in its “retrospective” application) is the methodology used in this research to evaluate the usefulness of the proposed notation, Political Emoji Notation (PEN). The purpose of this methodology is to check positive or negative impact of the applied process to the outcome (if any).

For this purpose, two case studies have been chosen:

The first case-study is based on the Mandata project, as it has been analysed by Sauer (1993). Sauer claims that Mandata demonstrates the importance of support for the success of any project.

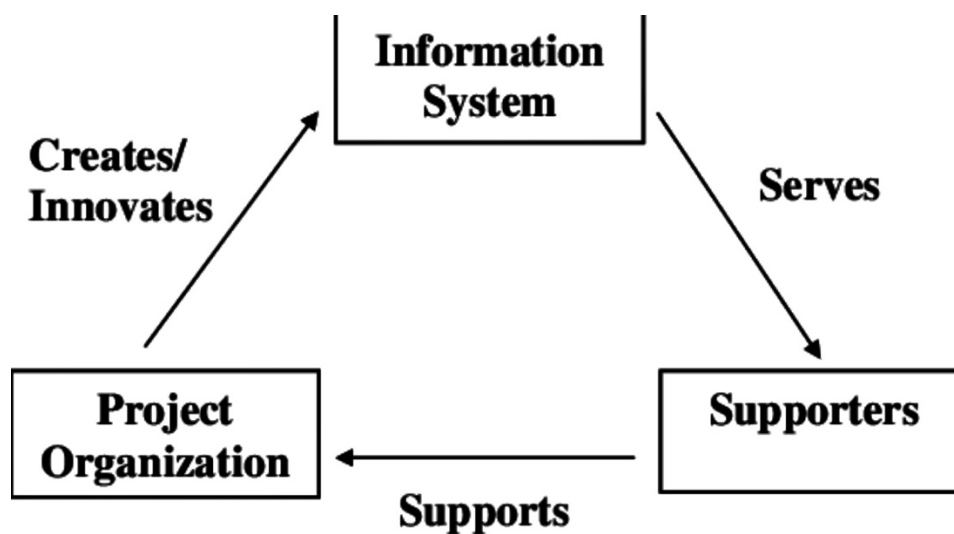


Figure 18: Sauer's conceptual framework for theorising about support and support management (Sauer, 1993)

Mandata was the name of the information system the Australian Public Sector Commission (APSC) envisioned in 1970 to “*automate the administration of personnel and establishments processes for the whole of the Australian Public Service*” (Sauer, 1993).

The complex project started in 1970, partially came to operation in 1977 and was eventually terminated in 1981 for being over time and budget that have been discussed in section 2.3. Always in the same paper mentioned above, Sauer wrote that

“what makes it particularly interesting for our present purpose is that support for the project varied considerably over its lifetime. At one stage the project was given a new lease of life by an injection of support while its final demise can be seen to have resulted in part from a chronic haemorrhage of support.”

Sauer’s paper was published in 1993 and as a result of several years of research and interviews carried out by the same author, together with the identification of a “*conceptual framework for theorising about support and support management*”, by which he clearly emphasises the important role of support and support management in the success or failure of any information systems.

In the following table, I summarise the main components of Sauer’s conceptual framework in order to highlight its proximity to my research questions regarding the role of politics and political relationships as crucial factors in any software development project.

Table 5: Summary of a framework for understanding support and support management

What do we mean by support?	Primary - funding, fixing, normal technical work, use Secondary - power-broking, promotion of social legitimacy, commitment
How does support work?	Provides necessary inputs Copes with flaws - overcomes them, absorbs them Modes - anticipatory, reactive
What are the bases of support?	Exchange of benefits for support, structural arrangements, cultural norms
What types of support management practice are there?	Structural, non-structural
What support management strategies are there?	Targeted, sprayspot, capture-a-champion, decouple system from support

Footprint of politics appears evident across the analysis Sauer provides of the Mandata failed project. For this reason, this case study was adopted as a suitable (almost straightforward) case-study to apply and test PEN and argue that, if PEN was available at the time, those playing the role of requirements engineers could have visualised politics in the Mandata project and act differently.

In the following section I will list all stakeholders and their role at each stage. After several times reading the article and having lengthy discussions within our team, the stakeholders and their roles in each stage was identified (5.2).

As for the second case study, the £10 Billion IT project, the National Program for IT (NPfIT) in the National Health Service (NHS) by the Department of Health has been identified. For this case-study, identification of politics might not have been as immediate as for the Mandata project: either way, I felt it was important to test PEN against case-studies that shared some features (significantly, they both ended up in failed projects) even though their “political dimension” could appear at different levels of “evidence”.

Also, I noticed that Dolfig (2019) analyses projects in the same spirit of this research, and I would argue it was appropriate to use his analysis as the second case study for this research, as it felt like we were working in parallel.

The goal of the system behind the NHS-DoH initiative was to improve the quality of the services across the board.

Dolfig believes that “*NPfIT in the National Health Service (NHS) was the largest public-sector IT program ever attempted in the UK, [...], the core aim of the NPfIT was to bring the NHS’ use of information technology into the 21st century*”.

The project was meant to link over 30,000 GPs to nearly 300 hospitals to have access to 50 million patients record. The project was running for nearly a decade before being finally cancelled by the government in September 2011, after many authors argued that time, resources and taxpayer's money were being wasted or poorly used (Randell, 2007; Mark, 2007; Brennan, 2009).

Maughan (2010) believed that *“Many of the lessons that can be learned from the failure of NPfIT are no more than common sense. Indeed, many of the mistakes have been obvious almost day 1 – but the lessons appear not to have been learned, or at least not until too late.”* Timelines and risk profile were some of the main factors to be blamed for the failed project.

Also, according to Justinia, (2017), *“the lack of adequate end user engagement, the absence of a phased change management approach, and understanding the scale of the project”* were among the reasons why the project was dismantled.

5.2 Case study 1 - Chris Sauer paper and the case study: impact evaluation on a consolidated study

The Mandata project has been analysed along the years by several practitioners and academics. Sauer worked extensively on this failed project, and by scrutinising its timeline he had the opportunity to test and validate his conceptual framework for understating the role of support in ensuring the success of a given IT project (Sauer, 1993).

I am hereby reporting the stages and the full list of stakeholders I have identified based on Sauer's work. I am then going to identify, stage by stage, the stakeholders involved and the PEN-based diagram one could derive by the initial analysis.

Stages:

Stage 1 = Initiation

Stage 2 = Initial Development

Stage 3 = Reorientation

Stage 4 = Consolidation

Stage 5 = Scale Back and Termination

Table 6: Identification of actors and their roles in based on 5 stages specified in Sauer's analysis

Stage	Actor id	Actor	Comment
1	1 A	Cabinet of the Commonwealth Government	To fund equipment and system software
1	2 B	Consultants	To help with writing report and approval process
1	3 C	The Board (shorter for Australian Public Service Board)	Funder, Chief sponsor, Powerbroker and Approve proposal before putting it to the Cabinet
1	4 D	Various specialist committees	Approve proposal before putting it to the Cabinet
1	5 E	The permanent heads of Public Service departments and statutory authorities	To support
1	6 F	The staff associations	To support
1	7 G	The public service Board's head of Management Services	Champion
1	8 G1	The second public service Board's head of Management Services	Less actively supportive manager
1	9 H	The ADP Departmental Systems Branch of the public service Board	Provided primary support and secondary support after new director appointed

2	10 I	The project organisation	Dependent on a number of other departments and agencies as below
2	11 J	Agency (The Australian Government Supply and Tender board)	For management of the formalities of the tender process
2	12 K	Agency (The National Capital Development Commission (NCDC))	For provision accommodation
2	13 L	Agency (The Department of Housing)	For site works
2	14 M	Agency (The Department of Services and Property)	Fitting out and maintaining Accommodation
2	15 H1	Head of ADP Division	Point of formal communication with board but slow to open the gate he kept
2	16 N	Consultative committee of middle managers from the user department	To build support among staff at levels lower than permanent head by encouraging commitment, and promoting legitimacy
2	17 O	Ministers	Supposedly acting on behalf of the taxpayer
2	18 P	The Auditor-General	Responsible for external criticism, but also secondary support
2	19 W	The Press	Responsible for external criticism
2	20 Q	Reviewer	Eventually generated more “disfunctional” secondary support
2	21 R	Project director	Direct access to the chairman of the Board and concomitantly greater Board support

3	22 S	The department of productivity's technical staff	Crucial fixer of applications problems
4	23 U	A new project director	<p>Responsible for the management of the 4th stage and known for maintaining a wide network of good links with the Board through regular formal/informal sessions.</p> <p>Interested in involvement of users at the management and supervisory levels.</p> <p>After external criticism, he called upon supporters to defend the project and he had support from The Board, own staff and by user departments.</p>
4	24 T	The Public Accounts Committee	External criticism, but also secondary supporter
5	25 X	Public	External criticism, but also implicit supporter (abandoning the project could create public political embarrassment)

Stage 1- Initiation

Table 7: Identification of actors and their roles in stage 1 as specified in C Sauer's analysis

Stage	Actor id	Actor	Comment
1	1 A	Cabinet of the Commonwealth Government	To fund equipment and system software
1	2 B	Consultants	To help with writing report and approval process
1	3 C	The Board (shorter for Australian Public Service Board)	Funder, Chief sponsor, Power broker and Approve proposal before putting it to the Cabinet
1	4 D	Various specialist committees	Approve proposal before putting it to the Cabinet
1	5 E	The permanent heads of Public Service departments and statutory authorities	To support
1	6 F	The staff associations	To support
1	7 G	The public service Board's head of Management Services	Champion
1	8 G1	The second public service Board's head of Management Services	Less actively supportive manager
1	9 H	The ADP Departmental Systems Branch of the public service Board	Provided primary support and secondary support after new director appointed

The full analysis from Sauer (1993) can be found in Appendix D.

PEN-based analysis of Stage 1

The first model indicates that under the support of the first head of management services for the public service board (G, the champion/supporter), the project started relatively smooth and was moving to the right direction. *“The project only had verbal support from the permanent heads of public service department and statutory authorities (E) and the staff association (F) This is what we referred to as “anticipatory support” which is the project's backer's view rather than the actual state of support by some of the stakeholders”.*

As a result, some actors’ emotional status (for A, E and F) has been ignored, simply because we do not know how they felt. However, the consultants (B) and the Australian Public Service Board (C, The Board) were either fine or pretended to be fine with the progress as the proposal were getting ready to be sent to Various Specialist Committees (D), before it could be put to the Cabinet of Commonwealth Government (A).

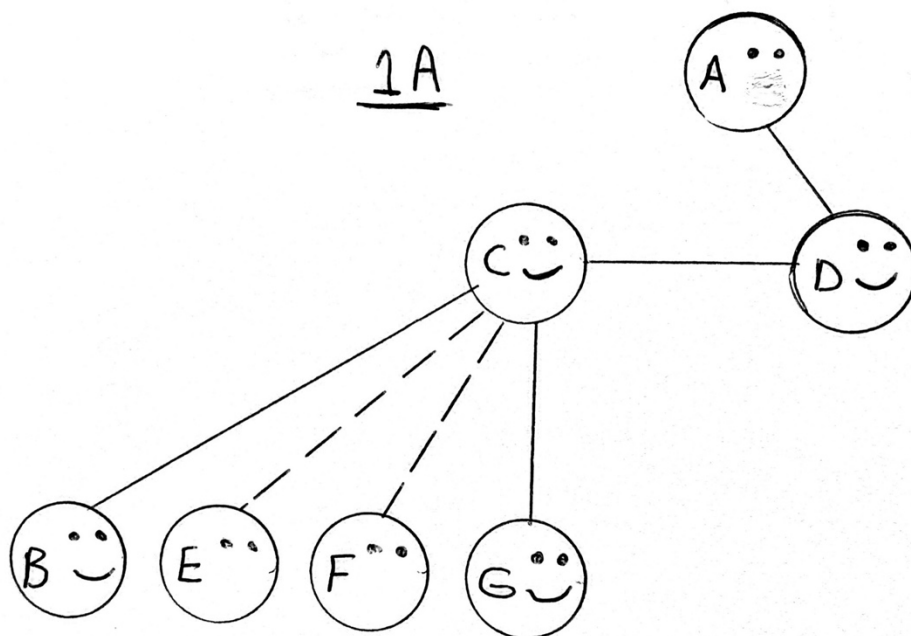


Figure 19: Case study 1 - Model 1A

Happy face: 4
Sad face: 0
Neutral: 0
Don't know: 3
Total: 7

Ratio

Happy: 0.57
Sad: 0
Neutral: 0
Don't know: 0.43

The second model (1B), still representing the project before February 1972, highlights the first head of management services for the Public Service Board (G) becoming a supporter (or “champion”!) for the project. In the same model, an informal link between G and the Cabinet of Commonwealth Government (A) for approval assurance is included. As it has not been stated otherwise in the analysis done by Sauer, and as this is an example based solely on Sauer’s description, it could still be assumed (like in the previous section) that the support from the permanent heads of Public Service Department and Statutory Authorities (E) and the Staff Association (F) was only verbal and not definite, and so it was not a type of support that could be relied on.

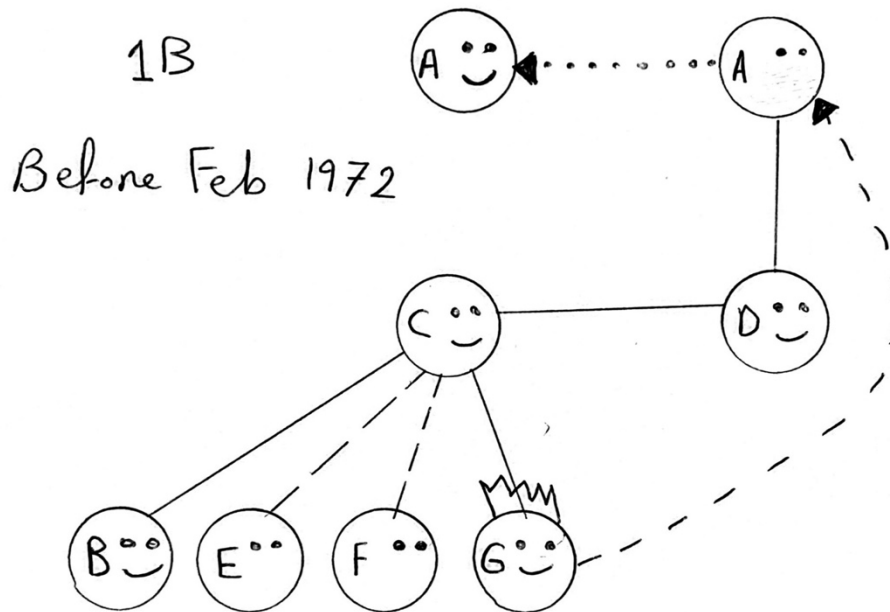


Figure 20: Case Study 1 - Model 1B

Happy face: 5
 Sad face: 0
 Neutral: 0
 Don't know: 2
 Total: 7

Ratio

Happy: 0.71
 Sad: 0
 Neutral: 0
 Don't know: 0.29

The third model, capturing the project after February 1972, illustrates the situation after the retirement of the public service Board's head of Management Services (G) who was replaced by G1, "a less actively supportive manager". The informal link between the head of management services for the public service board and the cabinet was removed. Again, as it has not been stated otherwise in the analysis done by Sauer, and as this is an example based solely on Sauer's description, it could still be assumed that the other actors' emotional status remains the same as in the previous model.

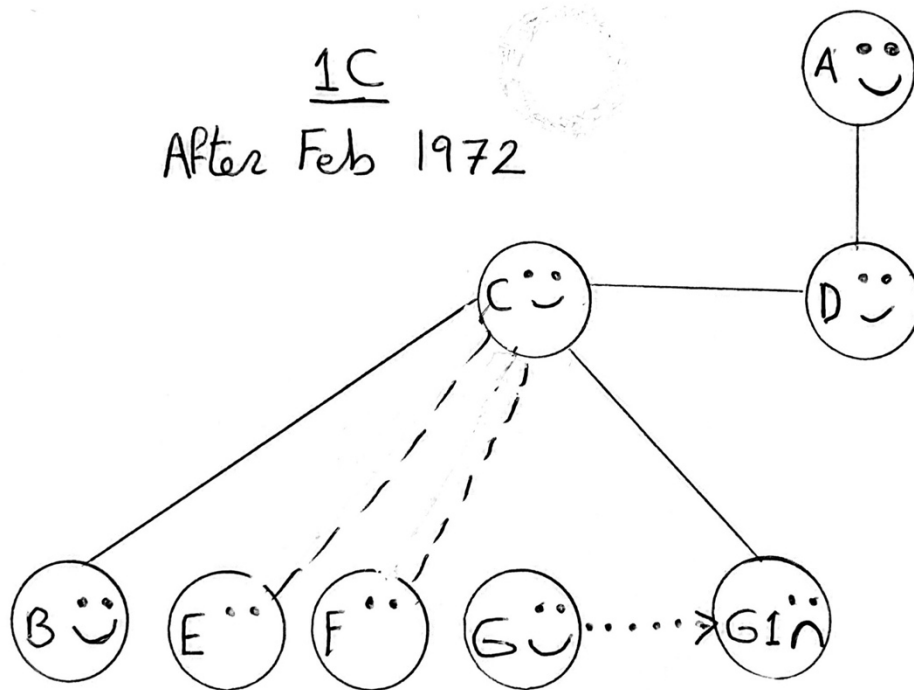


Figure 21: Case study 1 - Model 1C

Happy face: 4
 Sad face: 1
 Neutral: 0
 Don't know: 2
 Total: 7

Ratio

Happy: 0.57
 Sad: 0.14
 Neutral: 0
 Don't know: 0.29

In the final model (1D) for the initiation stage (stage 1), the loss of the project's champion (G), who has been replaced by (G1), has been combined with the fact that there was no primary or secondary support apart from general approval from the permanent heads of Public Service Department and Statutory Authorities (E) and the Staff Association (F). This is when the lateness of the Staff Association was given as one reason for the delays in putting forward the Cabinet (A) submission. Although the project at this stage has been eventually approved, the Cabinet (A) only supplied some equipment and not adequate staff or accommodation.

Given the partial resourcing and the dissatisfaction of the permanent heads of Public Service Departments and Statutory Authorities (E) and the Staff Associations (F) at the end of the 18 months delay, forecast for the success of the project did not look bright at that point in time.

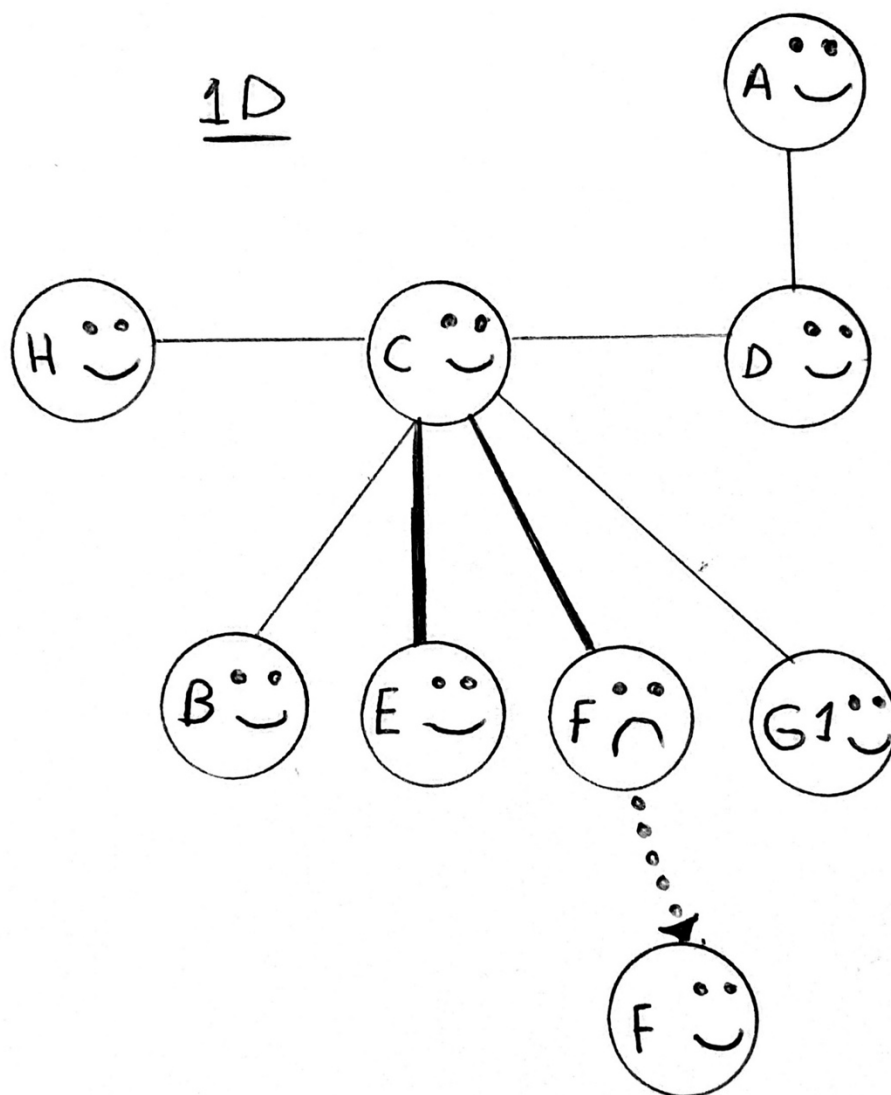


Figure 22: Case study 1 - Model 1D

Happy face: 8
 Sad face: 0
 Neutral: 0
 Don't know: 0
 Total: 8

Ratio

Happy: 1
 Sad: 0
 Neutral: 0
 Don't know: 0

Stage 2 - Initial Development

Table 8: Identification of actors and their roles in stage 2 as specified in C Sauer's analysis

Stage	Actor id	Actor	Comment
2	10 I	The project organisation	Dependent on a number of other departments and agencies as below
2	11 J	Agency (The Australian Government Supply and Tender board)	For management of the formalities of the tender process
2	12 K	Agency (The national Capital Development Commission (NCDC))	For provision accommodation
2	13 L	Agency (The Department of Housing)	For site works
2	14 M	Agency (The Department of Services and Property)	Fitting out and maintaining Accommodation
2	15 H1	Head of ADP Division	Point of formal communication with board but slow to open the gate he kept
2	16 N	Consultative committee of middle managers from the user department	To build support among staff at levels lower than permanent head by encouraging commitment, and promoting legitimacy

2	17 O	Ministers	Supposedly acting on behalf of the tax payer
2	18 P	The Auditor-General	Responsible for external criticism, but also secondary support
2	19 W	The Press	Responsible for external criticism
2	20 Q	Reviewer	Eventually generated more “disfunctional” secondary support
2	21 R	Project director	Direct access to the chairman of the Board and concomitantly greater Board support

PEN-based analysis of Stage 2

The proposal to the Cabinet of the Commonwealth Government (A) has been drafted and indicated the funding for the technology, staff and accommodation that was needed for the project. Tasks were to be carried out by other organisations that were out of control of the project organisation itself (I), like:

- The Australian Government Supply and Tender Board (J) for management of the formalities of the tender process,
- The National Capital Development Commission (NCDC) (K) for provision of accommodation,
- The Department of Housing (L) for site works
- The Department of Services and Property (M) for fitting out and maintaining accommodation.

I could not derive the emotional status of all stakeholders in this model at the beginning of this second stage, as it has not been mentioned by Sauer’s analysis: given there is no indication or mentioning of sadness or disappointment as the support from the management was concerned, it is safe to assume there was still some hope for the approval of funding for all the costs e.g., accommodation by the Cabinet (A).

However, later on (but always within the initial development stage) it becomes clearer that the Cabinet intends to fund the project only partially.

The Board (C) decides to be a funder and powerbroker, trying to meet a sudden, new requirement (“*As the stage progressed and it became apparent that not all the support needed was forthcoming, there became a requirement for power-broking*”), which could possibly mean that the lack of a champion and/or of power-broking is starting to become more obvious.

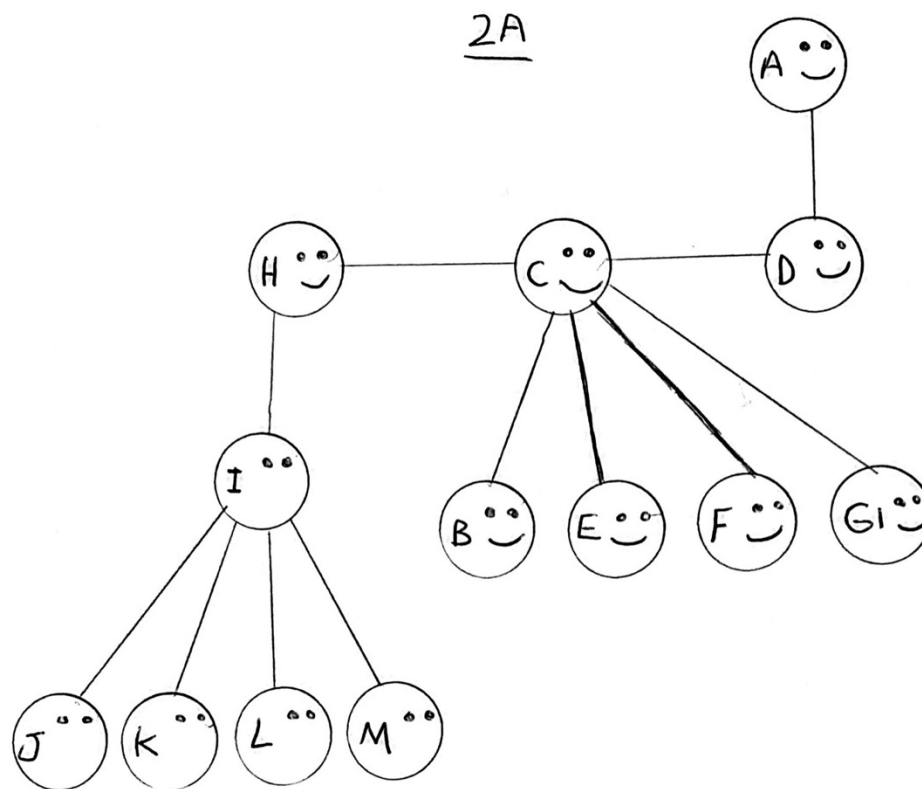


Figure 23: Case study 1 - Model 2A

Happy face: 8
 Sad face: 0
 Neutral: 0
 Don't know: 5
 Total: 13

Ratio

Happy: 0.62
 Sad: 0
 Neutral: 0
 Don't know: 0.38

The communication between the project organisation and the Board was now occurring through the head of ADP (H1), in a more formal manner than previously. The Consultative Committee of Middle Managers from the User Department (N) became established as management of secondary support (H) to gain support from staff at lower level.

The support for some processes, like the tender process (J1), was satisfactory, if compared with other processes with less support. Lack of champion become more obvious as it was difficult to access or have any influence over the Board. In general, the emotional status was changing due to the economic difficulties faced by the Board (C) and other departments, and due to the project being under-staffing.

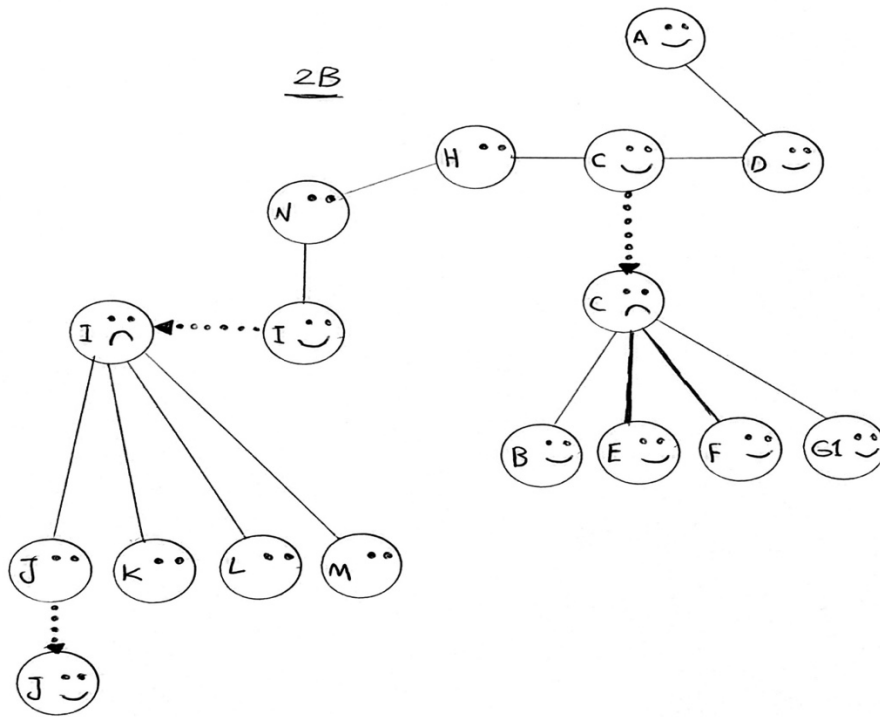


Figure 24: Case study 1 - Model 2B

Happy face: 7
 Sad face: 2
 Neutral: 0
 Don't know: 5
 Total: 14

Ratio

Happy: 0.50
 Sad: 0.14
 Neutral: 0
 Don't know: 0.36

As the Cabinet only provided partial funding for equipment and system software and not for all the other costs, the NCDC (K1) stopped their site-planning for the central computer. As a result, other ministers (O) got involved and tried to solve the problem. The project was still receiving lot of support, but it was no longer sufficient: even design of the application received less attention, and this led to serious delays in the overall development of the whole system. No promises were made to the Board (C) for the implementation date, due to lack of resources.

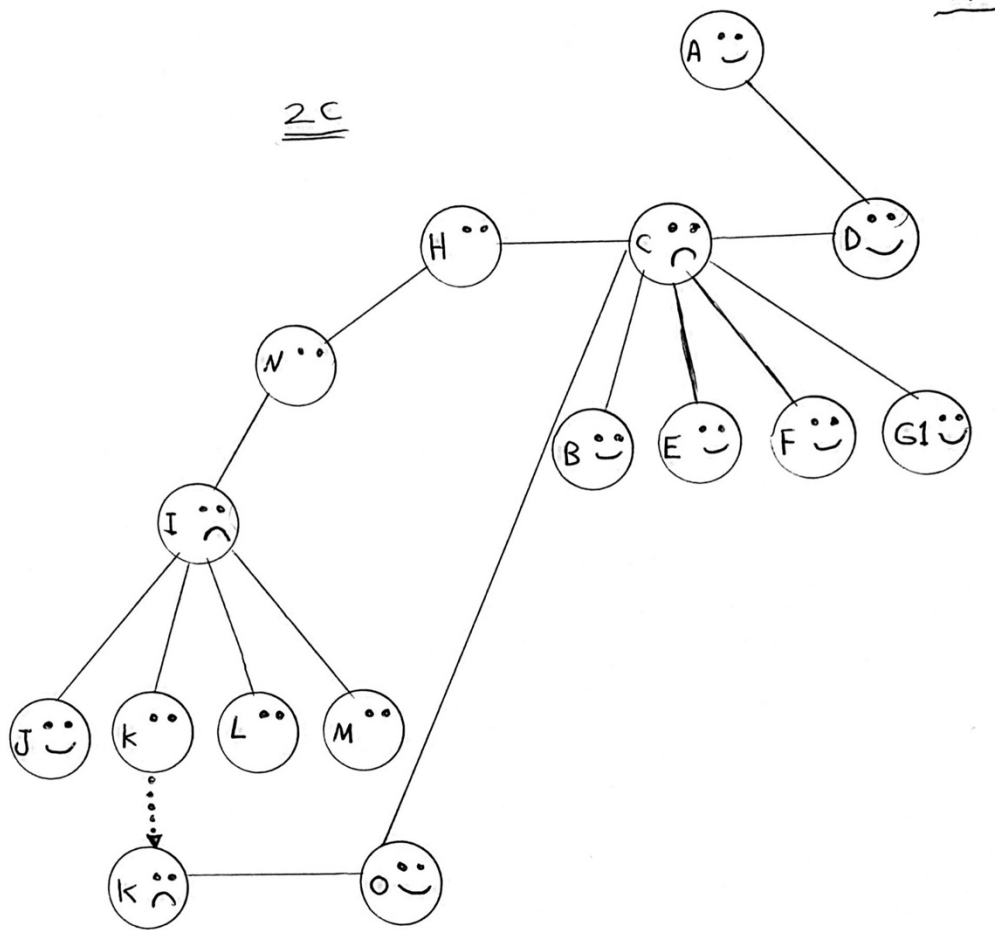


Figure 25: case study 1 - Model 2C

Happy face: 8
 Sad face: 3
 Neutral: 0
 Don't know: 4
 Total: 15

Ratio

Happy: 0.53
 Sad: 0.20
 Neutral: 0
 Don't know: 0.27

Project organisation (I) and the project as a whole started receiving external criticism and close watching from the Auditor-General (P) and the press (W) over the following 4 years and a political crisis had been created for the Board (C).

It was the decision time to either call off its support for the project completely or to inject even more support.

A new strategy, suggested by a reviewer (Q), was agreed among stakeholders. Project director (R) was also appointed at higher level to give direct access to the chairman of the Board, and this could give some hope for more support from the Board (C). Still, the new strategy proved not to be adequate and at that point the Board could not avoid any longer to make some essential decision for the future of the project.

The Board (C) decided to continue its support, although it had to justify the investments made and were going to review progress in 3 years. Sauer (1993) believes that *“for the researcher, the problem of explaining why the Board decided to continue its support remains outstanding.”*

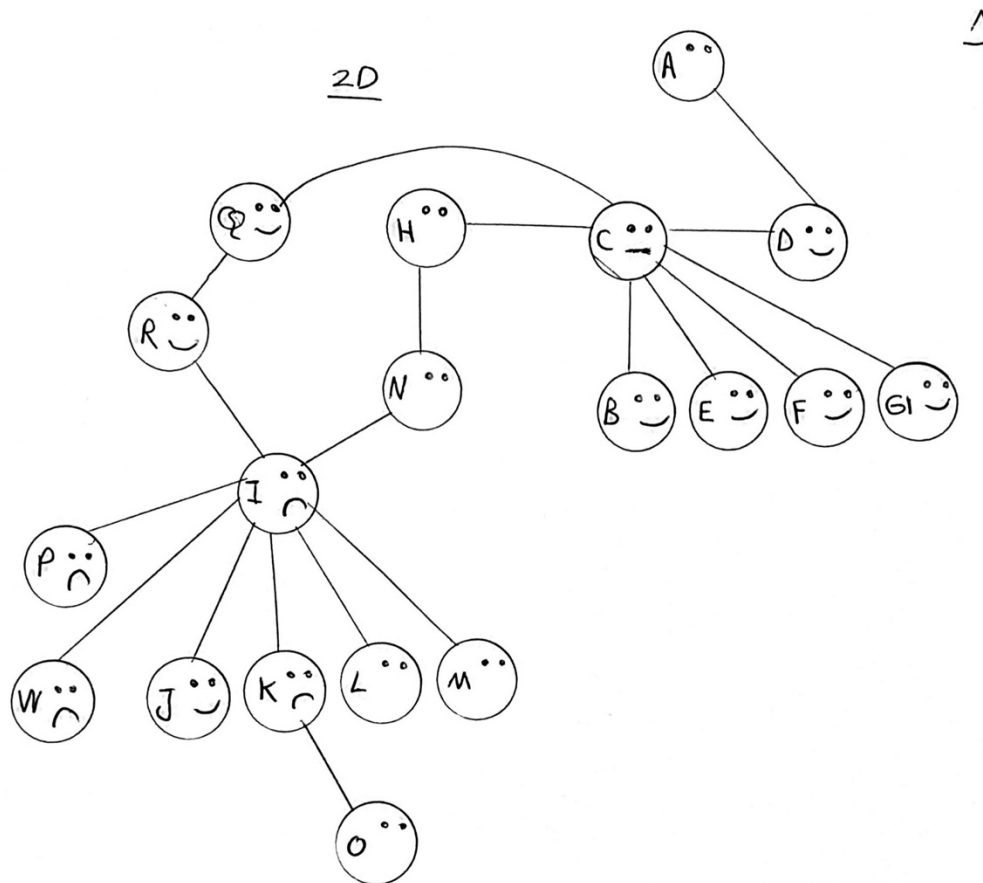


Figure 26: Case study 1 - Model 2D

Happy face: 8
 Sad face: 4
 Neutral: 1
 Don't know: 6
 Total: 19

Ratio

Happy: 0.42
 Sad: 0.21
 Neutral: 0.05
 Don't know: 0.32

Stage 3 – Reorientation

Table 9: Identification of actors and their roles in stage 3 as specified in C Sauer's analysis

Stage	Actor id	Actor	Comment
3	22 S	The Department of Productivity's technical staff	Crucial fixer of applications problems

PEN-based analysis of Stage 3

As mentioned in the previous model (Model 2D), the project was now under external scrutiny, pressure and criticism from different sources, like the Auditor-General (P) and the press (W), and this lasted for several years, for a number of reasons (for example, there was still missing sufficient computer accommodation).

As a result of all these pressures, a new strategy was devised by a group of stakeholders, consisting of the project organisation (I), the Board (C) and the technical staff from the Department of Productivity (S). The newly adopted strategy looked at the early implementation and permitted the project to be "in the hands" of the technical staff, who could assist fixing some technical problems and, at the same time, enabled the project to gain users' support, by involving them in the project.

Also, this proved to be beneficial to the Board (C) as it stopped their reputation being further damaged. However, primary, secondary and power-broking support was still needed for the adopted strategy to work. As mentioned in the previous stage 2 (which Sauer denominates "Orientation stage"), a newly appointed project director (R), who had access to the Chairman of the Board (C), was placed to push for even more funding, useful for hiring staff, and power-broking support for new office accommodation from the Board (C), even if the project still was "under resourced".

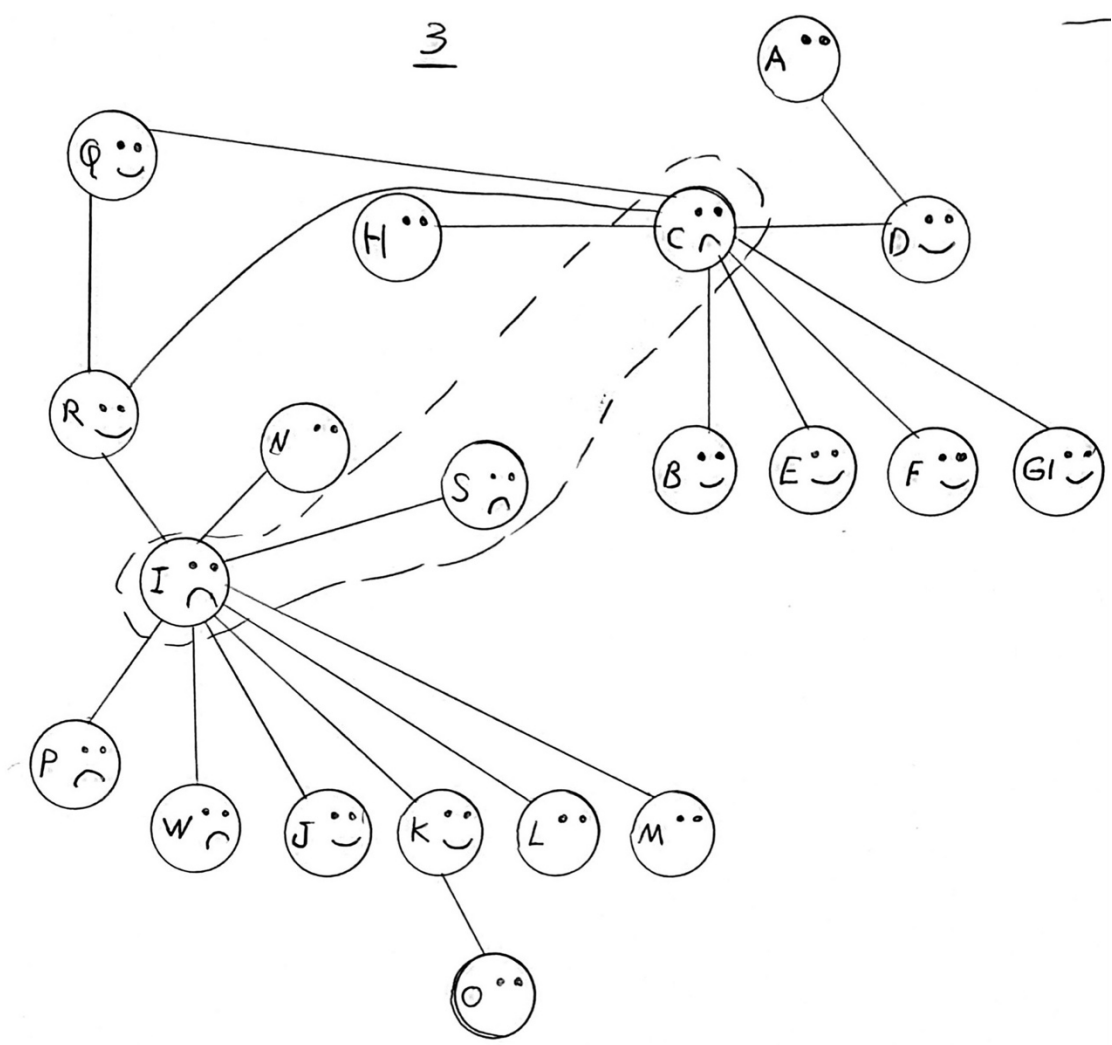


Figure 27: Case study 1 - Model 3

Happy face: 9
 Sad face: 5
 Neutral: 0
 Don't know: 6
 Total: 20

Ratio

Happy: 0.45
 Sad: 0.25
 Neutral: 0
 Don't know: 0.30

Despite all the above-mentioned challenges and rejection of some further support, Sauer still believes that “the project was mostly well supported during this [third] stage”, which Sauer

names “*Reorientation stage*”. This is not to say that there were not some disruptions, which Sauer hinted at when he mentioned about the “*political difficulties the Board*” was facing.

Stage 4 - Consolidation

Table 10: Identification of actors and their roles in stage 4 as specified in C Sauer’s analysis

4	23 U	A new project director	Responsible for the management of the 4 th stage and known for maintaining a wide network of good links with the Board through regular formal and informal sessions. Interested in involvement of users at the management and supervisory levels. After external criticism, he called upon supporters to defend the project and he had support from The Board, own staff and by user departments.
4	24 T	The Public Accounts Committee	External criticism, but also secondary supporter

PEN-based analysis of Stage 4

The Consolidation stage (stage 4) is a follow up of the Reorientation stage (stage 3) to bring the new strategy to life. Sauer indicates that the responsibility of managing the consolidation stage stands with a new project director (U), even if it is not clear whether he refers to the new project director, who was introduced in the initial development stage (stage 2), or this is another new project director. As it has been mentioned as “a new director” instead of “the new director”, I assume that this is a different new director. Sauer, in this stage, explains what the project director was hired for (“*maintaining a wide network of contacts throughout the Public Service*”), also because of the good formal and informal relationships existing between the new project director (U) with the board (C), and because he was involving and engaging the users through the two committees.

Rather than focusing on the reassessment of the Mandata computer system and possibly its improvement, the new project director proved to be able to secure both primary and secondary support, so to keep the project afloat whilst mitigating any growing criticism and scrutiny: he asked the supporters of the project to defend the project while it was under fire from the Auditor-General (P) (who was also responsible for external criticism in the initial development stage, or Stage 2) and then from the Public Accounts Committee (T). Although the project received enough support from “*the Board, the project’s own staff, and by user department*”, as the conversion to Mandata was delayed and for a short period the users had to run clerical systems in parallel, once again the project went under scrutiny from the Auditor-General, the Public Accounts Committee (T), the press (W) and the staff associations (F). Even the full support from the Board (C) could only moderate the situation.

Sauer concludes that

“overall, then, the support given to the project organisation was sufficient to sustain continuing development, operation and maintenance, but it was not sufficient to permit eradication of all flaws and shortcomings nor to generate the supportive context necessary to the development of the original system.”

However, as the system was unable to be fully operational, the critics from the Auditor-General, The Public Accounts Committee (T) and by press continued and the legitimacy of the project was seriously undermined. Interestingly, whilst the Public Accounts Committee (T) continued to provide their support, it looks it was eventually the project organisation (I) that was unable to offer their commitment to the Mandata project: this continued to be funded until the late 70s, with little hope that it could survive the 80s.

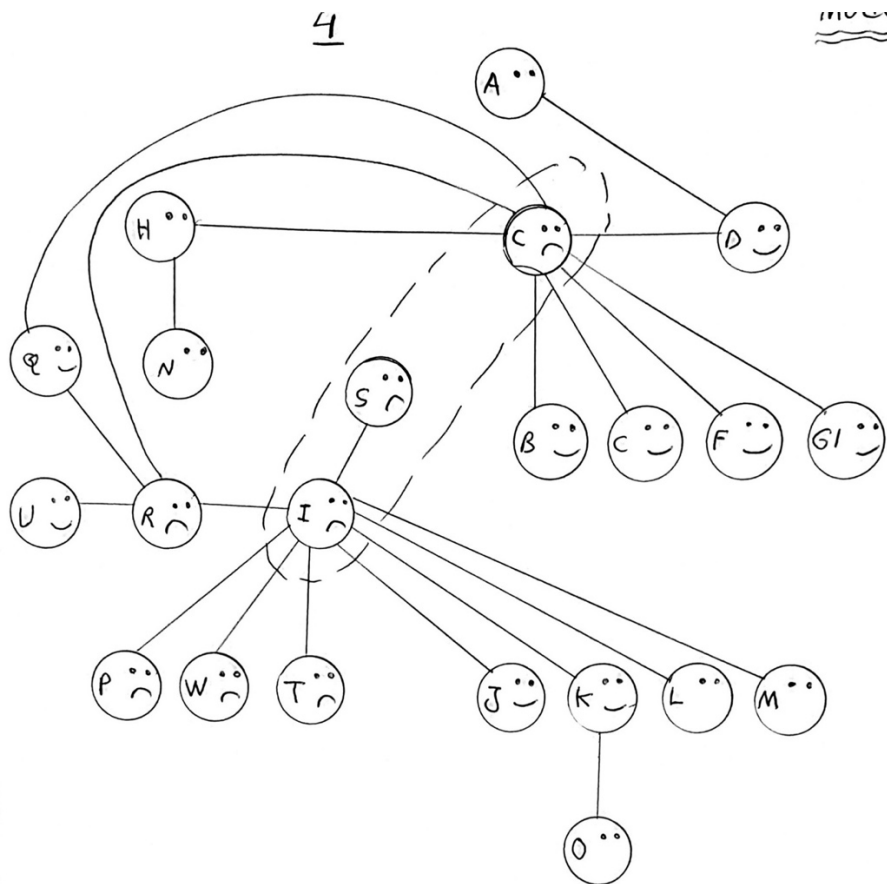


Figure 28: Case Study 1 - Model 4

Happy face: 9
 Sad face: 7
 Neutral: 0
 Don't know: 6
 Total: 22

Ratio

Happy: 0.41
 Sad: 0.32
 Neutral: 0
 Don't know: 0.27

Stage 5 – Scale Back and Termination

Table 11: Identification of actors and their roles in stage 5 as specified in C Sauer's analysis

5	25 X	Public	External criticism, but also implicit supporter (abandoning the project could create public political embarrassment)
---	---------	--------	--

PEN-based analysis of Stage 5

In the first half of the 80s, some of the departments that previously supported the project, for example the Auditor-General's staff and the departmental users, realised that there was little scope in the existing system to be extended extension and no hope it could be used for redeveloping purposes.

The project still required support and relied on users' help even when eventually the plan was to scale back. At that time the Public's (X) criticism rose too. As a result of negotiation between the Board (C) and users with the Auditor-General (P), the Auditor-General agreed to further extend their support. As a result of Government cut, the project organisation eventually admitted it was unable to support the full (re-)development and deployment of the Mandata system and terminating the project with such an excuse suddenly became the best scenario for the Board (C) who was trying to get away from it was an embarrassing situation.

Summary of the first case study

As part of the analysis of the Mandata project case study, a first set of draft emoji models for all the five stages as identified by Sauer (1993) have been drawn, with the intention of showing how the notation works and how it can help practitioners to understand political relationships, without the need for much digging into the history of the project itself.

Also, it is important to underline that the models have been based solely on Sauer's analysis, which could be a partial analysis.

In the initiation stage (stage 1), under the support of the first head of management services for the Public Service Board (G, the champion/supporter) and their informal link with the Cabinet of the Commonwealth Government (A), it was hoped for the project to start smoothly and to move to the right direction. Even at such an early stage, Maughan (2010) believes that there were some clear hints upon which to forecast the failure of the project, for example that *“top-down projects are much more likely to fail than bottom-up projects”*.

After the replacement of the first head of management services for the Public Service Board (G) and the elimination of their informal link with the Cabinet (A), lack of adequate staff, a not

yet identified accommodation of the system to be developed and 18 months delay in putting forward the proposal to the Cabinet, the project seemed already looking not too promising. During the initial development stage (stage 2), level of support remained the same as in the previous stage. Whilst some stakeholders were hoping for a full funding towards all the expected costs of the project, later on in this stage, it became clearer that the Cabinet (A) was keen to fund the project only partially. The Board (C) decided to become themselves a funder and powerbroker for the project. But not having any longer a champion the project made it more difficult to access or have influence over the Board.

The project was still receiving plenty of support, but it became evident this was not sufficient for covering the whole project in both its direct and indirect costs. Due to the financial problems, Ministers (O) got involved to “solve” the problem. The Project Organisation (I), and the project as a whole, had started receiving external criticism and close watching from the Auditor-General (P) and the press (W) over the next 4 years. A political crisis invested the Board (C): it was time to decide either to call off its support for the project completely or to inject even more support. The Board (C) decided to continue its support.

The reorientation stage (stage 3) started among continuing external pressures and criticism from different sources. As a result of all these pressures, a new strategy was identified: they looked back at the early implementation efforts and decided the project should have been in the hands of the technical staff, who could not only assist with fixing the several IT problems and flaws the system was affected by, but also to enable the project to gain support from the system “natural” users (at the time it was expected technical staff to be the main “user” of the Mandata computer system!) by involving them in the project. Still, some disruptions were mentioned in this stage too.

A consolidation stage (stage 4) followed the reorientation stage (stage 3) to bring the new strategy to life. Although the project received some significant support from “*the Board, the project's own staff, and by the user department*”, a full conversation and commitment to Mandata was delayed and for a short period all users had to run the two clerical systems (i.e., Mandata and the traditional system Mandata was supposed to replace) in parallel.

As a consequence, once again the project was under fire from the Auditor-General, the Public Accounts Committee (T), the press (W) and the Staff Associations (F). Even the full support from the Board (C) could only mitigate the adversary circumstances. The project continued until the late 70s with little hope that the project could be continued in the 80s.

In the first half of the 80s, some of the departments that previously supported the project, like the Auditor-General's staff and the departmental users, realised that there was little hope in the Mandata system to be capable of some “extension” and at the same time no hope or support at all for its full redevelopment. The project still required support and relied on users’ help in order to scale back, but at the same time public’s (X) criticism rose. As a result of Government cuts, the project organisation was eventually unable to secure support any further and terminating the project with this excuse was the best scenario for the Board (C) to escape from a very embarrassing situation.

5.3 Case study 2 - the National Program for IT (NPfIT) in the National Health Service (NHS)

As for the second case study, the National Program for IT (NPfIT) in the National Health Service (NHS), also known as the “£10 billion IT project” (Dolfing, 2019), run by the Department of Health of the British government, has been identified.

The content provided here relies wholly on the work of (Dolfing, 2019), and it’s also conceivable that due to political motivations, individuals might be withholding their perspectives and choosing to remain silent about their opinions. One could infer that their silence suggests unhappiness, as one would expect them to voice their contentment if they were indeed unhappy.

For this case-study, identification of politics might not have been as immediate as for the Mandata project: either way, I felt it was important to test PEN against case-studies that shared some features (for example, they both ended up in failed projects!) even though their “political dimension” could appear at different levels of evidence.

Also, I noticed that Dolfing (2019) analyses projects in the same spirit of this research, and I would argue it was appropriate to use his analysis as the second case study for this research, as it felt like we were working in parallel.

The goal of the system behind the NHS-DoH initiative was to improve the quality of the services across the board. Dolfing (2019) believes that “NPfIT in the National Health Service (NHS) was the largest public-sector IT program ever attempted in the UK, [...], the core aim of the NPfIT was to bring the NHS’ use of information technology into the 21st century”.

The project, which aimed at linking over 30,000 GPs to nearly 300 hospitals to have access to 50 million patients record, was running for nearly a decade before being finally cancelled by a new UK government in September 2011, after many authors argued that time, resources and taxpayer’s money were being wasted or poorly used (Randell 2007; Mark, 2007; Brennan, 2009).

Maughan (2010) suggested that

“many of the lessons that can be learned from the failure of NPfIT are no more than common sense. Indeed, many of the mistakes have been obvious almost day 1 – but the lessons appear not to have been learned, or at least not until too late.”

Timelines and risk profile were only some of the main factors to be blamed for the failed project. Also, according to Justina, (2017), *“the lack of adequate end user engagement, the absence of a phased change management approach, and understanding the scale of the project”* were among the reasons why the project was dismantled.

Similarly, to what it was done for the Mandata project (the first case study above), also for this case study I have identified all stakeholders and/or actors involved, and I then developed a number of PEN - based models for the analysis of identified political relationships.

Table 12: Identification of actors and their roles between 2002 to 2009 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

	Actor id	Actor	Comment
2002	A	Government	Announced by PM Government prestige staked on success
	B	NHS (The National Health Service)	
	C	NHS IT Director (Richard Granger)	Appointed to run the project as well as the other thing
2003	D	BT awarded contract for the national data spine	Prime contractor
	E	CSC – for North West and West Midland cluster	Split responsibilities for different local service provider
	D1	BT capital care Alliance for London cluster	
	F	Fujitsu for southern cluster	
	G	Accenture for North East and eastern England clusters	
2004	D2	BT – NHS Broadband Network – N* contract	Reinforce prime contractor status
2005	H	NHS connecting for health (NHS CFH) set up to deliver NPfIT	
	I	Martin Bellamy – director of programmes and system delivery NHS CFH	
2006		Accenture withdraws as local service provider	
	E1	CSC awarded 9 years contract for Accenture’s former clusters	
2007		NPfIT local ownership programme (devolves responsibility for local delivery of the program from NHS CFH to groupings of strategic health authorities.) Replaces original five clusters with three program areas: Southern (local service provider Fujitsu), London (local service provider BT), and North, Midlands and East (local service provider CSC). Contract reset 2 (BT) for “best of breed” London solutions	
2008	K	Fujitsu contract for local service provider in Southern area terminated; legal dispute continues	
		Contract reset negotiations 3 (BT) for new delivery model in London	
		Richard Granger, head of NHS CFH, leaves in January Christine Connelly and Martin Bellamy appointed to jointly lead NHS CFH in September	
2009	J	BT awarded additional contract to take over eight trusts formally with Fujitsu, plus 25 trusts for RIO and four additional acute trusts in Southern area. Other southern trusts given choice of local service provider solution from BT or CSC or from various suppliers in additional supply capability and capacity list (ASCC)	
	K1	Christine Connelly – NHS CFH is integrated with department of health informatics directorate	
2010	A1	New memorandum of agreement signed between BT and NHS CFH, including reduced number of deployments in acute trusts in London Contract discussions with CSC continuing UK general election in May – new coalition government	
2011		The National Programme for IT has finally come to an end, although the bill for the enormously expensive and controversial project will continue to be paid for years to come.	
		The deadline to exit NPfIT national contracts in the North, Midlands and East passed on 7 July, marking the end of the final chapter of the £12.7 billion attempt to bring the NHS into the digital age. Around £2.6 billion of actual benefits had been identified as of March 2011.	

2002:

Table 13: Identification of actors and their roles in 2002 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

Year	Actor id	Actor	Comment
2002	A	Government	Announced by PM Government prestige staked on success
	B	NHS (The National Health Service)	
	C	NHS IT Director (Richard Granger)	Appointed to run the project as well as the other thing



Figure 29: Case study 2 - Model for year 2002

Happy face: 3
 Sad face: 0
 Neutral: 0
 Don't know: 0
 Total: 3

Ratio

Happy: 1
 Sad: 0
 Neutral: 0
 Don't know: 0

2003:

- BT awarded contract for the national data spine
- Local Service provider 10 years contracts awarded (CSC for North West and west Midland cluster; BT Capital Care Alliance for London cluster; Fujitsu for southern cluster; Accenture for North East and Eastern England clusters).

Table 14: Identification of actors and their roles in 2003 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

Year	Actor id	Actor	Comment
2003	D	BT	Prime contractor
	E	CSC – for north west and west	Split responsibilities for different local service provider
	D1	BT capital care Alliance for London cluster	
	F	Fujitsu for southern cluster	
	G	Accenture for north east and eastern England cluster	

Figure 30: Case study 2 - Model for year 2003

Happy face: 8
 Sad face: 0
 Neutral: 0
 Don't know: 0
 Total: 8

Ratio

Happy: 1
 Sad: 0
 Neutral: 0
 Don't know: 0

2004:

- BT awarded N* (NHS broadband network) contract

Table 15: Identification of actors and their roles in 2004 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

Year	Actor id	Actor	Comment
2004	D2	BT – NHS Broadband Network – N* contract	Reinforce prime contractor status

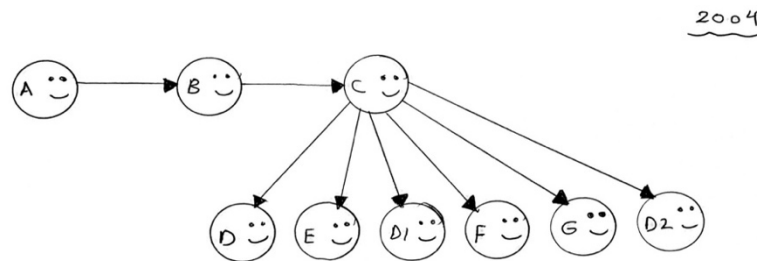


Figure 31: Case study 2 - Model for year 2004

Happy face: 9
 Sad face: 0
 Neutral: 0
 Don't know: 0
 Total: 9

Ratio

Happy: 1
 Sad: 0
 Neutral: 0
 Don't know: 0

2005:

- NHS Connecting for Health (NHS CFH) set up to deliver NPfit
- Contract reset 1 (BT) for “interim solutions” in London

Table 16: Identification of actors and their roles in 2005 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

Year	Actor id	Actor	Comment
2005	H	NHS connecting for health (NHS CFH) set up to deliver NPfIT	
	I	Martin Bellamy – director of programmes and system delivery NHS CFH	

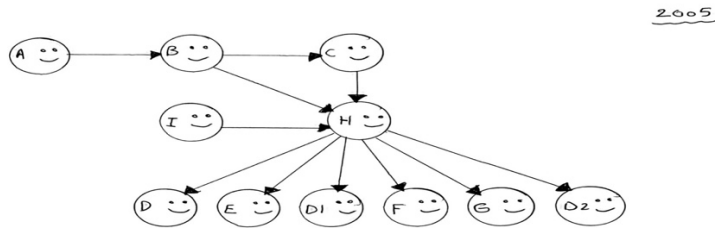


Figure 32: Case study 2 - Model for year 2005

Happy face: 11
 Sad face: 0
 Neutral: 0
 Don't know: 0
 Total: 11

Ratio

Happy: 1
 Sad: 0
 Neutral: 0
 Don't know: 0

2006:

- Accenture withdraws as local service provider
- CSC awarded 9 years contract for Accenture’s former clusters

Table 17: Identification of actors and their roles in 2006 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

2006	E1	Accenture withdraws as local service provider CSC awarded 9 years contract for Accenture’s former clusters	
------	----	---	--

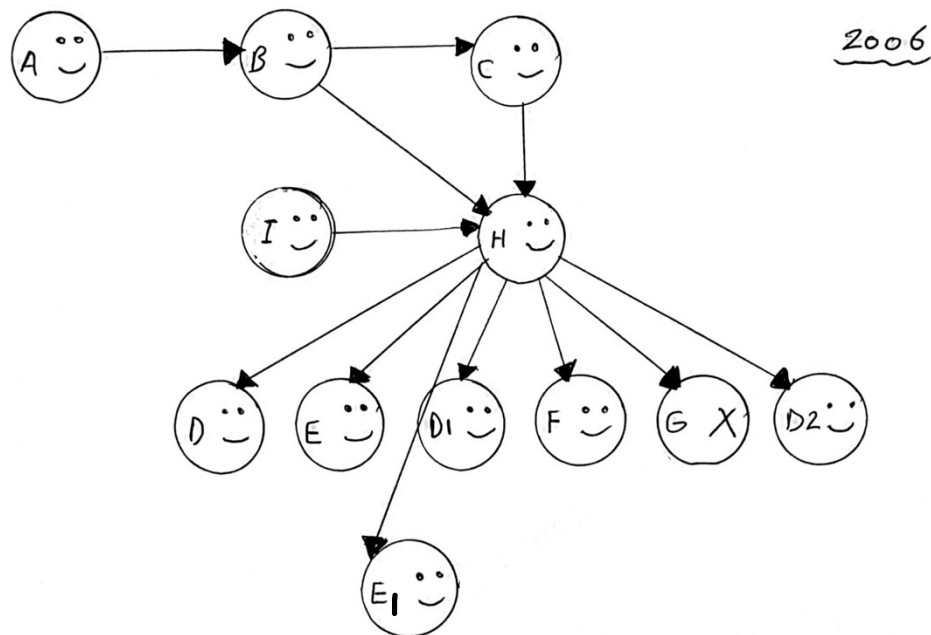


Figure 33: Case study 2 - Model for year 2006

Happy face: 11
 Sad face: 0
 Neutral: 0
 Don't know: 0
 Total: 11

Ratio

Happy: 1
 Sad: 0
 Neutral: 0
 Don't know: 0

2007:

- NPfIT Local Ownership Programme (devolves responsibility for local delivery of the program from NHS CFH to groupings of strategic health authorities)
- Replaces original five clusters with three program areas: Southern (local service provider FUJITSU), London (local service provider BT), and North, Midlands and East (local service provider CSC). The actors are remaining the same
- Contract reset 2 (BT) for “best of breed” London solutions

Table 18: Identification of actors and their roles in 2007 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

2007		<p>NPfIT local ownership programme (devolves responsibility for local delivery of the program from NHS CFH to groupings of strategic health authorities.)</p> <p>Replaces original five clusters with three program areas: Southern (local service provider Fujitsu), London (local service provider BT), and North, Midlands and East (local service provider CSC).</p> <p>Contract reset 2 (BT) for “best of breed” London solutions</p>	
------	--	--	--

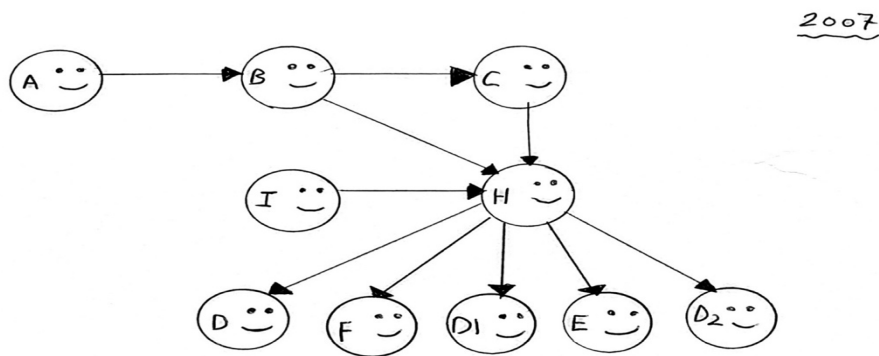


Figure 34: Case study 2 - Model for year 2007

Happy face: 10
 Sad face: 0
 Neutral: 0
 Don't know: 0
 Total: 10

Ratio

Happy: 1
 Sad: 0
 Neutral: 0
 Don't know: 0

2008:

- Fujitsu contract for local service provider in Southern area terminated; legal dispute continues
- Contract reset negotiations 3 (BT) for new delivery model in London
- Richard Granger, head of NHS CFH, leaves in January
- Christine Connelly and Martin Bellamy appointed to jointly lead NHS CFH in September

Table 19: Identification of actors and their roles in 2008 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

2008	K	<p>Fujitsu contract for local service provider in Southern area terminated; legal dispute continues</p> <p>Contract reset negotiations 3 (BT) for new delivery model in London</p> <p>Richard Granger, head of NHS CFH, leaves in January</p> <p>Christine Connelly and Martin Bellamy appointed to jointly lead NHS CFH in September</p>	
------	---	---	--

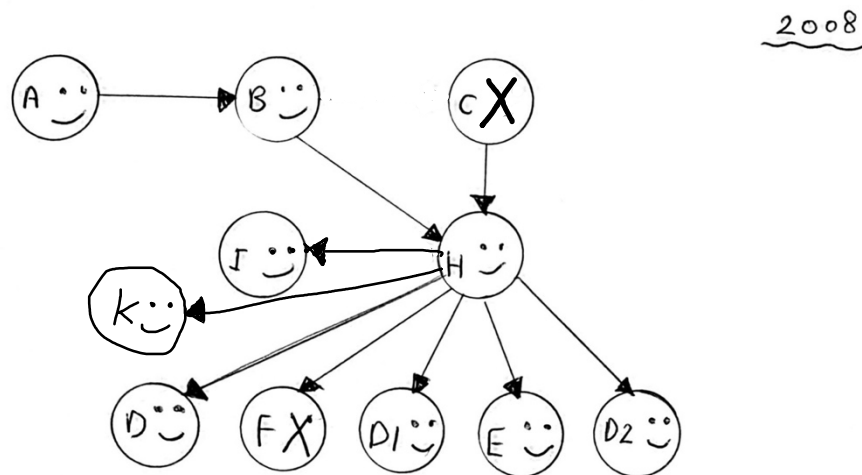


Figure 35: Case study 2 - Model for year 2008

Happy face: 9
 Sad face: 0
 Neutral: 0
 Don't know: 0
 Total: 9

Ratio

Happy: 1
 Sad: 0
 Neutral: 0
 Don't know: 0

2009:

- BT awarded additional contract to take over eight trusts formally with Fujitsu, plus 25 trusts for RIO and four additional acute trusts in Southern area.
- Other Southern trusts given choice of local service provider solution from BT or CSC or from various suppliers in Additional Supply Capability and Capacity List (ASCC)
- Martin Bellamy, director of programmes and systems delivery, NHS CFH, resigns
- NHS CFH, headed by Christine Connelly, is integrated with Department of Health Informatics Directorate
- Parliamentary announcement of contract negotiations with BT and CSC seeking NPfIT costs saving

Table 20: Identification of actors and their roles in 2009 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

Year	Actor id	Actor	Comment
2009	J	BT awarded additional contract to take over eight trusts formally with Fujitsu, plus 25 trusts for RIO and four additional acute trusts in Southern area. Other southern trusts given choice of local service provider solution from BT or CSC or from various suppliers in additional supply capability and capacity list (ASCC)	
	K1	Christine Connelly – NHS CFH is integrated with department of health informatics directorate	

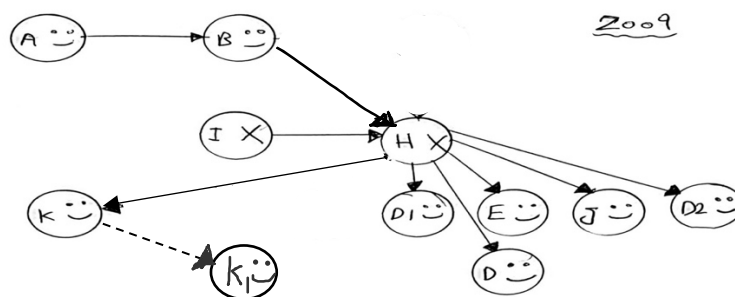


Figure 36: Case study 2 - Model for year 2009

Happy face: 9
 Sad face: 0
 Neutral: 0
 Don't know: 0
 Total: 9

Ratio

Happy: 1
 Sad: 0
 Neutral: 0
 Don't know: 0

2010:

- New memorandum of agreement signed between BT and NHS CFH, including reduced number of deployments in acute trusts in London
- Contract discussions with CSC continuing
- UK general election in May – new coalition government

Table 21: Identification of actors and their roles in 2010 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

2010	A1	New memorandum of agreement signed between BT and NHS CFH, including reduced number of deployments in acute trusts in London Contract discussions with CSC continuing UK general election in May – new coalition government	
------	----	---	--

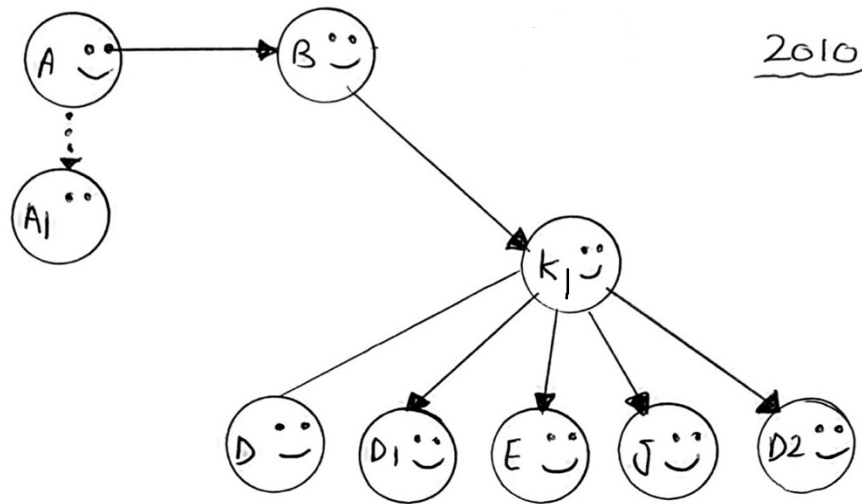


Figure 37: Case study 2 - Model for year 2010

Happy face: 8
Sad face: 0
Neutral: 0
Don't know: 1
Total: 9

Ratio

Happy: 0.89
Sad: 0
Neutral: 0
Don't know: 0.11

2011:

- The National Programme for IT has finally come to an end, although the bill for the enormously expensive and controversial project will continue to be paid for years to come.
- The deadline to exit NPfIT national contracts in the North, Midlands and East passed on 7 July, marking the end of the final chapter of the £12.7 billion attempt to bring the NHS into the digital age.
- Around £2.6 billion of actual benefits had been identified as of March 2011.

Table 22: Identification of actors and their roles in 2011 as specified in the National Program for IT (NPfIT) in the National Health Service (NHS) analysis

2011		<p>The National Programme for IT has finally come to an end, although the bill for the enormously expensive and controversial project will continue to be paid for years to come.</p> <p>The deadline to exit NPfIT national contracts in the North, Midlands and East passed on 7 July, marking the end of the final chapter of the £12.7 billion attempt to bring the NHS into the digital age.</p> <p>Around £2.6 billion of actual benefits had been identified as of March 2011.</p>	
------	--	---	--

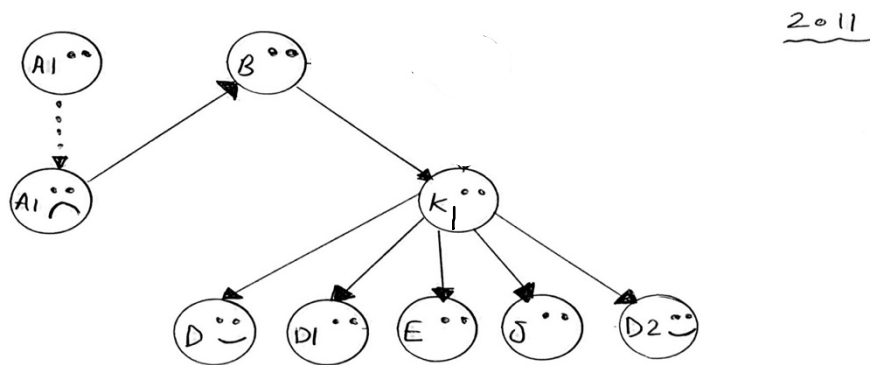


Figure 38: Case study 2 - Model for year 2011

Happy face: 2
 Sad face: 1
 Neutral: 0
 Don't know: 6
 Total: 9

Ratio
 Happy: 0.22
 Sad: 0.11
 Neutral: 0
 Don't know: 0.67

Summary of the second case study

As mentioned above, a National Program for IT (NPfIT) in the National Health Service (NHS) was launched in 2002 by the Department of Health.

Purpose of the new system was to link over 30,000 GPs to nearly 300 hospitals to share access to 50 million patients record in what was to be known in the future as the NHS Central Spine. In the same year and at the beginning of the project, an NHS IT director was appointed. In 2003, some well-known companies became involved with this project and had been awarded contracts for 10 years for 5 different clusters. BT was the most popular contractor throughout the project life.

In 2004, BT was awarded another contract (NHS broadband network) for the delivery of NPfIT. In 2005, NHS Connecting for Health (NHS CFH) was set up. In 2006, Accenture, one of the main contractors, withdrew: it is not clear from the source that I have used (Dolfing, 2019), since when this contractor became unhappy or whether, in the other words, they were they unhappy any earlier.

Another contractor, CSC, joined the project for the remaining of the contract term between NPfIT and Accenture for the “North East” and “Eastern England” clusters.

In 2007, a project reorganisation occurred, and the 5-clusters set up was replaced by a 3-clusters set up.

In 2008, another contractor, Fujitsu terminated from NPfIT, and again it is not obvious from this analysis when Fujitsu become upset: Fujitsu or customers might have not been happy if lawyers have got involved.

In 2009, BT took over those clusters, which were left by Fujitsu, plus some more. In the same year, the director of NHS CFH resigned: again, it is unknown if he was unhappy before. In the same year, NHS CFH became headed by and was integrated with the Department of Health Informatics Directorate.

5.4 Conclusive analysis over the two case studies: look for symptoms!

From the analysis of the above two case studies (both based upon clear failed projects), I have derived, and I am hereby sharing, as a guidance for practitioners, some symptoms to look for, by which to analyse and interpret the respective scenarios and stage components, and I would do so by simply looking at the models themselves.

- **1st symptom: check whether smiling faces decrease along projects' life cycle**

A first information (simple enough to capture by looking at the emoji models) is that they all show a decrease of smiling faces alongside the respective projects' life cycle.

In the first case study, Sauer (1993) would argue that losing a champion and support seems correlated to an increase in the number of sad faces and this can be interpreted as some bad signs and a negative forecast for the future of the project. I am less interested, at this stage, to actually perform a statistical analysis on the number of sad or happy faces: I would argue that a practitioner needs to rely on an “impression-based” quick decision-making process, and again this is exactly what would make PEN valuable in a real-world scenario.

- **2nd symptom: look for unaware smiling faces**

Sauer also points out that “*for the researcher, the problem of explaining why the Board decided to continue its support remains outstanding*”, which in our PEN-based models can be represented by either an unjustified presence of smiling face(s) or neutral face(s), “unaware” of the surrounding dynamics and circumstances: something that a model could capture if the practitioner is proactively looking for a similar pattern or heuristic.

- **3rd symptom: try to monitor everything by regularly pursuing a helicopter view**

A third potential heuristic would regard mismanaged support, as mentioned by Sauer when he claims that “*the strategies and practices for managing support outlined earlier in the paper do not address the problem encountered by the Mandata project organisation of the withholding of support such as staff resources*”: similar cases might happen when the requirements engineer and/or the organisation they serve do not have access to an “helicopter” view of the project and might miss important information about circumstances happening at the “periphery” and yet able to dramatically affect support and chances of success for that project. Again, potentially a “pattern” the practitioner might want to proactively look for within/across their models.

- **4th symptom: check whether the number of actors keeps increasing (unreasonably and unexpectedly)**

A fourth point: should an increase of actors as the project progressed be seen as a bad sign? If this happens unexpectedly or unreasonably, yes, it could be. If nothing else, because this might mean the initial stages of the project might have been grounded on a weak analysis of relevant stakeholders. But it might also mean that the project is evolving into uncharted territory, and more actors get involved with their own agendas or simply to fix problems ignored at previous stages.

Even if PEN could not have solved any problem, it would have raised awareness without any need to get involve with politics. PEN, let me reiterate, would need to be kept strictly confidential and should only be used by the requirements engineer and their team.

PEN could have offered this by highlighting, capturing and understanding most of the political aspects of the development process, and I would argue that it could even led to the earlier cancellation of project, which in turn could have resulted in saving time and money.

As for the involvement of the requirements engineer(s) in the Mandata project, it could have helped if they were able to find the key person/influencer to turn to, if needed. From the early stage, PEN could have also helped with recognition of stakeholders’ emotion and facial expressions, thanks to its easy emoji-based representation. This could have provided warning signs and would have eventually facilitated the understanding of why those sad faces were there, before it was too late.

Furthermore, PEN could have been useful when someone new was joining the project mid-cycle. This would have been particularly true for both the first and the second case study, in which people and companies were keep coming and going (see the 4th symptom).

All the advantages of using PEN, as mentioned for the first case study (5.2), seem to apply quite smoothly also to the second case study (5.3). In the life cycle of the National Program for IT of the NHS, it appears evident that PEN could have indeed been beneficial for generating some appropriate questions against the 4 symptoms above, with regards to the roles and relationships among all stakeholders, who were either directly presented in the project or indirectly influencing the several decision-making processes underlying the Program.

Also, in the second case study, the political aspects of the development process could have been better highlighted, captured and understood by using PEN-based models, without the need of any involvement in the underlying politics. PEN could have raised awareness and, even if it would have solved no problem, it could have led to early cancellation of the National Program for IT (NPfIT) project, which instead lasted over 10 years.

Chapter 6: Results and Findings (overall, how well the problems were addressed?)

6.1 A background recap to better focus on the aims of my research

In order to fully appreciate results and findings of my PhD research, I think it is useful for me to provide first a quick recap of how my research aims evolved since it started several years ago.

The initial focus of my research was on the current state of the art in requirements engineering (RE) practice: specifically, at the very beginning, the main aim was to investigate how and why requirements engineers' role has been changing over the years, in the hope of helping professionals and academics to make sense of the present state of affairs and to face (if not to drive) any future evolution of the job (and the underlying discipline) by relying on some sounder conceptual and interpretative tools.

One important component, functional to the achievement of that first aim, was about tracing the history of requirements engineering and its progress over the years and investigate whether any technological progress in the requirements engineering process (for example, the appearance on the market of always new and innovative tools and techniques aiming at supporting and improving the requirements engineer's job), was actually and effectively resulting in products of higher quality, resulting in more successful large-scale software development projects.

It became apparent, from the literature survey and the following analysis I performed, that the correlation between availability (and adoption) of innovative technologies and the likelihood to complete successful software projects was far from being clearly ascertained, as it looked the main causes of projects failure were -in last instance- of non-technical nature.

And whilst most of the practitioners and academics were committed to demonstrate and affirm their professional identity of software analysts and developers as “engineers” (from the well-known and previously cited Proceedings of the two NATO Software Engineering Conferences in 1968 and 1969 onwards), and to reiterate and propagate such a belief along decades of teaching in further and higher education (Sommerville, with his handbooks, is an example of musketeer supporting such a belief), a macroscopic confirmation bias was being established, based on the assumption that all main problems affecting software projects were in last instance solvable by investing more in technical facilities and in better “engineered” solutions.

It was a sort of awakening for me to realise and accept an almost self-evident fact, well reported in literature, at least from the Standish Group report (1995) onwards: non-technical problems (and, among them, as I noticed later, political problems) were not solvable only by technology, or we might not have had so many failed projects.

That's when, focusing on the non-technical nature of causes leading to failure of so many large-scale software projects, I noticed what I started referring to as the “elephant in the room” (Siadati et al., 2019a): in RE, *organisational politics* was playing, unacknowledged, a crucial role. That's also when a significant shift occurred in my PhD research, as I started investigating

how to capture and model this new conceptual key, “politics”, so that it could be controlled and used by practitioners.

Eventually, four new research questions had been eventually raised, as reported in previous section 3.7 of the present dissertation and are hereby revisited and discussed in the light of the impact evaluation-based analysis on organisational power and politics, which I could perform through my suggested modelling notation, PEN.

6.2 Revisiting research questions

This section shows how the practical steps taken in this research could address the research questions which I raised in Section 3.7, one by one, in the next four sub-sections.

- **6.2.1 Revisit - Why and how to focus on RE: politics as the conceptual key**

This was the first new question I identified, after an extensive background research on software, software engineering and requirements engineering allowed me to detect the elephant in the room: politics in RE.

In my thesis, I have addressed the importance of the role of the software requirements engineer in the development process, and I have agreed with many authors (as mentioned in the previous Chapters 2 and 3) that we might expect more successful projects by:

- getting the early phase of requirements engineering right,
- taking advantage of any insight we could gain on politics within organisations (for example, who the decision maker is when it comes to choose between priorities),
- identifying all the stakeholders (even the hidden ones), their organisation and their needs, and to map them against their goals.

In section 3.7, the crucial role of non-technical factors alongside the technical ones has been discussed and politics -as one of the non-technical components- has since gained the status of the key conceptual investigation object of this research. Although politics and power relationships among stakeholders have been indeed already identified and discussed by previous academics and practitioners in software requirements engineering, I argued how other non-technical factors have been addressed better than politics (e.g., finance or project management), as these -in many cases- became complementary part of the curriculum, upon which practitioners still get trained and/or educated in first instance.

An ability to navigate through political relationships among stakeholders should be included in the list of expected skills of requirements engineers, if nothing else for negotiation purposes, as failing to comply with “diplomatic” tasks would increase the risk of project failure. The requirements engineer should be able to identify who the main decision makers are within a certain organisation, what leverages and arguments to exploit in order to steer support among stakeholders towards agreed objectives, how to empower champions and let them drive projects in a proactive and positive manner: all of the above, by keeping in mind that organisational charts often fail to show the real decision maker(s) behind the official organisational structure and, either by design or unintentionally, tend to hide or confuse political dynamics and their source(s). Mapping organisational politics, the political dimension of finance and the politics of the technological dimension are the three non-technical tasks that have been considered in

section 3.7 and that should be delivered to practitioners and learners to mitigate and counterbalance what they have been taught for so many years, by the like of Sommerville (2007), who believe that *“engineers [just] make things work. They apply theories, methods and tools where these are appropriate”*.

- **6.2.2 Revisit - Where is RE from? Politics and power underlying human activities**

In revisiting the second research question, ‘Where is RE from? Politics and power underlying human activities’, it is necessary to remind ourselves that several authors, such as Nievergelt (1968) and Boehm (2006), tried to bring to our attention that in software requirements engineering either we learn from our mistakes, or we are to repeat the same mistakes again.

Whilst some “political” roots of RE were revealed and discussed in the background chapter (2), requirements engineers, both in industry and academia, have systematically denied them the degree of priority these would have deserved and demanded in their daily practice. The idea that a technological solution, no matter how sophisticated and effective, might mitigate the impact of egos and agendas based on (or pursuing) power relationships, is naïve and -in last instance- might prevent the achievement to the achievement of the very same goals any IT project would exist for, as goals (and requirements) are identified and decided -in first instance- as the by-product of an intrinsically political decisional game.

To further address this research question, I suggested in chapter 2 (and would suggest as future work) the need for the recognition and addressing of power and politics as a crucial part of a redefinition process of RE itself: system requirements are identified, analysed and prioritised according to the political power of each stakeholder. My modelling notation, PEN, aims at highlighting what the power for each identified stakeholder might be.

- **6.2.3 Revisit - What is RE really about? Politics and power in action: the subtle art of negotiation**

The third research question concerns requirements engineering itself as a discipline. To address this question, as it has been explained in section 3.7.3 (with reference to several quotes from practitioners and academics), requirements engineering could be safely referred to as a solution to the problem(s) we experience with software systems.

The ‘engineering’ component of the discipline (and all the technical activities it is compounded of) is not exhaustive for fully understanding the extent of challengingness practitioners need to be ready to embrace. Some authors, like Brooks (1995), believe that *“the hardest single part of building a software system is deciding precisely what to build”*. Depending on how we interpret that statement, we might be inclined to define RE either in terms of its major, official tasks alongside a development process (such as requirements elicitation, analysis and specification), and therefore by highlighting the technical component, or also in terms of communication and negotiation, which is exactly where the non-technical component would play a crucial role.

Recognition of who the stakeholders are and having them all engaged in the process should be one of the most productive activities that requirements engineer can do. This would clash against the idea of having one single user/stakeholder involved with all the knowledge that requirements engineer(s) might need, as sometimes is hoped within agile development environments, with their assumption that development would work better when supported by an on-site user. Extreme Programming (XP), for example, explicitly refers to a single on-site customer: this goes against one of the most commonly accepted basic assumption in RE. As

many practitioners, like Alexander and Baudouin (2009), and I am with them, would claim, the idea “to go and get the requirements from the users directly” is very naïve if not plainly misleading, and this because:

- *There isn't one uniform group of people who use a product.*
- *You don't know what using the product will be like until you've designed it*
- *The users don't know what they want until you show it to them.' Products create requirements, as much as the other way round.*
- *You often can't talk to users directly, but you must deal with surrogates including yourself and your project colleagues.'*

So, no: there is no shortcut when it comes to RE. As requirements engineers, we have to deal with different users and stakeholders, with different viewpoints, possibly with conflicting goals, with people who do not necessarily know what they want or what is possible, and with stakeholders who could be external and internal to the organisation. And no, users are not the only type of stakeholders, and not all stakeholders come with the same “power”: some might have relatively little power (e.g., staff), others huge power (e.g., government or trade unions).

Also, some stakeholders are more obvious than others, and some are just hidden, which doesn't help when it is time to identify all the stakeholders and their needs and priorities. And when the stakeholders are well-known, their expectations might still not be clear, sometimes because they might not know what they exactly want, sometimes because they know they cannot anticipate all problems they might be facing and need support for.

For all the above reasons, more attention should be spent on stakeholders (and their requirements) rather than on products (and their requirements), and practitioners should always assume (and be prepared) for uncertainties and mistakes. More than that: they should assume users might even sign-off requirements specifications without fully understanding them. Otherwise, as Easterbrook (2004) pointed out, requirements can cause problems rather than solving them.

We mentioned already how important for requirements engineers might be to know more about each stakeholder for the purpose of negotiation, in order to ensure they are able to find the right person to negotiate in case of conflicting requirements: this means practitioners need to get familiar with their client's organisation. This is why we proposed PEN in first instance: a tool to allow such a knowledge to be captured, supported and exploited by the requirements engineers.

But this is also to reiterate where the main goal of this research fits: requirements ultimately begin and end with people – stakeholders and looking for political relationships could simply prove to be a special strategy to better capture requirements and support the following negotiation process.

- **6.2.4 Revisit - Who does the RE job? The politically vested analyst and engineer**

In this section, it would be appropriate to concentrate on the requirements engineers themselves. In the early days of RE, the software requirements engineer could have been a person working alone or in a small team, in close contact to the main stakeholder. The practitioner then needed to fully understand the scope of the job/problem from the discussion they were having and come up with an appropriate solution accordingly. The overall situation could have become

problematic as soon as individual limitations were to be considered: the stakeholder might not know what exactly they want or what options are available to them. Requirements engineers also should have known everything (including all the technical methods, processes and tools) to produce a practical solution, which was (and even more now is) impossible.

Nowadays we know, as shown in previous chapters, that there won't be large-scale projects with a single main stakeholder: instead, many diverse stakeholders with diverse viewpoints are to be taken into account. Having all this in mind, with all the other previous discussion, even if we assume that requirements engineer masters enough of the technical knowledge and understanding, would there be sufficient understanding of negotiation techniques? And, if they don't have it, how are they supposed to gain these techniques? At the moment, answer seems to be from experience. Which is true, of course, and yet, in its vagueness, it denounces our inability to qualify and better define the actual "content", the "what" it would be that makes practitioners experienced: the only bit there seems to be an agreement on is that whatever it is, it is not part of the "technical" component of the job.

Supported by literature and availability of analysis on two major failed case-studies, my finding is that what makes experience so valuable is the ability gained by successful practitioners to manage the political dimension of any project, by learning about stakeholders, their organisation(s) and the power and politics in those organisations.

The above would already be enough to enter into an argument with the two major professional bodies (at least in Western Europe) that currently release a recognised status to requirements engineers, and the training handbooks they adopt: the German IREB (International Requirements Engineering Board) and the British Computer Society, here in the UK, given they do not consider organisational politics as a crucial part of the train requirements engineers should undertake for their mission.

Requirements engineers trained on those handbooks (only) will have barely heard of politics and power to be considered alongside their practice and might simply be unprepared for, and possibly unable to cope with, any associated challenge.

The well-prepared requirements engineer should be equipped to identify power and politics in and organisation, and to map and model it.

6.3 Modelling and Applying the notation

There are also lots of different notations available for modelling stakeholders' and system requirements, but none of them addresses and takes into account the political dimension of any requirements-related exercise whilst keeping in mind the real working scenarios practitioners might need to operate: informal, fast, confidential, potentially conflictual.

My Political Emoji Notation (PEN) has been designed having exactly the above scenario in mind: it is informal, fast and easy to use, prone to confidentiality and able to immediately detect and somehow understand potential conflicts. All this, by simply capturing and modelling relationship and emotions and/or attitudes towards a certain project among all the obvious and non-so obvious stakeholders.

If identifying all the stakeholders, who are linked to the project internally and externally, their roles and relationships, is very important to ensure we are enabled to collect and process as

many requirements as possible, then the identification of power and politics across the project would become crucial, if nothing else for validation purposes.

Modelling the actual power relationships in an organisation, against those identified from a traditional organogram, would help the requirement engineer to identify not just influencers, but also potential conflicts and inconsistencies.

It should be highlighted that political influence not necessarily comes from the person(s) above: my easy-to-learn and easy-to-apply emoji-based notation has been developed to highlight the details of all influencing relationships within a project and allow practitioners to add, to any existing requirements model and specification, an extra (confidential and informal) layer of “political” information. The notation should recognise and document power, politics and emotional aspects of software requirements: this will hopefully help to understand the customer organisation and assist requirements engineers as a result to identify whose problem are most urgent to solve.

The proposed notation comes with the advantage of showing power and direction of power: this way it supports the model to be easily adapted to better mirror what political changes occur when stakeholders change their views/attitudes towards the whole project, or parts (i.e. requirements) of it.

A full list of advantages in using PEN (together with its details) is available in section 4.1. Once the PEN notation has been introduced to support the graphical modelling of politics in requirements engineering, in Chapter 5, I could evaluate its impact on the analysis of two major case studies, reporting on failed projects.

Impact evaluation, as discussed in Chapters 3 and 4, is the prime academic methodology adopted for this research to validate efficacy and value of PEN.

We focused on two well-known and analysed failed projects, so to assess (retrospectively) how PEN could have been of value to the practitioners during the project. It is interesting to note both projects were from the public sector (one in Australia, the other -more recent- in the UK): failed projects in the private sector tend not to attract public scrutiny, and therefore are more difficult to analyse retrospectively, but there is room for arguing PEN would be as useful also when adopted during requirements-related activities in private projects.

The first case study is the Mandata project, as it has been analysed nicely by Chris Sauer. According to Sauer (1993), *“the system, called Mandata, was intended to automate the administration of personnel and establishments processes for the whole of the Australian Public Service”*.

The National Program for IT (NPfIT) in the National Health Service (NHS) by the Department of Health has been chosen as the second case study. Similar to the first case study, several authors could provide an accurate analysis for this second case study: I decided to base my impact evaluation exercise on Dolfing’s (2019).

Several models for each case study have been drawn and analysed, based on the two main available documents that have been chosen for this purpose. Emoji-based representation of emotions and/or attitudes from stakeholders, and other features captured through the PEN models (which should lead to the detection of the “four symptoms”, as addressed in previous

section 5.4), could provide not just a simplified narrative but also some elements or nodes of the political story of each project, which I have argued should enable (or, for those case studies, should have enabled) practitioners to detect signs on how the project is or was evolving. Identifying those signs (or symptoms) means for the requirements engineer to identify which stakeholders to help in order either to turn a negative trend within the project life cycle into a positive one or, if nothing else, to facilitate the end of the project and -by doing so- to avoid waste of finance, time and resources.

This is consistent also with guidance offered by Alexander and Beus-Dukic (2009) about negative stakeholder:

“If you know any negative or hostile stakeholder groups, you have an especially urgent task to make a peace with them, or at least to defuse their negative feelings. Find out what they are concerned about; make sure they are properly informed and set right any misconceptions they may have; [...] If at all possible, take time to negotiate an agreed way forward with them”.

PEN could help us to find stakeholders with “negative feelings”, but also stakeholders who are irrationally happy (maybe because unaware?) and stakeholders whose attitude and power can determine a change in the project.

Retrospective application of PEN-based modelling might be useful for better understanding possible causes of failures for past projects. However, I would argue PEN would be more beneficial when used on live projects: when a requirements engineer is involved in a live project, capturing emotions and attitudes is much easier than relying on documents left behind. One single example is that in the case study 2, if we consider any particular company, we do not really know how long they were unhappy for alongside the whole life cycle of that project.

It is important for me to highlight that, even if PEN might not solve all problems faced by the requirements engineers, at least it would raise awareness of the political components of any project and eventually, by engaging more proactively and strategically with key stakeholders and decision-makers, it could lead to better address some recurrent and well-known issues in requirements engineering.

Chapter 7: Conclusions and Future work

7.1 Overall Conclusions

The main aim of this research, after finding a big gap in the field of requirements engineering, was to identifying elements of a workable notation for documenting political and power relationships within a typical RE project, and through which the necessary negotiation processes might be contextualised and understood.

Once the first version of this notation was ready, the Impact Evaluation methodology has adopted, based on the retrospective counterfactual analysis of what difference an intervention would have made in outcomes: this methodology has been mostly developed within the policy-making sector and became a mainstream methodology after the World Bank adopted it extensively (Gertler et al., 2016). The best-case scenario would be to invite experienced practitioners to use this approach to analyse their previous projects and consider whether this would have helped them in their work, particularly in identifying and resolving political and power-related issues which they had to address; this is shown under future work below.

Sauer tested his own framework, known as Sauer's exchange framework (Sauer 1993), on the Mandata case study and this is how he explains it: *"This will do three things. One, it will allow us to observe its applicability - how useful is it, what does it help explain, what does it not explain? Two, it will raise some interesting research problems. And three, it will suggest some practical morals for project managers"*. I adopted a similar approach for testing the PEN modelling technique, on the basis of Sauer's analysis of the Mandata project (the first case study) and of Dolfing's (2019) analysis of another failed project, the National Program for IT (NPfIT) in the National Health Service (the second case-study), and for deriving some guidance for practitioners, especially as looking for "symptoms" of problems is concerned (see Chapter 5.4).

Along the dissertation -and especially in Chapters 2 and 3- it has been argued that taking into account political relationships, as they occur between stakeholders and within complex organisations, can be of crucial importance for ensuring the success on any development project.

As mentioned in Siadati et al., (2017), *"at the moment, practitioners in RE can exercise their professional expertise by being aware of the political dimension, or by simply assuming the engineering process is politically neutral"*. But should they decide to adopt a more proactive approach to dealing with politics in RE, no simple notational technique is available to them.

Therefore, a simple conceptual tool has been proposed, together with an even simpler technique to capture, at least partially, important features of any political relationship within organisations, so to enrich the project with (confidential) politically informed modelling outcomes. This could be of value for the individual practitioner as well as for whomever is due to substitute and/or replace the previous one, not just as a mean for identifying and describing potential sources of problems and conflicts, but also for communication purposes (whilst paying attention to the confidential aspect of this).

7.2 Future work

Whilst the thesis has represented (and demanded) significant amount of effort in terms of literature review, conceptualisation, creativity and analysis, a number of tasks have been identified for future research, mostly to extend and deepen the work so far completed, but also to test and better adapt it for real-world use by requirements engineers and other practitioners.

Once politics is reinstated within the core of RE activities, further investigation in this area would be valuable and I would argue its outcomes could produce simple and yet effective tools and practices, which practitioners can start pursuing in their daily activities.

Future work would include the following:

- Use Ratio analysis to test its usefulness in a real world project.
- Identifying a number of volunteers (experienced practitioners in both industry and academia) to use my approach and PEN to analyse their previous projects and consider whether this would have helped them in their work, particularly in identifying and resolving political and power-related issues which they had to address. This has not been possible during the present research for many reasons: some kind of “sponsorship” from either professional institutions or industry providers of CASE tools (like Doors®) would definitely be of help.
- Revisit other famous case studies, such as the famous London Ambulance Service (or LAS) system history, identify and add other stakeholders (for example, for LAS, that would be ambulance drivers and trade unions), and see whether PEN and the four symptoms I have introduced could provide further insightful understanding of the causes leading to the failure. Also, it would be interesting to see whether, in such an impact evaluation retrospective exercise, PEN models would provide practitioners and analysts with hints towards a potential second implementation and advise on how the new project could work better.
- Extending or adding new case studies to explore
 - The informal relationship in national (public sector) VS corporate (private sector) culture, and how usage of PEN could adapt to different “domains”
 - The role and effect of an organisational champion, and how this, in synergy with PEN, would provide extra “political” support
 - considering requirements engineers as a stakeholder themselves, and how this would change the overall politics and power relationships within a certain project
- Applying the Political Emoji Notation (PEN) to case studies derived from Safety Critical Systems (SCSs), as in a Safety Critical System project a broader range of political stakeholders is usually deeply involved, and adoption of PEN might prove to be more beneficial than in other domains.
- Using PEN in other fields in which politics are an important aspect, as at the moment my work has only been applied in the field of software requirements engineering.

- Whether an integration between PEN and i* might prove to be beneficial for both modelling techniques, to provide i* a higher usability and PEN with sounder formal foundations.
- Apply PEN to the other, this time successful projects, to explain successes as well as failures.
- Apply PEN to live projects: this part of the research could prove to be ethically arguable (unless the practitioners are the researchers themselves!) and challenging to manage, given the obvious constraints in terms of confidentiality and neutrality.
- As this work has highlighted the need for the recognition and addressing of power and politics as a necessity, it would be appropriate to explore some further redefinition of RE itself and eventually to provide suggestions for changes to the syllabuses of the professional bodies, as identified and discussed in previous parts of this dissertation.

7.3 Contribution to knowledge

This PhD contributes towards the following knowledge-based solution by doing extensive and innovative research in the field to solve existing problem in the field of Software requirements engineering.

1) Creation of Political Emoji Notation (PEN) and Metrics based on ratio

The proposed political Emoji Notation (PEN) model enables requirements engineers and their team to model power and politics. Sad (unhappy) stakeholders could be risky if not dangerous to the success of the project but, when they are identified, can be addressed by finding the reason for them not being happy. PEN uses the well-known Emoji faces that can be happy, sad or neutral, to represent users' attitude towards the project, connected by lines to show their dependencies/influence and arrowheads to show the direction of power, that is who has influence over whom. The number of lines shows the level of influence. Emojis representing stakeholders with no connection to others exist but are not involved in the process. PEN aims at showing "political" relationships among stakeholders, and how these affect each stakeholder's emotional response and attitude towards the project. The main drive behind PEN is simplicity, so to encourage practitioners to use it, and yet it comes with an important disclaimer: the intention of PEN-based modelling would always be to model power relationships within or across organisations during any system development project, but by no means of getting practitioners involved in politics.

2) Demonstrate that it is capable of working on the real world through case study.

PEN has been tested on two failed software projects (Mandata and the National Program for IT (NPfIT) in the National Health Service (NHS) by the Department of Health) to proof its capability. More details could be found in section 5.

3) **Derive some symptoms to identify when problems might be arising in a project due to political considerations.**

Four symptoms as a guidance for practitioners, by which to analyse and interpret the state of power and politics in software projects have been identified. More details could be found in section 5.4. above

- **1st symptom: check whether smiling faces decrease along projects' life cycle**

Noticing a decrease in smiling faces as projects progress, may suggest a negative forecast for the project outlook.

- **2nd symptom: look for unaware smiling faces**

PEN-based models can also be represented by either an unjustified presence of smiling face(s) or neutral face(s), “unaware” of the surrounding dynamics and circumstances

- **3rd symptom: try to monitor everything by regularly pursuing a helicopter view**

A third potential heuristic would regard mismanaged. Potentially a “pattern” the practitioner might want to proactively look for within/across their models.

- **4th symptom: check whether the number of actors keeps increasing (unreasonably and unexpectedly)**

This might mean the initial stages of the project might have been grounded on a weak analysis of relevant stakeholders. But it might also mean that the project is evolving into uncharted territory, and more actors get involved with their own agendas or simply to fix problems ignored at previous stages.

7.4 Closing thought

To conclude this research, it is worth to remind that my Political Emoji Notation (PEN) is not intended to solve the technical problems of software development (that is where most of the efforts in terms of innovation have been spent in the last three decades anyway), but it can help requirements engineers to understand political problems and issues that cannot be solved only with technology.

At the same time, it is important for me to underline that PEN is a modelling technique to understand situations, not to get actively involved with the politics of customer organisations. From the early stages of any project, finding the source of power and politics, confidentially documenting it, and drawing it up in PEN alongside the whole life cycle of requirements in a project could lead to some understanding, to the identification of leverages and hopefully some symptoms, which could better help the practitioner to act strategically and support in a more efficient manner any decision-maker.

In last instance, one crucial achievement of this thesis (and the research behind it) is the disclosure and acknowledgement of an elephant in the room, as politics is there in any RE

exercise: the more practitioners and academics are aware of the elephant, the abler they can become to mitigate the damages it could (and very likely would) lead to, if left un-monitored. There is still a steep mountain to climb but at least an important mountain has been identified, and what has made it so challenging to climb in the past. Hopefully, once the peak is reached, the view will come with an important contribution worthy to be considered and fully adopted by the whole RE community.

References and Appendices

3ie, 2016. Available at: <https://www.3ieimpact.org/> [Accessed 17th Dec 2016]

Alfor, M.W. and Lawson, J.T., 1979. *Software Requirements Engineering Methodology (Development)*. TRW DEFENSE AND SPACE SYSTEMS GROUP HUNTSVILLE AL.

Aguirregabiria V. 2012. A method for implementing counterfactual experiments in models with multiple equilibria. *Economics letters* 114(2012) 190-194

Alexander I. F. 1997. A Historical Perspective on Requirements from Requironautics Quarterly, the Newsletter of the BCS RESG.

Alexander, I., 2003. Stakeholders: who is your system for? *Computing and Control Engineering*, 14(2), pp.22-26.

Alexander, I.F. and Maiden, N. eds., 2005. *Scenarios, stories, use cases: through the systems development life cycle*. John Wiley & Sons.

Alexander I. F. and Beus-Dukic L. 2009. *Discovering requirements: how to specify products and services*. John Wiley & Sons.

Andriole, S., 1998. The politics of requirements management. *IEEE software*, 15(6), pp.82-84.

Armitage, A. and Keeble-Allen, D., 2008, June. Undertaking a structured literature review or structuring a literature review: tales from the field. In *Proceedings of the 7th European Conference on Research Methodology for Business and Management Studies: ECRM2008*, Regent's College, London (p. 35).

Atkinson, R., 1999. Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International journal of project management*, 17(6), pp.337-342.

Azuma J. and Ebner M. 2008. A stylistic Analysis of Graphic Emoticons: Can they be Candidates for a Universal Visual Language of the Future? In *Proceeding of World Conference on Educational Media, Hypermedia and Telecommunications (ED-Media)*, 2008, p. 972-977.

Bano, M., Zowghi, D. and da Rimini, F., 2018, June. Power and politics of user involvement in software development. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018* (pp. 157-162).

Baresi, L. and Heckel, R., 2002, October. Tutorial introduction to graph transformation: A software engineering perspective. In *International Conference on Graph Transformation* (pp. 402-429). Berlin, Heidelberg: Springer Berlin Heidelberg.

Barlas, S., 1996. Anatomy of a Runaway: What Grounded the AAS. *IEEE Software*, 13(1), p.104-106.

Bergman, M., King, J.L. and Lyytinen, K., 2002. Large-scale requirements analysis revisited: the need for understanding the political ecology of requirements engineering. *Requirements Engineering*, 7(3), pp.152-171.

Berry D. M. and Lawrence B. 1998. Requirements Engineering, *IEEE Software*, Vol. 15, No. 2, pp. 26-29

Boehm, B.W., 1975. Software design and structuring. In *Practical strategies for developing large software systems*. Addison-Wesley.

Boehm Barry W. (1976). Software Engineering" in *IEEE Transactions on Computers*, vol. 25, no. 12, pp. 1226-1241, 1976.

Boehm, B.W., 1981. An experiment in small-scale application software engineering. *IEEE Transactions on Software Engineering*, (5), pp.482-493.

Boehm, B., 2006, May. A view of 20th and 21st century software engineering. In *Proceedings of the 28th international conference on Software engineering* (pp. 12-29).

Boehm, B.W., 2007. Software engineering economics. *Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research*, 69, p.117.

Brandon, R. and Seldman, M. 2004. Survival of the savvy: High-integrity political tactics for career and company success. Free Press, New York.

Brennan, S., 2009. The National Programme for IT (NPfIT): is there a better way?. *Integrating Healthcare with Information and Communications Technology*, p.95.

Brooks Jr, F.P., 1995. *The mythical man-month: essays on software engineering*. Pearson Education.

Bubenko J. A. jr. 1995. Challenges in Requirements Engineering, *Proc. 2nd IEEE Int. Symposium on Requirements Engineering*, York, England, March 1995, pp. 160-162.

Cameron L. 2018, The 2018 International Conference on Software Engineering Breaks Attendance Record as it celebrates its 40th anniversary.

Available at: <https://www.computer.org/publications/tech-news/events/the-2018-international-conference-on-software-engineering-breaks-attendance-records-as-it-celebrates-its-40th-anniversary> [Accessed 12th Nov 2019]

Chen A. and Beatty J. 2012, Visual models for software requirements. Pearson Education.

Cheng, B.H. and Atlee, J.M., 2007, May. Research directions in requirements engineering. In *Future of Software Engineering (FOSE'07)* (pp. 285-303). IEEE.

Cockburn, A., 2008. Good Old Advice.

Collins, T. and Bicknell, D., 1997. *Crash: ten easy ways to avoid a computer disaster*. Simon & Schuster.

CRD, 2009. Systematic Reviews. CRD's guidance for undertaking reviews in health care. Centre for Reviews and Dissemination, University of York, 2009.

Currie, W., Galliers, B. and Galliers, R. eds., 1999. *Rethinking management information systems: an interdisciplinary perspective*. Oxford university press.

Dahl, R.A., 1957. The concept of power. *Behavioural science*, 2(3), pp.201-215.

Dalpia F., Franch X., and Horkoff J. 2016. i* (i Star) 2.0 Language Guide, (accessed on 4th April 2021)

Davis, A.M., 1996. Practitioner, Heal Thyself. *IEEE Software*, 13(3), p.4.

Davis, A.M. and Hickey, A.M., 2002. Requirements researchers: Do we practice what we preach?. *Requirements Engineering*, 7(2), pp.107-111.

Dolfing H. 2019. Case Study1: The £10 Billion IT Disaster at the NHS.

Available at: <https://www.henricodolfing.com/2019/01/case-study-10-billion-it-disaster.html> [Accessed 20th May 2020]

Donaldson, A.J.M. and Jenkins, J.O., 2000. Systems failures: An approach to understand what can go wrong. *European software day of EuroMicro '00, Sep*.

Easterbrook S. 2004. What is Requirements Engineering?

Available at: <https://www.cs.toronto.edu/~sme/papers/2004/FoRE-chapter01-v7.pdf> [Accessed 5th July 2015]

Edmonds, E.A., 1974. A process for the development of software for non-technical users as an adaptive system. *General Systems*, 19, pp.215-218.

Finkelstein, A., 1994, December. Requirements Engineering: a review and research agenda. In *Proceedings of 1st Asia-Pacific Software Engineering Conference* (pp. 10-19). IEEE.

Fitzgerald, G. and Russo, N.L., 2005. The turnaround of the London ambulance service computer-aided despatch system (LASCAD). *European Journal of Information Systems*, 14(3), pp.244-257.

Fricker, S.A., Grau, R. and Zwingli, A., 2015. Requirements engineering: best practice. In *Requirements Engineering for Digital Health* (pp. 25-46). Springer, Cham.

Gallivan, M.J., Truex III, D.P. and Kvasny, L., 2004. Changing patterns in IT skill sets 1988-2003: a content analysis of classified advertising. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, 35(3), pp.64-87.

GAO 1979. Contracting for computer software development – serious problems require management attention to avoid wasting additional millions. US Comptroller General and United States of America.

Ghezzi, C., Jazayeri, M. and Mandrioli, D., 1991. *Fundamentals of software engineering*. Prentice-Hall, Inc.

Gebauer, J., Tang, Y. and Baimai, C., 2008. User requirements of mobile technology: results from a content analysis of user reviews. *Information Systems and e-Business Management*, 6(4), pp.361-384.

Geethalakshmi, S.N. and Shanmugam A. 2008. Success and Failure of Software Development: Practitioners' Perspective. *Proceedings of the International Multi Conference of Engineers and Computer Scientists 2008, Vol I IMECS 2008, 19-21 March 2008, Hong Kong*.

Gertler, P.J., Martinez, S., Premand, P. Rawlings, L.B. and Vermeersch, C.M., 2016. *Impact evaluation in practice*. The World Bank.

Gibbs, W.W., 1994. Software's chronic crisis. *Scientific American*, 271(3), pp.86-95.

Goodman, N., 1947. The problem of counterfactual conditionals. *The Journal of Philosophy*, 44(5), pp.113-128.

Gugerty M. K. and Karlan D. 2018. Ten reasons not to measure Impact – and what to do instead. Available at:

https://ssir.org/articles/entry/ten_reasons_not_to_measure_impact_and_what_to_do_instead

[Accessed 3rd April 2020]

Haigh, T., 2010, June. Crisis, What Crisis? Reconsidering the Software Crisis of the 1960s and the Origins of Software Engineering. In *Proc. 2nd Inventing Europe/Tensions of Europe Conf.* (p. 2).

Hochwarter, W. A., Witt, L. A., & Kacmar, K. M. (2000). Perceptions of organizational politics as a moderator of the relationship between conscientiousness and job performance. *Journal of Applied Psychology*, 85, 472–478.

Hofmann, H.F. and Lehner, F., 2001. Requirements engineering as a success factor in software projects. *IEEE software*, 18(4), pp.58-66.

Hoh I., David O., Rodgers T., 2001. A requirements negotiation model based on multi-criteria analysis. In: *Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE)*, IEEE Press, 2001, pp. 312-313.

Hull E., Jackson K. and Dick J. 2002. *Requirements Engineering*. Springer, Cham.

Holmström, J. and Sawyer, S., 2011. Requirements engineering blinders: exploring information systems developers' black-boxing of the emergent character of requirements. *European Journal of Information Systems*, 20(1), pp.34-47.

IEG, 2005. *OED and Impact Evaluation: A Discussion Note*. Operations Evaluation Department, World Bank, Washington, D.C.

- Jarke, M. and Pohl, K., 1994. Requirements engineering in 2001:(virtually) managing a changing reality. *Software Engineering Journal*, 9(6), pp.257-266.
- Jirauch, D.H., 1967. Software design techniques for automatic checkout. *IEEE Transactions on Aerospace and Electronic Systems*, (6), pp.934-940.
- Johnson, L., 1998. A view from the 1960s: How the software industry began. *IEEE Annals of the History of Computing*, 20(1), pp.36-42.
- Justinia, T., 2017. The UK's National Programme for IT: Why was it dismantled? *Health services management research*, 30(1), pp.2-9.
- Kaindl, H., Brinkkemper, S., Bubenko Jr, J.A., Farbey, B., Greenspan, S.J., Heitmeyer, C.L., do Prado Leite, J.C.S., Mead, N.R., Mylopoulos, J. and Siddiqi, J., 2002. Requirements engineering and technology transfer: obstacles, incentives and improvement agenda. *Requirements Engineering*, 7(3), pp.113-123.
- Kitchenham, B., Linkman, S. and Law, D., 1996. DESMET: A method for evaluating software engineering methods and tools. *Keele University*.
- Kövecses, Z., 2000. "Metaphor and Emotion: Language, Culture, and Body in Human Feeling". Cambridge Univ Press, Cambridge, UK, 2000.
- Kuhn T. S. 1970. *The Structure of Scientific Revolutions*, Second Edition, University of Chicago Press.
- Leeuw F. L. and Vaessen J. 2009. Impact evaluations and development: NONIE guidance on impact evaluation.
- Lucas, H.C., 1975. *Why information systems fail*. Columbia University Press.
- Lukes, S., 2005. Power: a radical view [2005]. *Contemporary sociological theory*, 266.
- Lyytinen K. and Hirscheim R. 1987. Information system failures-a survey and classification of the empirical literature. *Oxford surveys in information technology*, Vol. 4, 257-309
- Maccari, A., 1999. September. The challenges of requirements engineering in mobile telephones industry. In *Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA 99* (pp. 336-339). IEEE.
- Mahoney, M.S., 2004. Finding a history for software engineering. *IEEE Annals of the History of Computing*, 26(1), pp.8-19.
- Mark, A.L., 2007. Modernising healthcare—is the NPfIT for purpose? *Journal of Information Technology*, 22(3), pp.248-256.
- Matson, J.E., Barrett, B.E. and Mellichamp, J.M., 1994. Software development cost estimation using function points. *IEEE Transactions on Software Engineering*, 20(4), pp.275-287.

Maughan, A., 2010. Six reasons why the NHS National Programme for IT failed. *ComputerWeekly.com*, 13, p.45.

McLuhan, M. (1962). *The Gutenberg Galaxy: The Making of Typographic man*. University of Toronto Press

Mead, N.R., 2013, July. A history of the international requirements engineering conference (RE) RE@ 21. In *2013 21st IEEE International Requirements Engineering Conference (RE)* (pp. 21-221). IEEE.

Milne A. and Maiden N., 2012. Power and politics in requirements engineering: embracing the dark side? *Requirements Engineering*, 17(2), pp.83-98.

ModernAnalyst.com (2013), "Roles of the Business Analyst", Available at: <https://www.modernanalyst.com/TheProfession/Roles.aspx> [Accessed: 10th Jan 2021]

Naur, P. and Randell, B., 1969. Software engineering: Report of a conference sponsored by the nato science committee, garmisch, germany, 7th-11th october 1968.

Nelson B. 2008. The politics of Agile.

Available at: <http://pragmaticmarketing.com/resources/the-politics-of-agile> [Accessed 15TH Jan 2016] (Dead link)

Nievergelt, J., 1968. Computers and computing-Past present Future. *IEEE spectrum*, 5(1), pp.57-61.

NIST, 2002. The Economic Impacts of Inadequate Infrastructures for Software Testing. Prepared by RTI for National Institute of Standards and Technology.

Available at: <https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf> [Accessed: 8th Aug 2021]

Nuseibeh, B. and Easterbrook, S., 2000, May. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (pp. 35-46).

Oakley, A., 2000. Experiments in knowing: Gender and method in the social sciences.

OECD (2006), Outline of Principles of Impact Evaluation. Documents for the International Workshop on Impact Evaluation for Development Hosted by the World Bank and the DAC Network on Development Evaluation, Paris - 15 November 2006

Available at:

<http://www.oecd.org/dac/evaluation/dcdndep/internationalworkshoimpactevaluationfordevelopment15november2006-hostedbytheworldbankandthedacevaluationnetwork.htm>

[Accessed 17th Apr 2016]

Oisen, R.P., 1971. As cited in Atkinson, R.(1999).“*Project management: Cost, time and quality, two best guesses and phenomenon, it's time to accept other success criteria.*” *International Journal of Project Management*, 17(6), pp.337-342.

Organisational Behavior 2010. Authors and publisher removed. ISBN: 978-1-946135-15-5. Available at: <http://doi.org/10.24926/8668.1501> [Accessed 17th Mar 2019]

Page D., Williams P. and Boyd D. 1993. *Report of the Public Inquiry into the London Ambulance Service*. HMSO, London (referred to as the Page report).

Paul, D. and Yeates, D. eds., Third edition 2014. *Business analysis*. BCS, The Chartered Institute for IT.

Philipson, G. (2004). A short history of software, the first draft of a chapter commissioned for a book on software development, to be published in 2004 by Routledge [Available at: <http://www.thecorememory.com/SHOS.pdf>]
[Accessed 13th Apr 2012]

Pinto J. K. 1996. Power & politics in project management.

Pohl, K. and Rupp, C. 2nd Edition 2015, *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam-Foundation Level-IREB Compliant*.

Pólya, G. and Pólya, G., 1977. *Mathematical methods in science* (Vol. 26). Cambridge University Press.

Popper, K.R. and Hudson, G.E., 1963. Conjectures and Refutations. *Physics Today*, 16(11), p.80.

Quillard J.A., Rockart J.F., Wilde E., Vernon M., Mock G. (1983). A Study of the Corporate Use of Personal Computers. CISR WP #109, Sloan WP #1512-83, Center for Information Systems Research, Sloan School of Management, Massachusetts Institute of Technology. available at: <http://dspace.mit.edu/bitstream/handle/1721.1/2066/SWP-1512-11139910-CISR-109.pdf?sequence=1>
[Accessed: 10th Mar 2016]

Randell, B., 1996. The 1968/69 NATO software engineering reports. *History of software engineering*, 37.

Randell, B., 2007. A computer scientist's reactions to NPfIT. *Journal of Information Technology*, 22(3), pp.222-234.

Randell, B., 2018. Fifty years of software engineering-or-the view from garmisch. *arXiv preprint arXiv:1805.02742*.

Rice, J.R. and Rosen, S., 1994. History of the computer sciences department at Purdue university. In *Studies in Computer Science* (pp. 45-72). Springer, Boston, MA.

Sadraei, E., Aurum, A., Beydoun, G. and Paech, B., 2007. A field study of the requirements engineering practice in Australian software industry. *Requirements engineering*, 12(3), pp.145-162.

Sanket S. 2019. The exponential cost of fixing bugs. Deepsource.io. Available at: <https://deepsourc.io/blog/exponential-cost-of-fixing-bugs/> [Accessed 8th Aug 2020]

Sauer, C., 1993. *Why information systems fail: a case study approach*. Alfred Waller Ltd., Publishers.

Sauer, C., 1999. Deciding the future for IS failures: not the choice you might think. *Rethinking management information systems*, pp.279-309.

Schwarz, G.M., 2006. Positioning hierarchy in enterprise system change. *New Technology, Work and Employment*, 21(3), pp.252-265.

Shaw M. 1990. Prospects for an Engineering Discipline of Software. In: *IEEE Software*, 7(6):15-24, 1990.

Shaw M., 1998. A profession of software engineering: is there a need? YES: are we ready? NO. In *Proceedings of the 6th ACM SIGSOFT international symposium on Foundations of software engineering. November 1998*. Pages 207–208.

Available at: <https://doi.org/10.1145/288195.295149> [Accessed 12th May 2016]

Shaw M. 2000. *Software Engineering Education: A Roadmap*.

Siadati, R., Wernick, P. and Veneziano, V., 2019a. Learning from history: The case of Software Requirements Engineering. *RE Magazine*, IREB

Available at: <https://re-magazine.ireb.org/articles/learning-from-history-the-case-of-software-requirements-engineering>

Siadati, R., Wernick, P. and Veneziano, V., 2017. Modelling politics in requirements engineering: adding emoji to existing notations. *arXiv preprint arXiv:1703.06101*.

Available at:

<http://arxiv.org/ftp/arxiv/papers/1703/1703.06101.pdf>

Siadati, R., Wernick, P. and Veneziano, V., 2019b. Modelling the Political Context in Requirements Engineering. The System is made for Man, not Man for the System. *arXiv preprint arXiv:1902.00883*.

Available at: <https://arxiv.org/ftp/arxiv/papers/1902/1902.00883.pdf>

Sillitti, A. and Succi, G., 2005. Requirements engineering for agile methods. In *Engineering and Managing Software Requirements* (pp. 309-326). Springer, Berlin, Heidelberg.

Sommerville I. and Sawyer P. 1997. Viewpoints: principles, problems and a practical approach to requirements engineering. *Annals of software engineering*, 3(1), pp.101-130.

Sommerville I. 2007 [1982]. *Software Engineering* (8th ed.). Harlow, England: Pearson Education. P.7. ISBN 0-321-31379-8

Sommerville, I., 2016. Component-based software engineering. *Software Engineering*, 10, pp.464-489.

Standish Group, 1995. *Chaos*. Technical Report. The Standish Group.

Available at: <https://www.projectsmart.co.uk/white-papers/chaos-report.pdf>

[Accessed 29th May 2016]

Stivers, R. and Stirk, P., 2001. *Technology as magic: The triumph of the irrational*. A&C Black.

Suliman, S.M.A. and Kadoda, G., 2017, November. Factors that influence software project cost and schedule estimation. In *2017 Sudan Conference on Computer Science and Information Technology (SCCSIT)* (pp. 1-9). IEEE.

Sutcliffe, A., 2002. *User-centred requirements engineering*. Springer Science & Business Media.

Toole, B.A., 2010. *Ada, the Enchantress of Numbers: Poetical Science*. Betty Alexandra Toole.

Udousoro, I., 2020. Effective Requirement Engineering Process Model in Software Engineering. *Software Engineering*, 8(1), p.1.

UKEssays. November 2018. The London Ambulance Service Computer Information Technology Essay. [online].

Available at: <https://www.ukessays.com/essays/information-technology/the-london-ambulance-service-computer-information-technology-essay.php?vref=1>

[Accessed 20th Nov 2019]

Van Lamsweerde, A., 2000, June. Requirements engineering in the year 00: A research perspective. In *Proceedings of the 22nd international conference on Software engineering* (pp. 5-19).

Walsh B. 2018 All Things Impact

Available at: <http://allthingsimpact.com/home/2018/6/15/d3j1pr707e8jaq2wemztfbnqra83wb>

[Accessed 14th Nov 2020]

Watson Jr, T.J. and Petre, P., 1990. Father, son & co. my life at IBM and beyond.

Wavell, B.B., 1982. Rationality in politics. *Zygon*®, 17(2), pp.151-162.

Wieggers K. E. 2000. [2012] When Telepathy Won't Do: Requirements Engineering Key Practices. Published in Cutter IT Journal, May 2000. Reprinted with permission from Cutter Information Corp.

Wirth N. 2008. A brief History of Software Engineering, IEEE Annals of the History of Computing, vol. 30, no. 3, pp. 32-39, July- Sept. 2008.

Wright, J.N., 1997. Time and budget: the twin imperatives of a project sponsor. *International journal of project management*, 15(3), pp.181-186.

Yang, H. and Liang, P., 2013, December. Reasoning about Stakeholder Groups for Requirements Negotiation based on Power Relationships. In *2013 20th Asia-Pacific Software Engineering Conference (APSEC)* (Vol. 1, pp. 247-254). IEEE.

Young, R., 2006, December. Estimating the value of IT Project Governance. In *International Research Workshop on IT Project Management 2006* (p. 8).

Young, R. and Jordan, E., 2008. Top management support: Mantra or necessity? *International journal of project management*, 26(7), pp.713-725.

Yu E. and Mylopoulos J. 1998. Why Goal-Oriented Requirements Engineering. Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality (8-9 June 1998, Pisa, Italy).

Yu E., 2011. *i**, an agent- and goal-oriented modelling framework. The Knowledge Management Lab, University of Toronto. <https://www.cs.toronto.edu/km/istar/> (last access: 20/05/2021)

Zave, P., 1979, November. A comprehensive approach to requirements problems. In *COMPSAC 79. Proceedings. Computer Software and The IEEE Computer Society's Third International Applications Conference, 1979.* (pp. 117-122). IEEE.

Appendices:

A – Publication:

- **Paper 1:**

Siadati, R., Wernick, P. and Veneziano, V., 2019a. Learning from history: The case of Software Requirements Engineering. RE Magazine, IREB <https://re-magazine.ireb.org/articles/learning-from-history-the-case-of-software-requirements-engineering>

- **Paper 2:**

Siadati, R., Wernick, P. and Veneziano, V., 2017. Modelling politics in requirements engineering: adding emoji to existing notations. *arXiv preprint arXiv:1703.06101*. <http://arxiv.org/ftp/arxiv/papers/1703/1703.06101.pdf>

- **Paper 3:**

Siadati, R., Wernick, P. and Veneziano, V., 2019b. Modelling the Political Context in Requirements Engineering. The System is made for Man, not Man for the System. *arXiv preprint arXiv:1902.00883*. <https://arxiv.org/ftp/arxiv/papers/1902/1902.00883.pdf>


B – Poster

Politics and Power in Software Requirements Engineering

Siadati R., Wernick P., Menon C. and Veneziano V.


ECS Conference (2019), University of Hertfordshire, Hatfield, UK

<https://147.197.131.104/handle/2299/21692>



University of Hertfordshire

School of Engineering and Computer Science



Politics and Power in Software Requirements Engineering

Rana Siadati (presenter)¹, Paul Wernick², Catherine Menon², Vito Veneziano¹

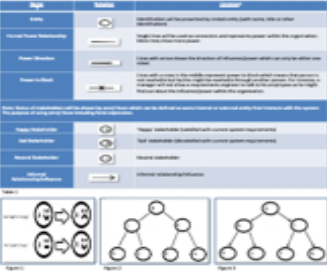
¹ University of Hertfordshire, Hatfield, UK
² r.siadati@herts.ac.uk

Overview

Background Notwithstanding the fact that most of large software projects fail for non-technical reasons (as reported by [1],[2]), traditionally politics and power have been seen as secondary factors, well below and/or after the technical component requirements engineers focus on.

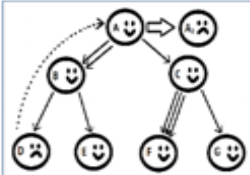
Goal This research study focuses on how politics and power relationships can (and unfortunately usually do) affect in a negative way the outcome of any software requirements engineering exercise within any organization, and it aims at enabling “technical” practitioners (requirements engineers, system analysts, software engineers) at capturing, modelling and somehow managing politics (and power dynamics) within that organization.

Concept Understanding, modelling and managing politics and power relationships within organisations would grant software projects more chances of being successful.



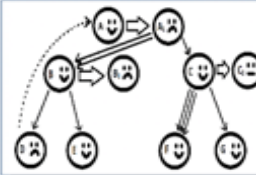
By means of a simple notation, crucial stakeholders and their “power” relationships are identified, analysed and (confidentially) “managed” by the requirements engineer and/or any other “technical” practitioners, to ensure their influence is transformed into a positive force or at least neutralized or mitigated.

Informal organisational relationships



Formal hierarchy, *informal influences*. Not all relevant relationships and influences are reflected in the formal structures documented by official organisational charts. For this reason, we argue that capturing and modelling *informal* power relationships would help practitioners to understand the complex political environment within (and for) which they are working.

Politics in action: managing changes



Managing politics. Other internal or external people, roles or entities that interact with the system and will have “crucial” influence over decisions on its requirements can also be represented in this notation using the dotted ‘informal influence’ lines shown below. Once those crucial stakeholders are identified, focusing on them would improve overall organisational status.

Conclusions & Perspectives

- We argue that project managing and developing software systems and then implementing them within large organisations are challenges that need to rely not just on sound technical expertise and competencies by all technical stakeholders involved, but also on an ability to quietly and confidentially master all those political and power relationships that exist behind and beyond official organisational charts.
- In this research we are using *impact evaluation* and *counterfactual analysis* as a methodology to assess whether and how the simple notational technique we have developed would have been of value to practitioners in their daily activities, mostly within the requirements engineering elicitation and negotiation stages of software development during any of their previous projects.
- We acknowledge this is an area worthy of further investigation, and are currently collecting feedback and comments from practitioners and academics active in the field. Our aim is ensure the concept behind our research and its notational outcome could produce simple and yet effective tools, which practitioners can actually use in their present and future work.

REFERENCES

[1] J. F. Alexander, A Taxonomy of Stakeholders, Human Roles in System Development. International Journal of Technology and Human Interaction, Pearson Education. [2012] Vol 1, 1, 2005, pages 23-59 (2005).

[2] J. Beatty and Chen, A. Visual models for software requirements. [2012]

C – evaluating software tools and methods given certain constraints

Table 23: List of all the criteria adopted for assessing relevance of sources

Criteria
History of Computer Software Systems
History of Software Engineering or other names that previously used for this profession
User requirements
Managements
History of development process and Methodology
Specification
Structured Design Methodology
The reasons for needing RE (e.g., to support factors, benchmarks and long-term process marked during the life of Software Engineering or birth of RE
RE 's other names, role, job description or meaning of RE in old days
RE's role today
Comparison of RE over the years to investigate a possible role change
Possible role changes or its only depends on different systems
Prediction for the future of Requirements Engineering
Prediction of RE's role change in the future
Story from the people who were involved in the history

Table 24: Comparative application of above criteria against two sources

<u>Criteria</u>	<u>Source A</u>	<u>Section</u>	<u>Source B</u>	<u>Page</u>
	Software Engineering Conference Proceeding sponsored by NATO Science Committee Garmisch, Germany, October 1968		A historical perspective on Requirements by Ian F Alexander	
History of Computer Software Systems	<ul style="list-style-type: none"> All the attendees in this conference were concerned with software problems and they all talked about design, production and service of software (Section 2) Failure in understanding System (5.1.1.) Large and Sophisticated Systems (Fig 6) Growth Rate and Demands (6.1.1.) Growth of software errors instead of bug free systems (6.1.3.) Complex systems (7.1.) Frequency of releases (7.1.2.) “Software crisis” or “Software gap” (7.3.) Software pricing or separation of software and hardware (7.3.3.) Advantages and disadvantages of separate pricing (7.3.4.) Rapidly increasing importance of software in many activities of society e.g. air traffic control (1.) 	2. 5.1.1. Fig. 6 6.1.1. 6.1.3. 7.1. 7.1.2. 7.3. 7.3.3. 7.3.4. 1.	<ul style="list-style-type: none"> Software was focused on the machine as a scarce and expensive resource. Attention gradually moved, in software terms, from code to design, and then on to specification. Reference to “The Software Crisis (1972)” Early computers were very expensive to build and operate. Computer time was far more expensive than human time. Computer efficiency came first, with people last. Technical matters turned computer professionals on Human matters turned computer professionals off. Users were troublesome petitioners somewhere at the end of the line who had to be satisfied with what they got. 	1, 2,
History of Software Engineering or other names that previously used for this profession	<ul style="list-style-type: none"> SE conference The nature of Software Engineering Software Engineering Management and Methodology Program design process (all the activities to produce a documented, tested and working program) Section 3.2 Sequence of steps for the programming task (I think it means development) Design process Software Engineering education Software Engineering or “Data systems Engineering” or “Mathematical Engineering” 	1. 3.1. 3.2. 3.3. 4.4. 7.2.	<ul style="list-style-type: none"> Reference to “The Software Crisis (1972)” 	2,
User requirements	<ul style="list-style-type: none"> Influence of users in the design Impossible to keep everyone happy User feedback early in the design process Users do not know what they want 	4.2.2.	<ul style="list-style-type: none"> Users were troublesome petitioners somewhere at the end of the line who had to be satisfied with what they got. Computer time was far more expensive than human time. Computer efficiency came first, with people last. User needs were not often met. Wrong attitude toward users. 	2,
Management	<ul style="list-style-type: none"> Poor reputation of System (Programming) Management Poor reputation of System (Programming) Management Software Management problems Lack of standard Management techniques Lack of progress measurements (Evaluation) Good understanding of their own team member’s job (staff) Organising the software work (development) 	3.2. 5.2.3. 5.2.4.	Still to be done	

History of development process and Methodology	<ul style="list-style-type: none"> • Talk about design, production and service of software • The system development process phase and functions for large-scale software • Design process • Communication within a production team e.g. Verbally • Documentation, Production, Testing, System Evaluation, User feedback • No feedback loop 	1. 3.1. Fig. 1 Fig. 2 4. 5.2.4. 5.3. 5.3.3. 6.3. 6.4. 6.5.	<ul style="list-style-type: none"> • Microsoft first begin conducting usability studies in the late 1980s to figure out how to make their products easier to use their • Microsoft researchers found that 6 to 8 out of 10 users could not understand the user interface and get to most of the features. 	2,
Specification	<ul style="list-style-type: none"> • Advantages and disadvantages of using natural languages to describe specification e.g., English • Conversational or non-conversational languages 	3.1. 4.4.	<ul style="list-style-type: none"> • Move, in software terms, from code to design, and then on to specification. • Specification has come to include a definition of the problem to be solved • Accuracy of specification of requirements • Consulting users when writing the "specifications" for a system • The manual is the <i>external</i> specification of a product • Separation between user and system requirements was discussed. 	1, 2, 3,
Structured Design Methodology	<ul style="list-style-type: none"> • Design Strategies and techniques ➢ Sequencing the design process ➢ Structuring the design 	4.3. 4.3.1. 4.3.1.	Still to be done	
The reasons for needing RE (e.g., to support factors, benchmarks and long-term process marked during the life of Software Engineering or birth of RE	<ul style="list-style-type: none"> • Problem recognition • Complex systems, Software crisis, Growth rate and More complex systems • Rapidly increasing importance of software in many activities of society • Need for specification • Production of large software systems/ growth in software requirements or “Problem of Scale” • Large team on a single project • Failure to complete systems with large programs on schedule or Underestimation of time to complete systems or Over-promising by “software people” • Planning the production of a system or overrunning projects • Performance in software production • Estimation of the cost • “Software crisis or gap” and cause • Lack of awareness of the requirements of the world 	3.1. 5.1.1. 5.1.2. 5.2.1. Fig. 7 7.1.2. 7.1.3. 7.2.	<ul style="list-style-type: none"> • Background of Requirements Engineering • References to the Computer Science literature • Place of requirements in development • Understanding the place of requirements in development (some trends): ➢ Falling price and increased accessibility of computer-based systems ➢ Growing interactivity, ➢ Rising size and cost of system failures ➢ Transform requirements engineering into full engineering disciplines 	1,
RE ‘s other names, role, job description or meaning of RE in old days	<ul style="list-style-type: none"> • Program Manager (possibly!) • Analyst (Possibly!) • Designer • Programmer 	4.3.1.	<ul style="list-style-type: none"> • Definition of Systems design as a technical activity • Technical solutions by systems analyst • Systems design • Understanding of the problem • Requirement’s definition. • Specification or description • Separation between user and system requirements 	2,
RE’s role today	N/A	N/A	N/A	N/A
Comparison of RE over the years to investigate a possible role change s	N/A	N/A	N/A	N/A
Possible role changes or its only depends on different systems	N/A	N/A	N/A	N/A
Prediction for the future of Requirements Engineering	N/A	N/A	<ul style="list-style-type: none"> • How to write specification • Accuracy of specification of requirements • Importance of consulting users when writing the specifications for a system 	3,

Prediction of RE's role change in the future	N/A	N/A	N/A	N/A
Story from the people who were involved in the history	Everywhere in the preceding	All the sections	Everywhere in the preceding	All the sections

D – Analysis

- **Case study 1:**

Sauer, C. 1993. *Why information systems fail: a case study approach*. Alfred Waller Ltd., Publishers.

Available from:

<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.428.7385&rep=rep1&type=pdf> [Accessed 12/01/2020]

- **Case study 2:**

Dolfing H. 2019. Case Study1: The £10 Billion IT Disaster at the NHS.

Available from:

<https://www.henricodolfing.com/2019/01/case-study-10-billion-it-disaster.html> [Accessed 20/05/2020]