Requirements Elicitation in Startup Companies

Jan-Marten Ahrend
University of Birmingham
Edgbaston, Birmingham
West Midlands B12 2TT
+447790233566
jma263@bham.ac.uk

ABSTRACT
Much has been written about Requirements Elicitation within small and large organisations. Surprisingly, the organisation type played a small role in the software engineering literature and therefore startups, a special type of Research & Development environment, did not receive the attention they reserve in research. Startups are a major factor of the economy’s growth and researchers in economic sciences have realised, that existing processes cannot simply be applied to startups. In this paper, a research project on Requirements Elicitation in startup companies is presented. A definition of startups and their distinction from traditional companies is given, Requirements Engineering and Elicitation, as well as their role in Software Engineering and Software Development investigated. A literature review is looking at existing work concentrating on Software Engineering in startups and is showing a lack of empirical research and the need for proper guidance on the handling of requirements in startups. An evaluation-based research approach is introduced that consists of a survey and multiple-case study. Its goal is to understand Requirements Engineering in startups, to provide empirical evidence of the usefulness of identified methodologies and propose guidelines for startups to adopt software methodologies that support Requirements Elicitation. Guidance for software startups will be beneficial, due to their lack of time and money for designing their own development processes. Furthermore, the outcomes of this study could be applied to other organisational forms that intend to foster agility, and to establish an innovative model to study implications of innovation, creativity, immaturity and speed-to-market through concentrated research.

General Terms

Keywords
Requirements Elicitation, Startup Companies, Research and Development.

1. INTRODUCTION
Two days after my graduation I find myself in a plane to San Francisco to start an unpaid internship at a startup called Fair Observer, an online media platform with over 300 contributors in 30 countries. Fair Observer was solely financed by the founders, their families and friends, and money was running low because of high expenses and low income. We would need to close our platform if we could not find an investor by December 2012. The team had high pressure of generating income and at the same time meet the expectations of potential investors. We realised from early on the necessity to provide a significantly higher user value compared to existing solutions, to successfully position our new product. Achieving significant higher user value is the outcome of innovation, the heart of most startups, and accompanied with increased creative thinking during the requirements process. At the day of my arrival, I was shown wireframes of a new mobile application that I was supposed to start developing that day. The team was fast. Driven by the fear to shut down, what we believed to be change the world, mockups and wireframes were developed at the speed of light and were ready to be developed. Not soaked in yet by the pressure of time and money, I foolishly asked one essential question: “Why? Why have we chosen a mobile strategy? Why do we have to develop a new platform in the first place? What was needed to achieve our goals? What are our goals?” Not only were requirements not documented to efficiently transfer knowledge to new team members like me, but requirements were not defined at all. No one could clearly explain why we did what we did.

1.1 Startup Companies
My story is far from unique. At a high level, software development processes are driven by the business goals and although all businesses have to cope with the similar challenges, certain issues tend to be presented in software
driven startups to an extreme degree (Rifkin, 1999). Fields outside computer science realised a long time ago that startups are not a smaller version of larger companies and that existing processes to run larger companies cannot simply be applied to startups. Characteristics that are widely representative for software startup companies lie within business and engineering concerns. A software startup is not defined by its size – it can consist of either a small or large development team – but often by its youth and immaturity (Sutton, 2000). Small companies have in comparison to startups “fewer internal communication and coordination problems; greater flexibility and reactivity (…), a foundation of established products, partners, and customers; and possibly a greater shared history and vision” (Sutton, 2000; p. 35). They tend to have a young and inexperienced character with limited operating history, limited resources, time-to-market pressure and operate in dynamic technologies and markets (Sutton, 2000; Zettel et al., 2001; Loch et al., 2008). Externally raised capital is usually raised over several rounds of financing to cover their expenses. To secure the next rounds of financing, software developers tend to concentrate on short-term achievements, instead of following a long-term orientation (Zettel et al., 2001). At the same time, startups try to reduce the number of financing rounds by generating earnings as soon as possible to reduce the dependence on other parties. Blank (2012) describes a startup company as a new venture, new division or unit within a larger company, which main task is to search for a repeatable business model. In Graham’s definition (2012), a startup is a company centred on growth, and does not need to be newly founded, or work on technology, rely on venture funding or needs to have an exit strategy. Furthermore, larger companies have started implementing ‘internal start-ups’, sub-divisions relatively independent form their parent company to foster innovation (Economist, 2011).

According to data from the Kauffman Index of Entrepreneurial Activity (Fairlie, 2012), an average of 0.32% of the American adult population founded a new company each month in 2011, leading to 534,000 new businesses per month. The Index collected data on entrepreneurship throughout several industry groups: construction, manufacturing, trade, services and other. As of 2011, 6.3% of Americans are self-employed business owners (a total of 11.5 million). Most of entrepreneurs are between 20 and 34 years old (30%) and only 72% native-born. Fairlie further describes two types of patterns in business creation: choosing to start a high-potential businesses and business ownership forced due to lack of job opportunities.

**Research & Development: The Pivot**

Startups act under high uncertainty and spend most of their time searching to validate requirements and market needs. Unlike in traditional scenarios where the problem is known and the solution is unknown, both problem and solution are unknown. A startup can also be perceived as Research & Development Company due to their focus on discovering and gathering knowledge for a new product or service. The startup’s success thereby is dependent on its ability to **pivot**, a term associated with a course change of the product, strategy or engine of growth (Ries, 2011). Pivots are a common characteristic and often necessary make the company succeed. The founders of Tune In Hook Up, an online dating platform, realised that users were mainly using their service to upload and share videos and hence pivoted to what is today known as the video-sharing website **YouTube** (Hopkins, 2006). An other example of a pivot is the photo-sharing and social networking service **Instagram**, which was once called **Burbn**, a location-sharing service (Sengupta, 2012). Although only examples, the pivot represents a differentiation factor that is not found in the core of other organisational forms. The Startup Ecosystem Report found that “[s]tartups that pivot once or twice times raise 2.5x more money, have 3.6x better user growth, and are 52% less likely to scale prematurely than startups that pivot more than 2 times or not at all” (Marmer et al., 2011, p. 5).

**Types of Startups**

Marmer et al. (2011) identified four major groups of startups and claim that B2C (business to costumer) and B2B (business to business) are no meaningful segmentation for startups. The four types of startups differ in their customer acquisition, time, product, market and team (Marmer et al., 2011, p. 7 - 9):

**Type 1**: Self-service customer acquisition, consumer focused, product centric, fast execution, often atomise a manual process.
- Market size is 2x bigger for Type 1 compared to Type 3.
- More likely to tackle existing markets.
- Need the least capital of all types.

Examples: Google, Dropbox, Eventbrite, Slideshare.

**Type 2**: Self-service customer acquisition, critical mass, runaway user growth, winner take all markets, complex User Experience, network effects, typically create new ways for people to interact.
- Need 50% longer than Type 1 and Type 3 to reach scale stage.
- Business heavy and balanced teams perform better than technology heavy teams.
- Market size is 2x bigger for Type 2 compared to Type 2.
- More likely to tackle new markets.
- More likely to have large team growth at the scale stage.
- Need more capital than Type 2 and Type 3.
- More likely to have large user growth.
Examples: Ebay, Skype, Airbnb, Flickr, Facebook.

Type 3: Lead generation with inside sales reps, high certainty, product centric, early monetization, small and medium enterprise focused, smaller markets, often take innovations from consumer Internet and rebuild it for smaller enterprises.
- Business heavy and balanced founding teams perform better than technology heavy teams.
- More likely to tackle existing markets with a product that is cheaper.
- More likely to maintain small teams even when they scale.
- Monetize a high percentage of their users.
Examples: PBworks, Kissmetrics, Xignite, Zendesk.

1.2 Requirements Engineering
In 1977, the first requirements method Structured Analysis and Design Technique was introduced (Ross, 1977), after the call for professionalization in the “Software Crisis” in 1968 and the realisation that software, not hardware is now the limiting factor of a project’s success (Boehm, 1884). In the following years, Requirements Engineering, as a process in Software Engineering, gained wider recognition and followed engineering trends, such as structured methodologies, object orientation, formal methodology and aspect orientation. The first international conference was held in 1993, and first dedicated journal published in 1996. Nuseibeh and Easterbrook (2000) defined software systems requirements engineering (RE) in their widely recognised road map as the process of discovering the purpose of the intention of a software system. The degree to which it meets this purpose is thereby the primary measure of its success. “The success of a software system depends on how well it fits the needs of its users and its environment” (Cheng & Atlee, 2007, p. 285). In order to discover the purpose, stakeholders and their needs must be identified, documented, analysed, communicated and subsequently implemented (Nuseibeh & Easterbrook, 2000). These activities resulted in the division into sub-processes, such as: Requirements Elicitation, Requirements Analysis, Requirements Management, Requirements Specification and Requirement Validation. Requirements are thereby distinguished between functional and non-functional requirements (Nuseibeh & Easterbrook, 2000). Functional requirements represent concrete functions, capabilities, or services, which the system must provide. Non-functional requirements are properties or qualities, which the functions, capabilities, or services must possess.
With growing complexity of software systems, the quality of requirements analysis and management remains one of the greatest sources of risk (Smite, 2006). Studies in the recent years have shown the importance and problematic nature of RE. The major factor in the failure of large software projects is of 90% due to the accuracy in capturing system requirements (Davis et al., 2006). A software project is more likely to fail due to poor requirements management (71%), than because of bad technology, missed deadlines, and change management issues (Lindquist, 2005). Additional studies revealed that the correction of mistakes made at the elicitation stage accounts for 75% of all error removal costs (Davey & Cope, 2008) and poor execution of requirements elicitation will almost guarantee the failure of the final outcome (Hickey & Davis, 2003). The consequences of costs associated with failure of RE can be illustrated by looking at a case study of the London Ambulance Service (LAS) Computer Aided Dispatch (CAD) System. The tragic failure of the system lead to the attribution of the death of three patients and an estimated financial cost of £1.1-£1.5 million. The failure was caused due to the lack of user participation in the project and the misunderstanding of work practices, among other reasons (Finkelstein et al., 1996).

### Requirements Elicitation

As the first step in Requirements Engineering, information is gathered for subsequent analysis, modelling and validation (Goguen & Jirotka, 1994). The process is called Requirements Elicitation and its goal is to identify stakeholders, underlying problems that need to be addressed, and as a result identify system boundaries and goals (Nuseibeh & Easterbrook, 2000). Stakeholders are individuals or organisations affected by the system (e.g. users, developers, clients, investors) and are divided into user classes (Sharp et al., 1999). Goals define the objectives the system has to meet and are broken into high-level (business related) and lower-level goals (technical goals). The focus in Requirements Elicitation is purely on the problems and needs of the stakeholders and does not look into the solution domain (Nuseibeh & Easterbrook, 2000).

It is acknowledged, that the articulation of requirements is difficult for users and elicitation techniques therefore concentrate on collecting information about the current state of the situation (e.g. tasks users currently perform) and those they could preferring to perform (Johnson, 1992). Various elicitation techniques are in use, whose selection is based on the information needed to be obtained and constraints of time and resources. Nuseibeh and Easterbrook (2000) summarise a number of elicitation technique classes in their RE roadmap:

- **Traditional techniques**, a broad range of data gathering techniques.
- **E.g. Surveys, interviews, analysis of existing documentation.**
- **Group elicitation techniques** to get a richer understanding of the needs and supporting stakeholder agreement and buy-in. **E.g. brainstorming, focus groups, RAD/JAD workshops.**
- **Prototyping**, to gather early feedback and to address great uncertainty. **E.g. paper prototyping, software prototyping.**
- **Model-driven techniques**, to drive the elicitation process by providing a model matched to the information that is to be elicited. **E.g. goal-based methods, scenario-based methods.**
- **Cognitive techniques**, to gather information for knowledge-based systems. **E.g. protocol analysis, laddering, card sorting, repertory grids.**
- **Contextual techniques**, to elicit information by using ethnographic techniques. **E.g. participant observation, conversation analysis, ethnography.**

The techniques applied in a particular project to elicit requirements are closely related to other RE activities, such as modelling schemes. Requirements Modelling schemes often imply particular elicitation techniques (Nuseibeh & Easterbrook, 2000).

### 1.3 Requirements Engineering in Software Development

Looking at the discussion of development processes, it is surprising to see that the organisation type plays a small role in the software engineering literature (e.g. Dybå & Dingsøyr, 2008). Abrahamsson et al. (2003) conclude in their comparative analysis that Agile Software Methodologies (ASM) were found to be effective and suitable for many situations and environments but guidance is needed on how the methods should be used in certain software development situations (Nuseibeh & Easterbrook, 2000). “Empirical works that study the effects of particular (agile) methods, their ease of use, costs, and possible negative implications for different sizes and lines of business, are needed in particular” (Abrahamsson et al., 2003, p. 252).

Requirements Engineering is a process in Software Engineering and is applied in various forms within Software Development Models. Traditional software methodologies, like the Waterfall Model, consist of extensive planning to make development a predictable activity (Boehm, 2002). Requirements are separated from...
the design of the system in traditional methods, are defined in the beginning of the life cycle and are therefore believed to be less suited for the frequently changing environment of a startup (Nuseibeh, 2001). On the other hand, iterative and incremental software development methodologies, such as ASM, are said to be better suited. They were not intended to replace traditional methodologies, but create a symbiotic relationship, dependent on the application domain and innovativeness (Cohen et al., 2004). Agile methodologies intend to have continuous control of the software requirements and continuously validate them. They aim to provide maximum flexibility regarding changing requirements and focuses on a better understanding of what the user needs are. Agile software development consists of various methodologies, of which Extreme Programming, Scrum and Crystal are the most well-known.

Agile methodologies in general were found to be effective and suitable for many situations and environments, but very few empirical studies support this claim (Nuseibeh & Easterbrook, 2000). Agile methods cannot be applied by book to every situation and environment and need to be adopted (e.g. to startups). Previous studies have adopted ASM to small and large software teams. Most studies researched the introduction and adoption, human and social factors, the perceptions on agile methods and conducted comparative studies on small software development teams (Dybå & Dingsøyr, 2008). More recent studies have also proven the implementation of agile processes to be beneficial in larger teams. But apart from the distinction between the size of the development team, other factors were identified that would influence the software engineering. The company’s size in terms of employees working on the product, knowledge and their experience and hardware resources determine the process (Lindvall & Rus, 2000).

1.4 Software Engineering in Startups

The probability for a startup to be successful is dependent on many internal and external factors, such as human elements, capital funding, processes, location and luck. One factor that is controllable by the startup is the processes that are being used to operate and steer the startup. The processes need to match the environment in which they are applied. The following attributes were earlier defined as being characteristic for a startup environment:

- In-experienced
- Time-to-market pressure
- Operating in a dynamic market
- Search for markets and needs (pivoting)

Due to the startup’s focus on searching and validating hypothesis concerning the user’s needs, processes are needed that closely interact with users. Startups were often associated with ASM (e.g. Sharp & Robinson, 2004). Although an association in this context seems natural, the agility of a startup was not assessed yet (e.g. using the Agility Measurement Index). Zettel et al. (2001) claim that startup companies do not make use of established software methodologies by book and the used practices are immature and ad-hoc, as well as implemented by accident. Own observations support this statement. Sharp & Robinson (2004) conducted an ethnographic study of Extreme Programming (XP) practices in a startup to characterise the XP culture. They were “struck” by the lack of requirements documents, but concluded that XP showed to be useful in their case (Sharp & Robinson, 2004; p. 373).

Another methodology that claims to support the environment of a startup is Lean Software Development (LSD). LSD is an agile software methodology adapted from lean manufacturing, a socio-technical system first used in Toyota’s production system in the 1950s (Poppendieck & Poppendieck, 2003). Lean has its focus on the user, and measures success by the value provided to the user. LSD refers to seven principles; eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity in, see the whole. Poppendieck & Poppendieck (2003), from whom the term Lean Software Development originates, further introduced 22 practices to manage the principles. Dybå & Dingsøyr’s (2008) found in their systematic review of Agile Software Development, that only 3% of studies researched LSD.

Academic research on Software Engineering in startups is otherwise, since most work lacks empirical support and is based on the experiences of practitioners or consultants. Eric Ries’ New York Times best seller “The Lean Startup” (2011) knows to sell the lean methodologies with unassailable marketing expertise to startup companies. His claims on the effectiveness of such approaches are based on anecdotal evidence and not empirical research. The Lean Startup Movement and personal experiences and observations indicate a strong need for proper guidance on the handling of requirements in startups. By contrast, the lack of academic work is not the case in economic sciences. Universities worldwide offer courses on principles and methods on starting and maintain startup companies.

1.5 Requirements Engineering in Startups

One of the biggest challenges in Requirements Engineering is to select the appropriate and best-suited method (Tsumaki & Tamai, 2005). Elicitation techniques are commonly selected based on a combination of: a) knowledge of a particular technique, b) the individuals preference, c) the application of some explicit methodology that supports a particular technique and d) the intuitive understanding of a technique suitable to the current circumstance (Hickey & Davis, 2003). Methods provide guidance for the variety of techniques the requirements
engineer has to choose from. With individual strengths and weaknesses, methods are suited for specific scenarios and need to be chosen appropriately (Nuseibeh & Easterbrook, 2000). In some cases, as Maiden and Rugg (1996) argue, technique-selection guidance is more advantageous than a rigid elicitation method. With its multi-disciplinary nature, Requirements Engineering techniques are conducted in different stages of the development cycle (Brinkkeeper & Joosten, 1996). This lead to the application of Method Engineering, which concentrates on designing methods to deploy requirement methods for particular situations or problems.

To summarise, the elicitation of requirements is of special interest in startups due to their focus on searching and validating market needs. We have shown, that startups are a special environment, for which Agile Software Methodologies need to adapted to. One such methodology is Lean Software Development, which believed effectiveness for startups is based on anecdotal evidence.

2. METHOD
A literature review will be conducted prior to primary research, making use of the proposed guidelines for systematic reviews for software engineering researchers by Kitchenham (2004).

Information Sources
The search strategy will focus on the following rang of information sources:
- Conference papers
- Journal articles
- Technical reports
- Theses.

Papers based on expert opinion will be retrieved as well, with limited significance. Preferred language of the papers is English and German and the following electronic databases will be searched:
- ACM Digital Library
- Compendex
- Google Scholar
- IEEE Xplore
- ISI Web of Science
- Kluwer Online
- Science Direct: Elsevier
- SpringerLink
- Wiley Inter Science Journal Finder

2.1 Research Questions
The research question will be refined after the literature review, from the following:

Q1: How do startup companies elicit software requirements (considering the startup types)?
Q2: Do the practices match known software engineering methodologies or principles (e.g. ASM: LSD)?
Q3: Is the introduction based on an adoption by book or by accident?
Q4: What are the challenges when conducting Requirements Elicitation in startup companies?
Q5: How useful is Lean Software Development for eliciting requirements in startup companies (considering the startup types)?
Q6: What implications have venture capital investors on Requirements Elicitation?

2.2 Research Method
The research method follows an evaluation-based research approach, which first assesses the state of the practice and then evaluates proposed solutions to the state of the art. The evaluation-based approach was chosen over a solution-based research approach due to the lack of existing research that looked into the practices currently applied. The solution-based would instead identify RE problems in startups, propose ideas for solving those and be supported by a proof-of-concept study to show their potential.

The research method consists of a survey and a case study. The empirical research methodology by Petersen (2010), which evaluated the usefulness of agile practices in a large-scale industrial setting, will be adapted. Startups will be studied in real life situation due to the practicality of software engineering.

A survey was chosen to do a first research in the large and help identify subjects for the case study, as suggested by previous studies (e.g. Cheng & Atlee, 2007; Nuseibeh & Easterbrook, 2000). The motivation to choose case studies as the primary research method was implied by the objective to investigate software requirements in startups in a real-world setting and their previous success and acceptance in evaluating research ideas when applied to the industry (e.g. Cheng & Atlee, 2007).

Requirement practices will be studied in an online questionnaire on a sample of software startups. The data will be used to draw conclusions on the software requirements and software development in startup companies (Q1 – Q3). A multiple-case study will investigate requirements in software startups in-depth. The case study is of confirmative nature and will enable to draw conclusions on software requirements in startup companies (Q1-Q6). The study design is fixed and agreed on with the startup companies and involved investors. The analysis of case studies ranges from a purely qualitative analysis where raw data from interviewees is categorised and coded to the exclusive use of statistical inference. Participants will
reflect and give feedback on the usefulness of the methodologies for handling requirements and challenges in working with them. Sources for quantitative data will include closed company data and data from internal reporting tools of the business incubators under provision of quality checks to assure validity.

Alternative research methodologies have been taken into consideration. It will not be insightful to replicate the environment in an experiment, or obtain similar qualitative data in a simulation. Action research is a suitable alternative, but relies on the willingness of the startups to change their processes and is associated with greater efforts. Semi-structured interviews and workshops will be used as sub-methods for data collection.

2.3 Measures
The online questionnaire will collect data on team size, level of agile experience, level of development experience, project duration and domain, as well as questions directed at the way they develop software requirements at the present and questions to assign their startup type.

Qualitative variables of the case study include:
- Structure, plan and control of the development process.
- Stakeholders influence on the selection of software development methodologies.
- Characteristics of requirements elicitation, analysis, specification and management.
- Perception of bottlenecks, unnecessary work, avoidable rework for the requirements.
- Feedback from practitioners on approaches they believe to be useful.
- Ability of the approaches to support practitioners to elicit, analyse, define and manage software requirements.
- Ability of the approaches to show presence of inefficiencies or opportunities.
- Investors' influence on software requirements.

2.4 Participants
Inclusion factors for the survey (N=100) include a limited operating history, a scalable or repeatable business model and software development as a key activity. Participating startups will be based in Silicon Valley (USA), London and Oxford (UK) and Munich and Berlin (Germany). These three locations were chosen due to their known importance and influence for entrepreneurial thinking and innovation. Eight startups will participate in the case studies (two of each startup type) and will be selected using the data obtained in the survey, based on their representativeness for the population.

The importance of access to startups that are willing to participate is recognised and the risk minimized by the use of established connections to startup companies and startup programs. Personal relations to contact persons in the following places were established through my attendance at workshops, conferences, competitions and employment:
- Plug and Play Tech Center: global accelerator (Silicon Valley)
- Birmingham Science Park Aston (UK)
- Oxygen: accelerator (UK)
- Entrepreneurs for the Future: incubator and accelerator (UK)
- Central Working: incubator (UK)
- Entrepreneur First: incubator (UK)
- F6S: startup community (UK)
- Oxford Entrepreneurs: student society (UK)
- Cambridge University Entrepreneurs: student society (UK)
- Entrepreneurship Center of the Ludwig Maximilian University of Munich: incubator (Germany)
- Strascheg Center for Entrepreneurship: incubator (Germany)
- Social Entrepreneurship Akademie: incubator (Germany)
- Thinc!: student initiative for entrepreneurs by the University of Mannheim (Germany)
- Vodafone Stiftung Deutschland: supporting of social businesses (Germany)
- BonVenture: supporting social entrepreneurs (Germany)
- Faculty of Organizational Sciences by the University of Belgrade: incubator (Serbia)
- StartupWeekend: startup competition by the Kauffman Foundation (international)

2.5 Outcomes
The intended final outcomes of the research will be:
- Understanding of Requirements Elicitation in startups.
- Empirical evidence of the usefulness of Lean Software Development in a software startup setting to elicit requirements.
- Proposal of guidelines for startups to adopt Lean Software Development that support Requirements Elicitation.

3. SIGNIFICANCE
Startup companies are a major striving factor of the economy’s growth (Autio et al., 2012). Researchers in economics sciences realised that startups are not a smaller version of larger companies and that existing processes cannot simply applied to startups. There is a great need to adopt the software methodologies that are being in use to
the special environment of a startup and little is still known
about the processes in use to elicit requirements. Since
startups are more likely to fail from a lack of customers
than from a failure of product development, it is essential to
find suitable approaches to develop requirements
effectively. Guidance for software startups will be
beneficial, due to their lack of time and money for
designing their own development processes.

The outcomes of this study could also be applied
to other organisational forms that intend to become more
agile. It is further hoped that the research will establish an
innovative model to study implications of innovation,
creativity, immaturity and speed-to-market in a
development process. These characteristics are present to
an extreme degree in startups, which offers the opportunity
for concentrated research. Speaking of an example, the
creativity and innovation in startups is most likely resulting
in ethical conflicts that need to be handled faster then usual
due to the speed-to-market.

4. CONCLUSION
Empirical research on Software Engineering in startup
companies is still scarce although their unique
characteristics are likely to have implications on the
process. This research aims to provide a better understanding of Requirements Elicitation in startups,
empirical evidence of the usefulness of identified software
development methodologies to develop requirements in this
setting and provides guidelines on their adoption.
Innovation, creativity, immaturity and speed-to-market are
present to an extreme degree in startup companies and the
combination could lead to new insights and discussions in
Software Engineering.

5. REFERENCES
comparative analysis. In Software Engineering, 2003. Proceedings. 25th International Conference on (pp. 244-
254). Iee.
Autilo, E., Cleevley, M., Hart, M., Levie, J., Acs, Z., & Szerb, L. (2012). Entrepreneurial Profile of the UK in the
Light of the Global Entrepreneurship and Development Index. Available at SSRN 2070320.
Software Technology, 38(4), 259.
1-66.
Information Technology, 5(1), 543-551.
elicitation techniques: Empirical results derived from a systematic review. In Requirements Engineering, 14th
IEEE International Conference (pp. 179-188). IEEE.
software development: A systematic review. Information and software technology, 50(9), 833-859.
Economist (December, 2011). “Hong Kong in Honduras”. Retrieved Feb 20, 2013, from
http://www.economist.com/node/21541392
Fairlie, R. (2012). Kauffman Index of Entrepreneurial
Communications of the ACM, 43(3), 115-118.
errors: the London Ambulance Service case study. In
Proceedings of the 8th International Workshop on Software
Hopkins, J. (2006). Surprise! There’s a third YouTube co-
youtube-karim_x.htm
McGraw-Hill.
Mess; The requirements process, literally, deciding what
should be included in software, is destroying projects in
ways that aren't evident until it's too late. Some CIOs are stepping in to rewrite the rules. CIO, 19, 53-60.


