OPTIMAL CAPABILITY MATURITY IN STARTUP SOFTWARE COMPANIES

A Dissertation
Submitted to
The Temple University Graduate Board

In Partial Fulfillment
of the Requirements for the Degree
DOCTOR OF BUSINESS ADMINISTRATION

By
Michael Thomas Bujnowski
Diploma Date May 2017

ABSTRACT

Overview

This dissertation presents the theory of optimal capability maturity as a function of company size. Optimal capability maturity is presented within the framework of growth stage models, which purports potential crises between stages threaten the success and survival of a company. The first three chapters lay the foundation for analysis that culminates in the proposed Optimal Maturity Model. Research encompasses a series of mini case studies, with data collected via semi-structured interviews. The objective of this research is to create a diagnostic tool for company analysis, as well an aid for small software companies to operate efficiently and to navigate growth risks.

The Chapters

Chapter 1 presents the Software Company Growth-Stage Model that contends there a relationship between company size and required capability maturity. The independent variable, company size, is defined as core employees or those engaged in product development. This is in contrast to prior models that use time as the independent variable.

Chapter 2 offers a Capability Maturity Appraisal Instrument as a capability maturity appraisal tool used in the research. This tool has potential to be used for thumbnail appraisals in the practitioner community.

Chapter 3 presents complexity as a moderator of required maturity. The Software Platform Complexity Model extends existing theory on software complexity to provide a clear conceptual model for understanding and categorizing complexity differences in software platforms. Augmenting the model, Complexity Factors are compiled through literature and interviews, further providing insight into potential moderators of required capability maturity.

Chapter 4 proposes the Optimal Maturity Model. A key distinction in this model is the difference between “optimal” and “optimizing”. “Optimal” capability maturity could be restated “adequate” or “sufficient” maturity. In contrast, “optimizing” capability maturity is the highest level of capability maturity in which systems are measured, reviewed, and continually improved. The theory of the Optimal Maturity Model is that companies need not consume excess resources advancing capability maturity, and that optimal maturity, separate from complexity, can be approximately determined based company size.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1.

SOFTWARE COMPANY GROWTH-STAGE MODEL

Introduction

Growth-stage theory sets the stage by explaining, in part, why startup and small software companies often fail. At each stage of growth (or contraction), a company may face crises that threaten success and survival. A Software Company Growth-Stage Model is proposed to explain the relationship between the size of a company and its required capability maturity (e.g. operating systems, processes, and procedures).

Existing growth-stage theory generally consists of defined stages, with potential crises inherent at the transition points between stages. The Software Company Growth-Stage Model builds upon existing literature. However, unlike existing growth-stage models with the independent variable as "time," the Software Company Growth-Stage Model sets the independent variable as "employees" involved in product development. The Software Company Growth This model is tested using field case studies.

Literature Review

In the literature, there are several analogous terms that describe "maturity." "Organizational maturity" is often used in growth-stage papers. "Capability maturity" is used in practitioner literature as well as in strategic literature in business science. "Maturity processes and procedures" is commonly used in engineering, and may be the most universal. Although in many writings these terms may be interchangeable, in this paper the similar the term "capability maturity" will be used. In part the rationale for using capability maturity is because the term is consistent with the Capability Maturity Model of the CMMI Institute, which is utilized in this research.

The seminal work by Greiner (1972) provided a growth-stage model (Figure 1) with five phases. Each phase has an associated Evolutionary stage and a Crisis stage. During the Evolutionary stage, the organizational structure, management practices and operational processes remain relatively consistent. Following time and growth, the organization will face a series of crisis points. In order to advance the organization, change is necessitated in its structure, practices, and processes. Greiner proposes that an organization must encounter and overcome a crisis before it can progress to the next Growth Phase, and that the Phases occur in sequence.

Five Phases of Growth

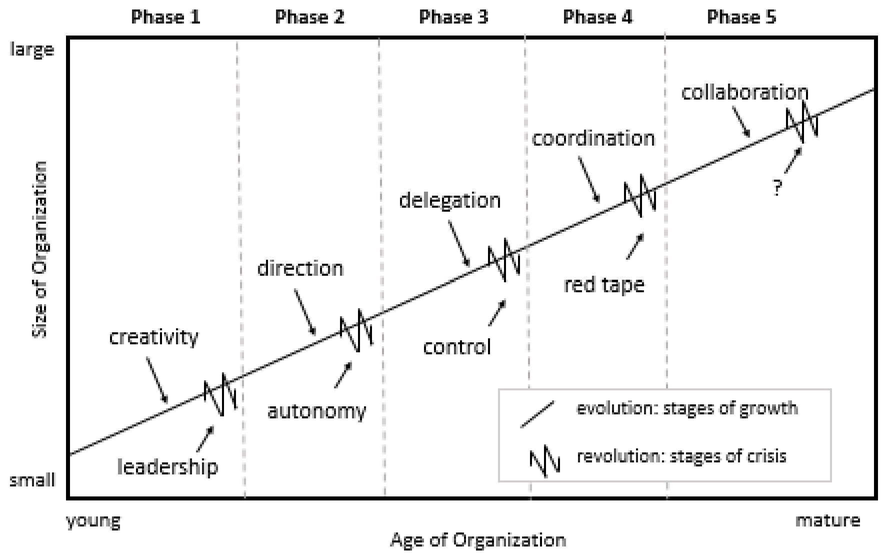
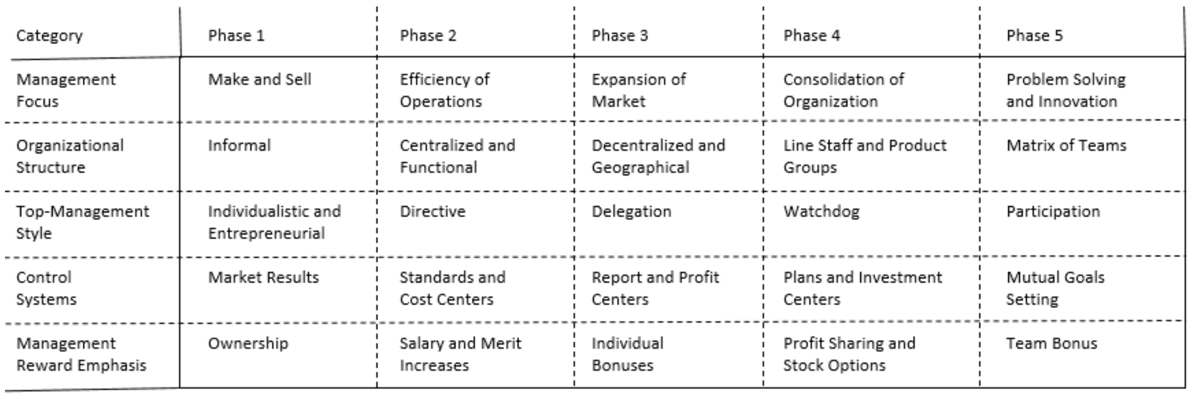


Figure 1. Greiner (1972, page 5). "Evolution and Revolution as Organizations Grow"

In each of the phases of the Greiner Five Phases of Growth model, there is a defined set of distinguishing characteristics. These characteristics are divided into five categories describing the nature of a company's operational practices. (Refer to Table 1.) Greiner's definitions are not evidence-based and thus not fully embraced in later works; however the model does provide a foundation for the understanding of typical characteristics of stages of growth.

Organizational Practices in the Five Phases of Growth



Category	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
Management Focus	Make and Sell	Efficiency of Operations	Expansion of Market	Consolidation of Organization	Problem Solving and Innovation
Organizational Structure	Informal	Centralized and Functional	Decentralized and Geographical	Line Staff and Product Groups	Matrix of Teams
Top-Management Style	Individualistic and Entrepreneurial	Directive	Delegation	Watchdog	Participation
Control Systems	Market Results	Standards and Cost Centers	Report and Profit Centers	Plans and Investment Centers	Mutual Goals Setting
Management Reward Emphasis	Ownership	Salary and Merit Increases	Individual Bonuses	Profit Sharing and Stock Options	Team Bonus

Table 1. Greiner (1972, page 8). “Evolution and Revolution as Organizations Grow”

The work of Steinmetz (1969) predates Greiner, but the work is not as well cited and does not provide the same degree of description and rationale for each growth stage. Steinmetz suggested four stages of organizational growth with three Critical Phases between each stage of growth. The Steinmetz model is notable for suggesting numbers of employees and assets a company is likely to have, along with a guide of notable challenges that a company may face during each Critical Phase. Two notes of interest are that Steimetz does not define the y-axis of his conceptual model, and his definition of a small business exceeds 1,000 employees.

Stage of Organizational Growth and Their Critical Phases

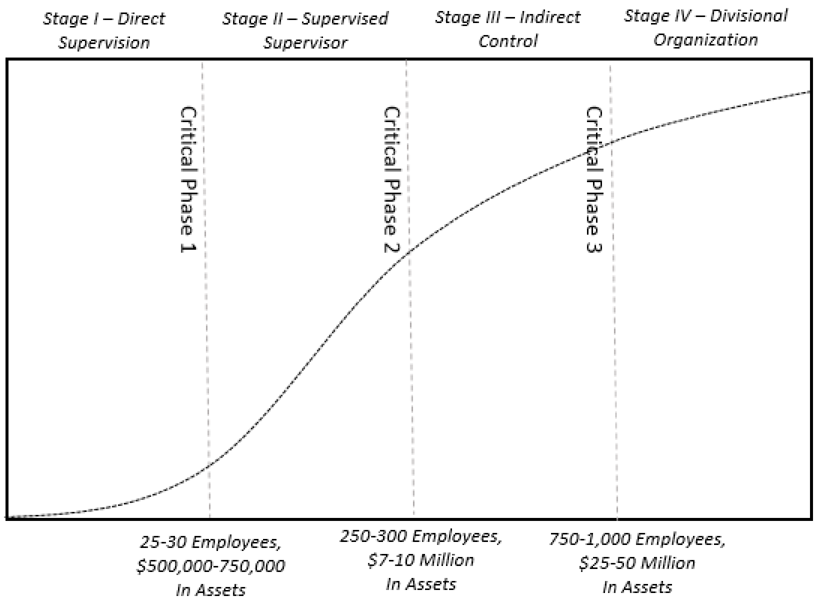


Figure 2. Steinmetz (1969, page 38). "Five Stages of Growth in Small Business"

The growth model (Figure 3), "Five Stages of Growth in Small Business" by Scott and Bruce (1987), builds upon prior works, particularly the Greiner model. Similar to Greiner, the Scott and Bruce model is dependent on time and a sequential progression of stages. Scott and Bruce also produce extensive definitions of stage characteristics, but they state that stage characteristics are only a guide - they are not deterministic. This model emphasizes that the growth process is not linear and that crisis may result in one of three outcomes: "overcoming" the crises to continue with growth, "containment" of the crises with limited further growth, or "failure" ("fold" or decline) of the company.

Five Stages of Growth in Small Business

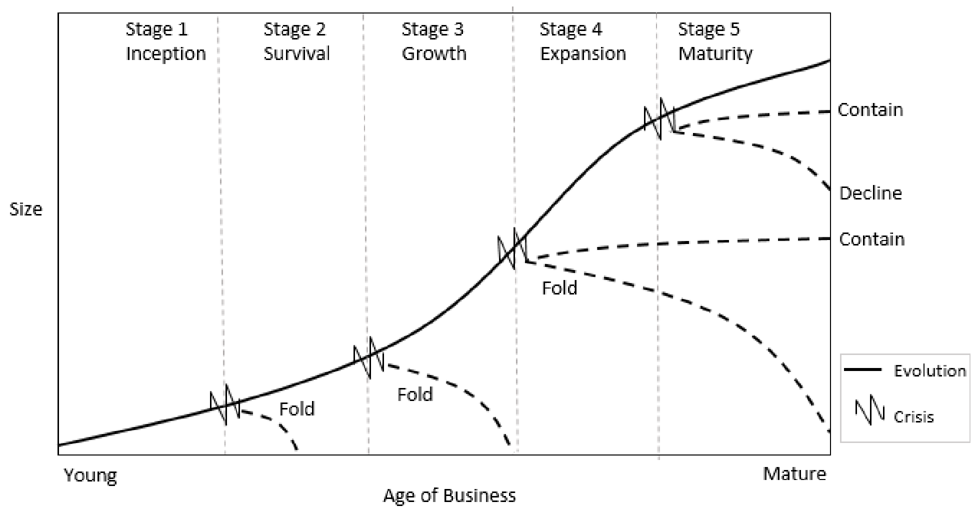


Figure 3. Scott & Bruce (1987, page 47). “Five Stages of Growth in Small Business”

Gaibraith (1982), proposed a stage-growth theory specific to technology ventures, titled “Organization Development Model.” In this model, there is a strong product orientation displaying the name of each stage (see Table 2).

Organization Development Model

	Description
Stage I	Proof of Principle
Stage II	Model Shop
Stage III	Start-Up Volume Production
Stage IV	Natural Growth
Stage V	Strategic Maneuvering

Table 2. Gaibraith (1982, page 74). “The Stages of Growth”

Like the Greiner and the Scott and Bruce models, the stages of the Gaibraith model are sequential. Unlike the Greiner and the Scott and Bruce model, there is no identified time or company size component; instead the model offers a life-cycle stage.

Gaibraith provides the characteristics of each stages, which are defined to the extent of being “reasonably predictable.” Gaibraith makes the point that because the model is “reasonably predictable,” it provides business managers insight into what they should expect next in their company’s growth.

Rutherford et al. (2003) adopted a four-stage model in their empirical research. Notably, in their survey of 31 research papers, they found no uniformity of stages. Their research is significant because they found no relationship between company age and stage.

Similarly, sequential-stage models based on age, or consisting of a “time” axis, are discounted by Levie & Lichtenstein (2010) who promote a Dynamic States Model (Table 3). There are two significant developments in the Dynamic States Model. First, there is the position that because market forces are dynamic, dynamic conditions will drive actions and adaptation in companies. Second, growth stages are not sequential and there may be any number of crisis encountered and any number of growth states through which the company passes, depending on a company’s response to market conditions.

Assumptions and Propositions of Stages of Growth Models and the Dynamic States Model

	Stages of Growth Models	**Dynamic States Model**
Assumption	Organizations grow as if they were organisms	Each state represents management's attempts to most efficiently/effectively match internal organizing capacity with the external market/customer demand
Propositions: WHAT	Configuration of structural variables and management problems	Configuration of structural variables and organizational activities (aspirations)
Propositions: HOW	A specific number of progressive stages	Any number of states
	Sequence and order is predictable	Sequence and order may be predictable depending on context
	Incremental and punctuated transitions	Incremental and punctuated transitions, and emergence
Propositions: WHY	Immanent program of development	Adaptive process of retaining the sustainability of a business model
	Prefigured rules of development	Interdependent rules for development
	"Regulated" by environment	Driven by market change and opportunity creation

T Table 3. Levie & Lichtenstein (2010, page 55). "A Terminal Assessment of Stages Theory: Introducing a Dynamic States Approach to Entrepreneurship"

While Levie & Lichtenstein advanced the theory that growth models must recognize the dynamic impact of external market conditions on the growth state of an organization, Levie & Lichtenstein do not state whether or not they accept that growth stages may exist between times of dynamic change. However, their acceptance that growth states may be predictable would indicate that there is qualified acceptance of growth stages between times of dynamic market change.

Hansen & Hamilton (2011) noted that not all small companies intend to become large. In fact, the objective and accomplishment of significant growth of a small company is a rarity. Most companies will attain and remain at a certain size somewhere along a growth continuum. Therefore, a growth model that is suitable for most small companies cannot impose time on a growth relationship. A company may attain a certain size and a certain state of growth at which the company will remain.

To the preceding growth model literature, concepts of capability maturity are drawn from the CMMI (CMMI, 2010), SPICE (Eshan, 2010), and OPM3 (PMI 2013, 2). The resulting combination of growth models and maturity models produce my proposed software company Growth-Stage Model.

Theory Development

The "Five Phases of Growth" of Greiner (1972), provides a suitable model structure that includes an adopted "evolutionary" growth phase followed by a "crisis" that requires a "revolutionary" response before achieving the next stage of "evolutionary" growth. However, time is not a suitable independent variable because, as advanced by Levie & Lichtenstein,stages may not be in order. Similarly, Hansen & Hamilton point out that some companies choose not to grow sufficiently to encounter stages. Different from the Greiner model, this paper's proposed Growth-Stage Model eliminates the focus on "time" and instead focuses on "company size."

The Software Company Growth-Stage Model (Figure 4) draws from aspects of existing growth models but is driven by the factors of capability maturity and company size. Applied in this model, company size is defined as the number core employees or those engaged in software development. In the model, "size" is scoped for small businesses defined as from 1 to 50 employees; including full-time-equivalents and contractors. The titles and stage definitions are intended to be intuitive and readily understood by practitioners. This model does not purport to invalidate prior conceptual growth models previously discussed. This model can be applied concurrently with other models that utilize other factors.

To respond to the fast-paced nature of the software industry, the Software Company Growth-Stage Model adopts the Dynamic States Approach of Levie and Lichtenstein (2010), which advances the position that market conditions may disrupt the progression of a sequential-stage model. The Software Company Growth-Stage Model also adopts Levie and Lichtenstein's second position that the growth stages may be repeated through growth-contraction cycles. For example, external market conditions could create a crisis that could disrupt a company's growth sequence. Similarly, an influx of venture capital could result in a significant increase of experienced employees, causing a sudden jump in stages.

Each growth model has a given advantage, varying on factors of size, time, product, or organization life cycle. As such, multiple growth models could be applied to a single company at the same time, without conflict.

There is little reason that the Dynamic States approach could not be universally applied to past growth-stage models by adding a modicum of flexibility to them. Given that that businesses operate in a dynamic market environment, it follows that business conditions may disrupt the growth-stage process, which may then force a company into a reformulated growth stage. Similarly, different divisions within a company or even multiple teams may be at different growth stages at the same time.

Software Company Growth-Stage Model

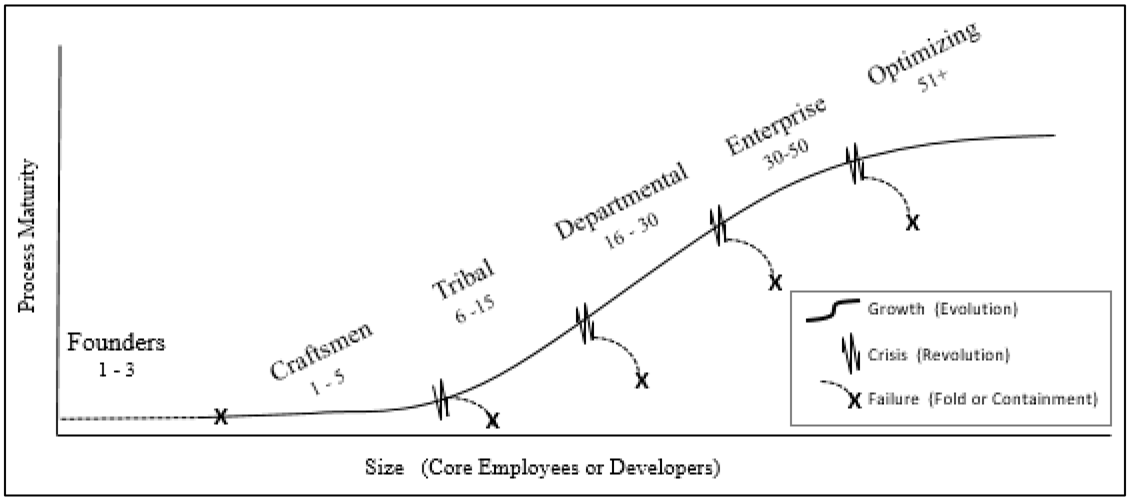


Figure 4. Software Company Growth Stage Model

The Software Company Growth-Stage Model addresses the relationship between size and capability maturity. However, there is not a specific empirical basis for grouping employee numbers into distinct stages. However, to present a possible application of stages to the numbers of employees, stages are defined based on the concept that stages are driven by characteristics of knowledge-sharing within and between groups. The size distinction would be highly fluid based on varied conditions in company. While this model is not the subject of this research, it is utilized late in the interviews during case studies to provide a relatable model for practitioners to understand that relationship between size, capability maturity, and the risk of growth-crises.

Stage Definitions of Proposed Growth-Stage Model

terms	*developers*	• *descriptions*
Founders	~1-3	• This is a planning and financing stage before the launch of business operations. • One to three founders is typical.
Craftsmen	~1-5	• At this stage, there is a small number of employees working very closely, often in the same room. Osmotic knowledge sharing is the norm, with each employee being fully aware of the work other employees are performing. • There is very little documentation at this stage. • Software coding standards are absent, relying on the professionalism of the employees.
Tribal	~6-15	• At this next stage of growth, there is an increase in work specialization. • Knowledge is still relatively easy to share among groups of two or three individuals. The organization can be effectively cross-trained to assure overlapping knowledge of all tasks. • Documentation is increasing but remains limited, and it tends to be held by individuals with little or no knowledge base indexing structure. • Software coding standards & processes are beginning to be established, but likely not entirely consistent or fully enforced.
Departmental	~16-30	• Characterizing this stage is the formalization of team responsibilities into what may be classified as formal departments. • Documentation may be collected into a growing knowledge base or files library. However, indexing is in a marginal state, with gaps in the documentation. • Software coding standards & processes are established. Gaps in standards and processes exist, and at times those standards & processes may not be fully enforced.
Enterprise	~31-50	• All important functions are documented with that documentation following established standards for completeness and understanding. • The knowledge base is fully indexed, and obsolete documents (sediment) are regularly purged. • Software coding standards & processes are established, comprehensive, and enforced--without exception. • Performance measures are quantified and reviewed.
Optimizing	~50+	• There are ongoing reviews of standards and processes for continual improvement. • Performance measures are an integral part of reviews for continual improvement. • Employee participation is integral to continual improvements.

Table 4. Stage Definitions of Proposed Growth-Stage Model

Summary

- A Growth-Stage model is presented that advocates “core employees” as the independent variable and “capability maturity” as the dependent variable, in contrast to prior models that use “time” as the independent variable.
- While highly fluid, the points of differentiation between the growth stages are determined by the number of core employees.
- It is proposed that knowledge-sharing characteristic of team or group size drive the points of differentiation of the growth stages.
- The Software Company Growth Stage Model is believed to be a conceptual model that will be readily accepted by practitioners as an aid to understanding growth stages.

CHAPTER 2.
CAPABILITY MATURITY APPRAISAL INSTRUMENT

Introduction

This research of this dissertation requires a means for a quick means to measure capability maturity within startup and small software companies. The tool draws from the CMMI Institute and the Project Management Institute to create a means for a thumbnail appraisal of capability maturity.

The Capability Maturity Appraisal Instrument assesses seventeen appraisal factors of processes and artifacts of an organization. These appraisal factors are grouped into three categories: Project Management, Software Controls, and Knowledge Sharing. Each appraisal factor is rated from 0 to 10.

An ancillary benefit of the Capability Maturity Appraisal Instrument is that it could be adopted by other researchers and practitioners when assessing capability maturity in organization. This instrument shows potential as a self-assessment tool for business managers within organizations.

Literature Review

Many software companies are facing challenges to improve quality, adapt to change, and optimally manage growth (or contraction). Performance improvements are often sought via advancements in organizational maturity (Humphrey, 1988; Jiang, Klein, Hwang, Huang, & Hung, 2004). However, most literature overlooks small and medium-sized businesses. Literature that does exists appears to assume the relevance of software process improvement applies only to large companies (Laporte, Alexandre, & O'Connor, 2008; Pino, García, & Piattini, M. 2008).

Capability maturity appraisals are available to companies at relatively high costs (Staples, Niazi, Jeffery, Abrahams, Byatt, & Murphy, 2007).[1] The Software Engineering Institute provides a self-evaluation questionnaire, but it too lacks feedback beyond an indication of a general need for increased organization maturity (Blanchette and Keeler, 2005). "Appendix B" contains a "CMMI Self-Evaluation," which displays the limited approach of CMMI questionnaires that appear designed to little more that generate consulting leads. Online, there are a few CMMI appraisals tools, each limited in scope and accessible either for a fee or through disclosure of contact information from companies selling training and consulting services.[2]

There are two widely-recognized models used to assess organizational maturity in software companies: the CMMI (Capability Maturity Model Integrated) and the ISO/IEC 15504 (Marcal, de Freitas, Soares, & Belchior, 2007). The first version of the Capability Maturity Model was published in 1993 in partnership with the Software Engineering Institute of Carnegie-Mellon University and the United States government. The initiative arose from the need to evaluate suppliers bidding for government software and internet technology contracts (Paulk, 1993). Today, the United States, along with eleven other nations, provides financial support for the CMMI Institute, which has now certified organizations in in 101 countries (CMMI Institute, 2015).

The ISO/IEC 15504 is comprised of ten "Parts" or separate technical reports that continue to be updated (Ehsan, Perwaiz, Arif, Mirza, & Ishaque, 2010).

The CMMI and the ISO/IEC 15504 are not identical, but they possess similarities in their concepts and approach to defining organization maturity and are generally considered compatible (Peldzius & Ragaisis, 2011; Pino, Baldassarre, Piattini, & Visaggio, 2010; Rout & Tuffley, 2007). The following table summarizes the maturity levels used by the CMMI and the ISO/IEC 15504. The similarities are evident in the table as the maturity levels of each model align.

Comparison of CMMI and ISO/IEC 15504 – Maturity Level Summaries

	CMMI Staged Representation	**ISO/IEC 15504**
Level 0		**Incomplete** There is a general failure to attain the purpose of the process. There are little or no easily identifiable work products or outputs of the process.
Level 1	**Initial** The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.	**Performed** Rigorously planned and tracked. There are identifiable work products for the process, and these testify to the achievement of the purpose.
Level 2	**Managed** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.	**Managed** The process delivers work products according to specified procedures and is planned and tracked. Work products conform to specified standards and requirements.
Level 3	**Defined** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.	**Established** The process is performed and managed using a defined process based upon good software engineering principles. Individual implementations of the process use approved, tailored versions of standard, documented processes to achieve the process outcomes.
Level 4	**Quantitatively Managed** Detailed measures of the software process and product quality are collected. Both the software the software process and products are quantitatively understood and controlled.	**Predictable** The defined process is performed consistently in practice within defined control limits, to achieve its defined process goals.
Level 5	**Optimizing** Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.	**Optimizing** The performance of the process is optimized to meet current and future business needs, and the process achieves repeatability in meeting its defined business goals.

Table 5. The table is compiled from separate charts from Hwang, 2009.

Both models require companies to allocate considerable costs and resources towards formal appraisals in order to obtain the necessary certifications for a higher Level of attainment. (Fayad & Laitnen, 1997; Staples, Niazi, Jeffery, Abrahams, Byatt, & Murphy, 2007).[1] The high cost of appraisal and certification is the primary reason that capability maturity appraisals are more likely to be used by large companies and are relatively uncommon in small-to-medium size software companies (Anacleto, von Wangenheim, Salviano & Savi, 2004; Khurshid, Bannerman, & Staples, 2009; Staples, & Niazi, 2008).

Of the two models, the CMMI is more prevalent in the United States. The CMMI Institute makes its capability maturity reports available at no cost, in contrast to the relatively high cost for ISO/IEC 15504 documents[3]. Additionally, the CMMI Institute has made efforts to make capability maturity appraisals more accessible to small-to-medium size software companies, in part with the release of the Continuous Representation of CMMI (CMMI Product Team, 2010). Furthermore, the CMMI SCAMPI initiative (The Standard CMMI Appraisal Method for Process Improvement) now offers a less rigorous Class C appraisal that allows for assessments without embarking on a lengthy certification process (SCAMPI Upgrade Team, 2011). For these reasons, the CMMI is utilized by the CMA-Instrument.

The CMMI Institute offers two versions of the CMMI: Staged Representation and Continuous Representation. The Staged Representation originated in the Capability Maturity Model (CMM), published in 1993 (Paulk, 1993). The Continuous Representation was first published in 2006 as part of CMMI Release 1.2 and has progressed to the current release Version 1.3 (CMMI Product Team, 2010).

Comparing CMMI Stages Representation and CMMI Continuous Representation

CMMI Staged Representation *Maturity Levels*	**CMMI Continuous Representation** *Capability Levels*
	0. Incomplete
1. Initial	1. Performed
2. Managed	2. Managed
3. Defined	3. Defined
4. Quantitatively Managed	
5. Optimizing	

Table 6. Compiled from “CMMI for Development, Version 1.3 (CMU/SEI-2010-TR-033)” (CMMI Product Team, 2010)

Because both the CMMI Staged Representation and the CMMI Continuous Representation utilize levels and similar titles, they are often shown in a single chart, as seen above in Table 6. However, it is important to note that the CMMI Staged Representation rates “Maturity Levels” while the CMMI Continuous Representation rates “Capability Levels.”

Using the Continuous Representation gives a company the flexibility to advance in specific CMMI Process Areas, without a requirement to advance all process areas equally. This allows a company to progress by identifying areas that need maturity in the early stages of advancement, which may vary depending on the company’s initial approach to adoption and advancement of processes. The following table from CMMI for Development, Version 1.3 (2010), illustrates the progression of Capability Levels 1 and 2, within Maturity Level 2; with the subsequent progression to Capability Level 3 in all Process Areas of Maturity Levels 2 and 3 to achieve Maturity Level 3.

The 22 CMMI Process Areas
Illustrating CMMI Capability Levels within Maturity Levels

Process Areas	**Staged Representation**	**Continuous Representation Capability**		
		1	**2**	**3**
Project Monitoring and Control (PMC)	2	░	░	▓
Project Planning (PP)	2	░	░	▓
Requirements Management (REQM)	2	░	░	▓
Supplier Agreement Management (SAM)	2	░	░	▓
Configuration Management (CM)	2	░	░	▓
Measurement and Analysis (MA)	2	░	░	▓
Process and Product Quality Assurance (PPQA)	2	░	░	▓
Product Integration (PI)	3	▓	▓	▓
Requirements Development (RD)	3	▓	▓	▓
Technical Solution (TS)	3	▓	▓	▓
Validation (VAL)	3	▓	▓	▓
Verification (VER)	3	▓	▓	▓
Organizational Process Definition (OPD)	3	▓	▓	▓
Organizational Process Focus (OPF)	3	▓	▓	▓
Organizational Training (OT)	3	▓	▓	▓
Integrated Project Management (IPM)	3	▓	▓	▓
Risk Management (RSKM)	3	▓	▓	▓
Decision Analysis and Resolution (DAR)	3	▓	▓	▓
Organizational Process Performance (OPP)	4			
Quantitative Project Management (QPM)	4			
Organizational Performance Management (OPM)	5			
Causal Analysis and Resolution (CAR)	5			
Legend *Capability Levels 1-2 within Maturity Level 2* *Capability Levels 1-3 leading to Maturity Level 3*				

Table 7. From "CMMI for Development, Version 1.3 (CMU/SEI-2010-TR-033)" (CMMI Product Team, 2010, p. 38)

The SCAMPI initiative (Standard CMMI Appraisal Method for Process Improvement) furthers the flexibility of the CMMI for small and mid-size companies by providing three classes of appraisals (Table 8). Of the three appraisals, the CMMI "Class A" appraisal is the only class in which the results qualify a company for a CMMI Level of Maturity certification. CMMI certification at a given Level may be required for a company to qualify to receive a government or other contract. Class B and Class C appraisals are less rigorous but provide valuable information without embarking on a lengthy and costly certification process (SCAMPI Upgrade Team, 2011). As stated in the "Appraisal Requirements for CMMI *Version 1.3",* Class C appraisals are intended for periodic maturity assessments that can be performed relatively quickly and may be suitable for self-assessment (SCAMPI Upgrade Team, 2011). For this reason, the CMMI Class C appraisals are well-suited to the needs of small to mid-size companies. "Appendix E" provides a table of the CMMI requirements for each appraisal Class (Blanchette and Keeler, 2005).

Requirements of CMMI Appraisal Method Classes

Requirements	**Class A**	**Class B**	**Class C**
Types of Objective Evidence Gathered	Artifacts and affirmations	Artifacts and affirmations	Artifacts and/or affirmations
Ratings Generated	Goal rating required	Not allowed	Not allowed
Organization Unit Coverage	Required	Not required	Not required
Appraisal Team Leader Requirements	Certified Lead Appraiser	Person trained and experienced	Person trained and experienced

Table 8. Form "Appraisal Requirements for CMMI, Version 1.3 (ARC, V1.3", SCAMPI Upgrade Team, 2011, p. 5)

It is worth mentioning that the Project Management Institute also developed a maturity model titled, "Organizational Project Management Maturity Model" or OPM3 (Project Management Institute, 2013a). OPM3 is an extension of the "Project Management Body of Knowledge" (PMBOK) of the Project Management Institute (Project Management Institute, 2013b). Worldwide, there are over 420,000 certified Project Management Professionals who are members of the Project Management Institute. Thus, it would seem reasonable to expect the OPM3 would gain increasing attention (Project Management Institute, 2014). However, the OPM3 has not been embraced by the industry, possibly due to an overly complex structure. As a result, in 2015, the Project Management Institute withdrew its support for OPM3 with a new model slated for publication in the third quarter of 2017 (Project Management Institute, 2015). While the OPM3 is not a considered in the CMA-Instrument, the PMBOK has significant recognition in the practitioner environment.

Theory Development

The Capability Maturity Appraisal Instrument (CMA-Instrument) is an assessment tool appropriate for small to mid-sized software companies. The CMA-Instrument has four key characteristics that meet the needs of smaller companies:

1) The CMA-Instrument is compatible with the CMMI;
2) Practitioners can perform maturity appraisals in one day or less;
3) Applying the CMA-Instrument does not require highly specialized training, thus it can be used as a self-assessment tool;
4) The selected maturity measures are those most likely essential to advance an organization’s maturity.

The CMA-Instrument focuses on project management, software development processes, and knowledge management. It utilizes the CMMIs Section on “Development,” focusing on Process Areas, Levels 2 and 3, but departs from the CMMI by omitting the Sections on “Acquisition” and “Service,” along with the related Processes Areas.

The CMA-Instrument deemphasizes, but does not eliminate process assessment. However, detailed process analyses are beyond the scope of a Class C appraisal. The CMA-Instrument is intended for appraisals that can be completed in one day and do not require a high degree of specialized skills. The CMA-Instrument assesses 1) artifacts (i.e. what is "work products" as described in CMMI) and 2) observable process attributes. The artifacts and attributes are key indicators of underlying processes, systems, and practices.

In a relatively fast and effective manner, the CMA-Instrument will provide key indicators of maturity status for companies operating within Maturity Levels 2 and 3. To be most relevant to small and mid-sized companies, it targets an eleven-area subset of CMMI Process Areas as seen in Table 9.

Selected CMMI Process Areas included in the Organization Maturity Appraisal Instrument of this Paper

Selected CMMI Process Area	**CMMI Abbreviations**
Project Monitoring and Control	PMC
Project Planning	PP
Requirements Management	REQM
Configuration Management	CM
Measurement and Analysis	MA
Process and Product Quality Assurance	PPQA
Product Integration	PI
Requirements Development	RD
Organizational Process Definition	OPD
Organizational Training	OT
Integrated Project Management	IPM

Table 9. Selected CMMI Process Areas Distilled from "CMMI for Development, Version 1.3 (CMU/SEI-2010-TR-033)", (CMMI Product Team, 2010, p. 33-24)

The CMA-Instrument stays consistent with the CMMI Continuous Representation by assessing Capability Levels within Process Areas (CMMI Product Team, 2010, p. 34-37). The

“CMMI for Development, Version 1.3” states “The continuous representation enables the organization to choose the focus of its process improvement efforts by choosing those process areas, or sets of interrelated process areas, that best benefit the organization and its business objectives” (CMMI Product Team, 2010, p. 31).

The CMMI establishes distinct levels for the Stage Representation (1-5), and for the Continuous Representation (0-3). The CMA-Instrument further subdivides the Levels 1 through 3 (see Table 10). These levels are the most applicable to micro and small software companies and merit the greatest scrutiny and differentiation. This rationale is similar to the CMMI Continuous, which also focuses on Levels 1 through 3. While the CMA-Instrument does include Levels 9 and 10 (level 4 and 5 of the CMMI Staged Representation), they are generally not applicable to micro and small companies.

Maturity Level of the Capability Maturity Appraisal Instrument

CMMI Stage Representation Maturity Level	CMMI Continuous Representation Capability Levels	Capability Maturity Appraisal Instrument
	0 Incomplete	0. Not Performed
1. Performed	1. Performed	1. Performed, very limited
		2. Performed, ad hoc
2. Managed	2. Managed	3.Managed, some proactive efforts
		4.Managed, partial process adoption
		5.Managed, broad process adoption
3. Defined	3. Defined	6. Defined, universal, initial
		7. Defined, universal, established
		8. Defined, universal, controlled
4. Quantitatively Managed		9. Quantitatively Managed
5. Optimizing		10. Optimizing

Table 10. Maturity Level of the Capability Maturity Appraisal Instrument

The Appraisal Factors used by the CMA-Instrument are identified in the following table (Table 10). The "A/P" column denotes whether the Appraisal Factors is an "Artifact" = "A" (something physical, including electronic, like a document) or a "Process Attribute" = "P" (an action that can be observed). The "CMMI" column references the corresponding CMMI Process Area (see Table 9), and the "PMBOK" column references the corresponding section in the Project Management Body of Knowledge (Project Management Institute, 2013a).

Capability Maturity Appraisal Instrument – Appraisal Factors

Appraisal Factors	*A/P*	*CMMI*	*PMBOK*	*Description*
Project Management				
• Change Control Log	A	CM	4.5.3	Log of requested changes to projects, with actions approved.
• Charter	A	RD	4.1	An authorization document for each project with key summary information.
• Project Schedule	A	PMC, PP	6.1-6.6	Defined schedule for project and programs that are maintained with updated status.
• Project Schedule Control Process	A	REQM	6.7	Scheduled is integrated with individual task lists, with change control and management oversight.
• Requirement Document (Traceability Matrix)	A	PI, RD	5.2 & 5.3	A document containing a list of requirements (features) from origin to deliverables, with success measures.
• Scope & Change Control Process	P	REQM	4.5, 5.5, 5.6	Defined process for proposing and approving changes to projects.
• Work Breakdown Structure	A	PI	5.4	Breakdown of requirements into feature and functions for assigning tasks.
Software Controls				
• Defect Logs	A	PPQA		Tracking of defects, and the resulting actions to the identified defects. (Defects are broadly defined.)

• Defined Coding Standards	A	OT		Written document(s) defining coding methods and representations.
• Peer Review of Code	P	OT, PPQA		Review of code assuring compliance to standards. Should be performed internal to development team before testing.
• Performance Baselines	A	PPQA REQM		Performance baselines maintained throughout systems sufficient such to allow for testing proposed releases.
• Testing & Release Segmentation	P	PPQA		Stage release process with testing at segments such to isolate major potential performance bugs.
• Verification & Validation	P	VER, VAL,		Quality control processes to assure releases meet specifications and specification meet objectives,
Knowledge Sharing				
• Documentation & Archiving (e.g. Knowledge Base)	A	OT		Standards and processes are documented.
• Integrated Knowledgebase Index	A	OT		Documentation is centrally maintained and integrated into processes throughout the company
• Training Documentation	A	OT		Documented standards and processes extended to training format
• Training Sessions	P	OT		Training session regularly held to maintain and advance standards
• Collaboration Systems	P	OT		Near real-time means gaining information from, and sharing information with others.

Table 11. Artifacts & Process Attributes of the Organization Maturity Model. Draw from “CMMI for Development, Version 1.3” (CMMI Product Team, 2010, 89-93) and drawn from the “Project Management Body of Knowledge” (Project Management Institute, 2013a)

Summary

- The Capability Maturity Appraisal Instrument conforms to a CMMI, Class C Appraisal, with the qualification that CMMI Levels/Representation 1 through 3 are further subdivided within the CMA Instrument.
- It is believed that the CMA Instrument appraisal measurements produce reasonably accurate measures of capability measures.
- The CMA Instrument should show potential to produce useful results as a self-assessment tool for practitioners to assess their own companies.

CHAPTER 3.

COMPLEXITY: MODERATORS OF MATURITY REQUIREMENTS

Introduction

In this chapter, the complexities of organizations and operating systems are investigated in relation to capability maturity. As complexities are imposed on an organization, the required level of capability maturity increases. In this chapter, a list of identified complexity factors are provided, plus software complexity, which can have a significant effect on software companies, is addressed in detail. The Software Complexity Model is proposed as a means to categorize software platform complexity and as a vehicle to present this concept to practitioners.

Literature Review

Software Complexity Measures

The most noted early software complexity measures were created by Halstead (1977) and McCabe (1976), both of whom advanced the use of quantitative metrics in their research. Halstead introduced metrics that assess the complexity of the source code starting with "Program Volume," which he defined as a function of distinct operators and operands, combined with the total number of operators and operands. Halstead further assessed software complexity, building from Program Volume to include equations for Difficulty, Effort, Time (to program), and Bugs.

McCabe (1976) coined the term "Cyclomatic Complexity," which defined software complexity based on the number of linear logic/processing paths within a software's source code. Cyclomatic Complexity is a more accurate measure of complexity than early methods of simply counting lines of code. Counting lines of code defined volume, but may not correlate to complexity.

Software Modularity and Information Flow

The modularity of software architecture was advanced as a response to software complexity. Parnas (1972) promoted the benefits of modularization as 1) shortened development time; 2) improved flexibility from the ability to improve one model without involving other modules; and 3) comprehensibility from the ability to study one module at a time. As an early example of modularization, McClure (1978) noted that IBM released a technical brief stating that code should be divided into modules, with no module consisting of more than 50 lines of code. While McClure found the limitation of 50 lines highly arbitrary, he noted that modularity was a major consideration in architecting software.

McClure (1978) also suggested that complexity measures must include factors of modularity stating that, "McCabe's methods cannot be a complete indicator of complexity." Complexity is, to a significant degree, driven by the interrelationship of the modules as determined by variables of the path structures for the process logic. The concept of software architecture as a major determinant in software complexity was significant.

Consistent with McClure, Henry and Kufura (1981) advanced the concept that complexity in software structure resulted from information flow between subsystems or modules. They performed a broad analysis of large-scale UNIX system and found a strong correlation between the information flow between system layers and the measures of complexity. This validated their hypothesis that the architecture of a system is a dominant factor in complexity, which can be determined in advance of development.

Software Maintainability

The concepts of complexity measures and architectural/modularity-driven complexity lead to the related concepts of software maintenance and decay. Harrison, Magel, Kluczny, and DeKock, (1982) presented aspects of software complexity pertaining to maintaining software systems over time: software maintainability is primarily driven by “how difficult the program is to comprehend and work with,” for which existing measures at that time were inadequate.

Basil & Perricone (1984) stated that there are three measures of maintenance complexity; 1) software understandability, 2) software modifiability, and 3) software testability. They performed an error analysis and found

1. “modified modules appeared to be more susceptible to errors”;
2. “errors contained in modified modules were found to require more effort to correct”; and
3. “if better specifications could be developed, it might reduce the more expensive errors.”

Furthermore, Basil & Perricone state that “these are the tradeoffs between modifying an existing module as opposed to creating a new one.” The inference of their statement is that depending on inherent system architectural/modular complexity, and subject to the quality of specifications, there is a breakeven point at which it is more cost-effective to create a new software system rather than maintain an increasingly costly old system.

Parnas (1994, p. 279) stated it most clearly: “Programs, like people, get old.” Parnas states that are two type software aging. First, there is aging caused by failure to modify the software to make necessary changes. Second, there is aging caused by modifications to the software. Therefore, software aging is unavoidable.

Banker, Davis, & Slaughter (1998) extend theory on software maintainability as well as software complexity measures. They present three variables for software complexity (see Figure 5):

1. “Software Component Complexity: a greater number of data elements per line of code.”
2. “Software Coordinative Complexity: ‘decision density’ where there is multiple decision branching that obscures the relationships between inputs and outputs of dispersed code segments.”
3. “Software Dynamic Complexity: an example is code logic ‘decision volatility’ where multiple process flow paths could invoke multiple and variable executes.”

In their conclusions, Banker et al. (1998) emphasize a need for greater consideration of maintenance life-cycle costs “in the adoption of design and development practices.” Research by Dedrick, Gurbaxani, & Kraemer (2003) found that 70% to 80% of information technology budgets are consumed just to keep existing software systems running.

Software Complexity Variable: Component, Coordinative, & Dynamic Complexity

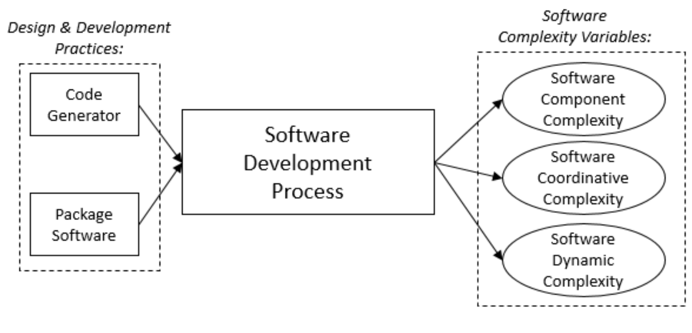

Design & Development Practices:
Code Generator
Package Software
Software Development Process
Software Complexity Variables:
Software Component Complexity
Software Coordinative Complexity
Software Dynamic Complexity

Figure 5. Banker, Davis, & Slaughter, (1998). Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study. (p. 438)

The topic of code decay was studied by Eick, Graves, Karr, Marron & Mockus (2001) who analyzed of 15 years of accumulated data. They found that code decays based on 1) the number files touched/changed, 2) the number modules touched /changed, 3) the frequency/recency of change, and 4) the span and size of the changes by feature.

In their conclusions, Eick et al. (2001) state, "We anticipate that all projects of sufficiently large scale will exhibit decay to some extent: that is, code decay is a generic phenomenon." In response to decay, Brown, Cai, Guo, Kazman, Kim, Kruchten, & Zazworka (2010) advocate scholars address "technical debt" to avoid an unaltered path of software decay, with the resulting escalation of costs. One strategy Brown et al. (2010) advances is that of proactive "refactoring." Refactoring is the restructuring, replacement, and updating of an existing body of code, without changing its external behavior. However, Brown et al. (2010) also stated that "refactoring efforts cannot compensate for the lack of a coherent system-wide architecture."

Theory Development

Software Platform Complexity Model

The Software Complexity Model generalizes the majority of software into three categories: 1) Individual Projects, 2) Componentized Products, and 3) Integrated Platforms. This model is titled "Software Complexity Model."

Software Platform Complexity Model

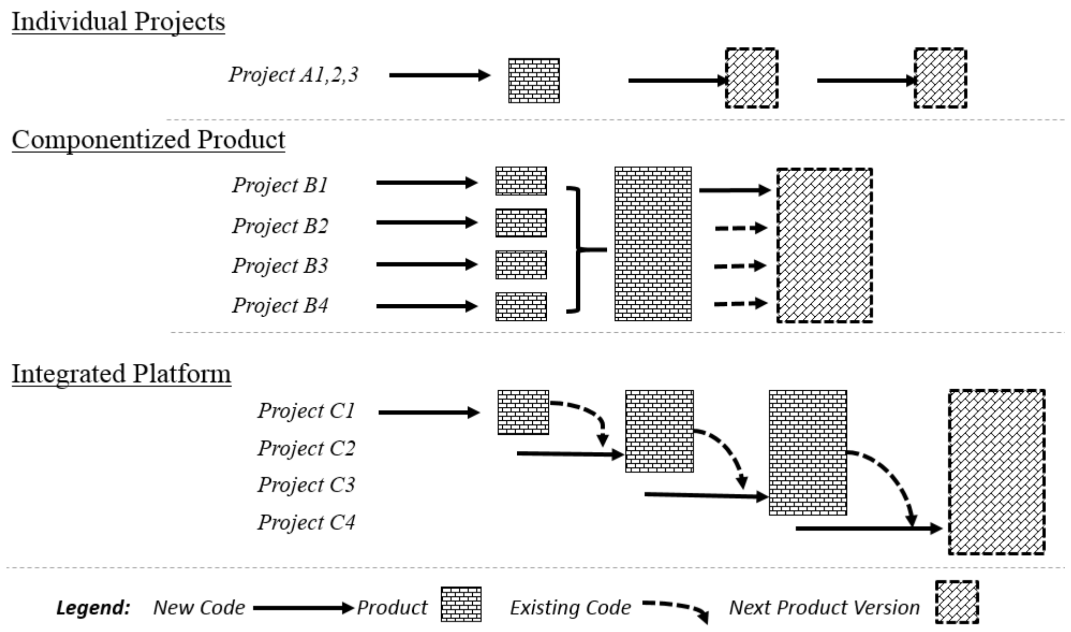


Figure 6. Software Platform Complexity Model

Figure 6 illustrates the conceptual nature of development within each category. Companies whose project portfolio is limited to stand-alone Individual Projects could produce software projects with considerably fewer constraints of organizational maturity, Companies who develop and maintain Componentized Products have greater maturity requirements than a company engaged solely in Individual Projects. Those companies that create and maintain Integrated Platforms have the greatest maturity requirements.

Complexity Factors

The following is a list of complexity factors (Table 12) that may increase the maturity requirements of software companies. The list is a compilation of notes from field research, which may provide a topic for future research in complexity factors that will affect requirements for organization maturity in software companies.

Other Complexity Factors

Engineering	
Hardware	Inherent complexity or sophistication of the hardware management requirements
Software	Inherent complexity or sophistication of the software language requirements
Internal Complexity Factors	
Outsourcing Software Development	Outsourcing likely requires a higher degree of documentation and communication sophistication.
Financial & Personal Data	Handling of financial or personal data entails liability and requires heightened security, both from external intrusions and well as internal controls.
Staffing Maturity	Less experienced engineers and developers will likely require additional documentation, defined processes & practices, and increased oversight
Contractors	
External Complexity Factors	
Government	Government entities may have requirements for security, documentation, process controls, project management processes, etc.
Public Stock	When providing services to companies that are publicly traded, there are likely to be restrictions on update releases to assure that there are no appearances of stock price manipulation. There may be additional records requirements that result from oversight.
Large Clients	Large clients may impose requirements that meet their system needs, plus large clients may have specific expectations that must be met.
Financial Institution	Financial institutions may have security requirements, data integrity requirements, and performance expectations.

Table 12. Complexity Factors Compiled from Literature and Research

Summary

- The Software Complexity Model groups software platforms into three categories for ease of analysis and for understanding of software complexity by practitioners.
- Complexity of software platforms is a moderator of capability maturity requirements that explain variability in capability measure in companies that are the same size.
- A list of other complexity factors is provided as additional moderator of capability maturity requirements.

CHAPTER 4.

OPTIMAL MATURITY MODEL

Introduction

The proposed theory of optimal maturity states that there is a relationship between company size and the required level of capability maturity. The required level of capability maturity can be generally determined for each company, moderated by the complexities in a company.

It must be noted that there is a distinction between “optimized” maturity and “optimal” maturity. Optimized or optimizing is the highest level of capability maturity. In contrast, optimal maturity is the adequate of minimally sufficient level of capability maturity for a company.

As a diagnostic tool, the proposed Optimal Maturity Model would be company would rated ahead or behind the optimal maturity curve (average capability maturity for all companies). Are the company’s operations deficient, thus prone to failure? Are the company's resources being consumed for excessive processes and procedures? Is a company a worthy investment, or are its operations insufficient for growth?

As a guide to business managers, the Optimal Maturity Model would indicate whether an organization’s capability maturity ranks ahead or behind the norm for companies of its size. With that information business manager could address potential capability inefficiencies if the measures are high, or capability deficits if the measures are low. When facing growth, business managers could anticipate and implement needed operational advancements prior to hiring more employees and encountering growing pains.

.

Literature Review

There is considerable literature on achieving optimized capability maturity. What is absent is literature that advocates achieving only the level of maturity necessary for an individual company's stage of development.

A minimal level of maturity is similar to the concept of the Minimally Viable Product presented in The Lean Start-Up (Ries, 2011). The key is to only implement a level of maturity that is required for the company's stage of development. To borrow from Ries, the concept of maturity advancement could be labeled Minimally Viable Maturity.

Similarly to Lean Start-Up, the Optimal Maturity Model draws from Toyota's concepts of Just-In-Time inventory control (Sugimori, Kusunoki, & Uchikawa, 1977; Womack, Jones, & Roos, 1990; Holweg, 2007). By adapting the concepts and terminology, Just-In-Time Maturity would implement improvements in an organization's maturity just in time for each step in growth and complexity. An organization would not consume resources to improve operational maturity prior to such improvements being required to meet an organization's functional needs.

Finally, the Optimal Maturity Model draws on the software development models of RAD, Agile, and Lean, all derived from foundational Toyota production concepts (Martin, 1991; Fowler & Highsmith, 2001; Poppendieck, 2007), which emphasize incremental and adaptive development. Applying these concepts to optimal capability maturity, incremental improvements in maturity should occur as needed, or in anticipation of need, and the organization should adapt the maturity systems as needs change.

Theory Development

There is a presumption that there are economies of scale when implementing maturity systems within an organization. Economies of scale make intuitive sense and are substantiated by statistics from the CMMI Institute: the majority of companies gaining CMMI maturity certification are larger than 50 employees and many of those under 50 employees are divisions of large companies (CMMI Institute, 2015). However, while economies of scale may exist and benefit large companies, this paper asserts that there is a diseconomy of scale for small companies (i.e. micro companies of 0 to no more than 20 employees) implementing capability maturity systems. Figure 7, below, illustrates the economies of scale and diseconomies of scale.

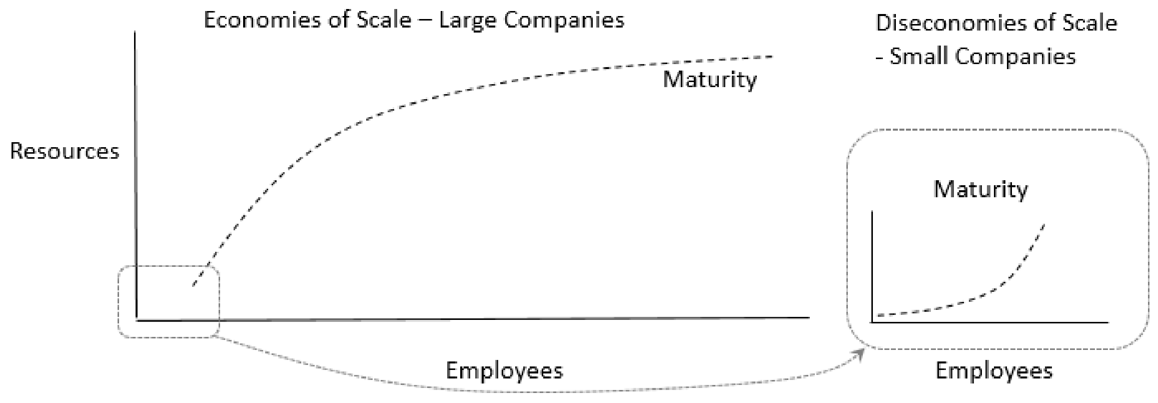


Figure 7. Economies and Diseconomies of Scale

The rationale for the concept of diseconomies of scale is founded in team dynamics and knowledge sharing. Small teams can more easily share information than can a large team. A small team can more easily locate members more closely together than can a large team. A small team requires few defined policies, procedures, and systems; as larger teams grow, they require more detailed systems, which increases the risk of bureaucratic inefficiencies. Therefore, the combined maturity curve for resources per employee is an S-curve that recognizes both the advantages of small companies and the advantages of large companies (see Figure 8).

Capability Maturity S-Curve

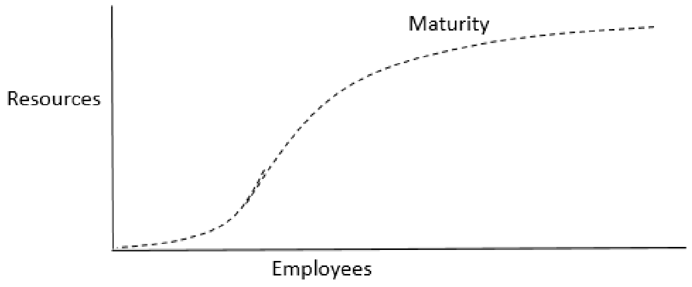


Figure 8. Conceptual Optimal Capability Maturity S-Curve

It is postulated that if the Complexity Factors are excluded, organizational maturity observed within companies will follow the illustrated S-curve in Figure 20. As a company grows and transitions from requiring a low level of capability maturity to a high level of maturity, it will likely encounter a steep increase in resource requirements. These demands could trigger a growth crisis, similar to crises of the growth-stage models introduced by Greiner (1972) and Scott & Bruce (1987), presented in Chapter 3.

The Optimal Maturity Model theorizes that assessed maturity levels in companies, excluding complexity factors, will follow an S-curve. The Model presents the independent variable of “size,” determining the dependent variable of “maturity.” Both variables are not simply conceptual but measured. If this expectation holds in empirical research, then the Optimal Maturity Model would provide practitioners a valuable guide to their own company’s maturity requirements. It is expected that there is far too much variability between companies for the guide to be prescriptive. However, this could be an invaluable guide for companies to identify potential challenges associated with growth.

Optimal Maturity Model

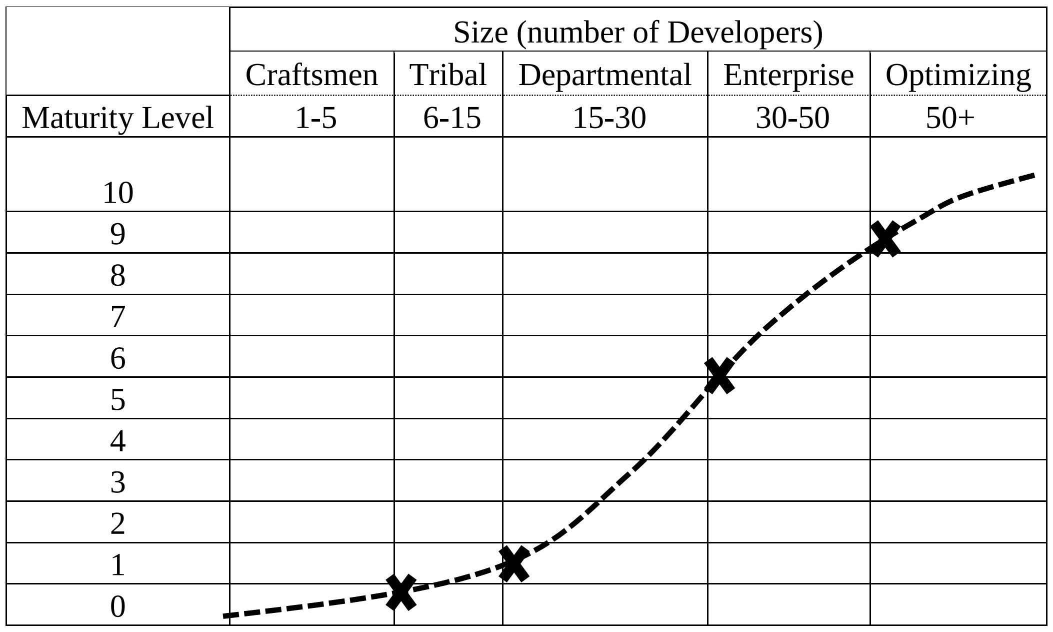


Figure 9. Conceptual Optimal Maturity Model Combines; the S-Curve, CMA-Instrument, Growth-Stage Model's Size Groupings, and Growth-Stage Crisis Points

The Optimal Maturity Model utilizes grouping of core employees (Figure 9) as presented in the Growth Stage Model in Chapter 1. The Model is not dependent on these groupings. These groupings are included to facilitate the presentation of the chart, at the same time noting the increasing scale of developers (N + 5x) with each step could be a future investigation. The use of the groups also is used to identify potential points of crises as introduced by by Greiner (1972) and Scott and Bruce (1987). While the Optimal Maturity Model visually implies a sequential progression of growth, it should be noted that growth is not always sequential but can take large steps forward or backward (Levie & Lichtenstein, 2010).

Summary

- The Optimal Maturity Model is presented, contending that there is a minimum capability maturity required for a given size company. Thus, a distinction is drawn between “optimal” (i.e. sufficient) and “optimizing” (i.e. highest level of capability maturity).
- The collective required capability maturity of Optimal Capability Maturity forms an S-curve, with micro startup companies able to differ capability maturity until growing to a slightly larger size)
- The Optimal Maturity Model can be used as a diagnostic tool to aid in identifying if capability maturity for a given company is above or below the optimal level for a company’s size and conditions. At the same time, there would be an indication whether a company is sufficiently capable of taking on growth.
- The Optimal Maturity Model can be used as a guide to aid a company in determining the required capability maturity for anticipated growth and changes in business conditions.

CHAPTER 5. METHODOLOGY & DATA

Research Methodology

Structure.

The research-design structure is adapted from Case Study Research: Design and Methods by Yin (2013). In particular, Yin's "Components of Research Design" provides the desired research structure (Yin, 2013, p. 29).

Semi-structured interviews are performed that adopt the "Responsive Interviewing Model" (Rubin & Rubin, 2011, p. 96). The responsive interviewing model allows for the desired flexibility in exploring issues with the subjects, and at the same time utilizing an interview mythology that assists in obtaining valid results (i.e. results less apt to be misinterpreted or infused with bias).

Interviews

The research relies on foundational literature for theory development. Theory development is then explored through mini case studies. The mini case studies are derived from semi-structured interviews performed with twenty small software companies located across the United States.

The subject companies range in size from one to forty-two core employees. (Refer to Figure 22.) "Core Employees" are those employees engaged in product development. Core employees specifically excludes those limited solely to such roles as accounting, customer services, and sales. However, within small companies it is common for involvement and input into product development to be shared among diverse roles, thus the count of core employees may include those engaged in sales and service roles.

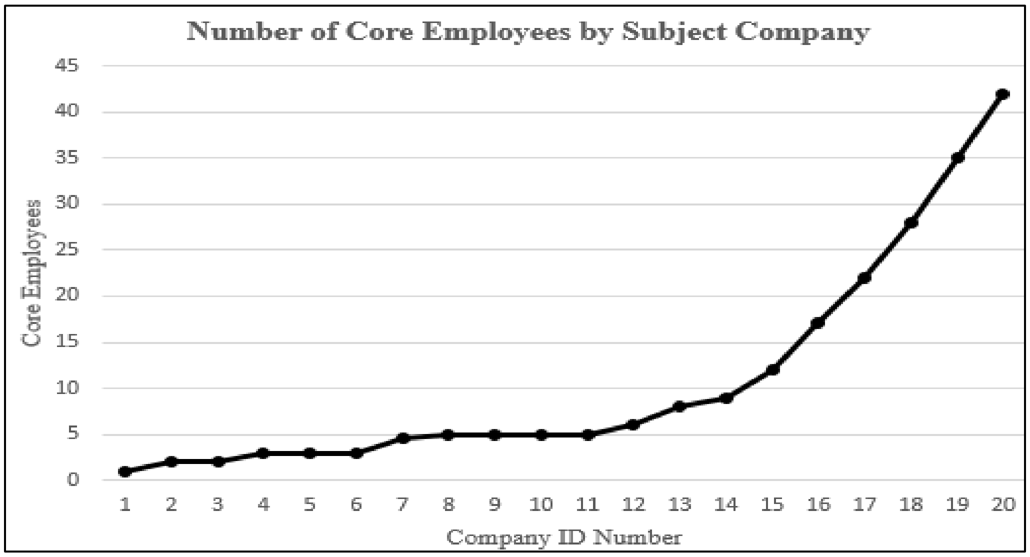


Figure 10. Graph of the Count of Core Employees of Subject Companies

The years in business of seventeen of the subject companies is three years or less. Three subject companies are between eight and fifteen year in business. The three older companies inject the possibility of diversity in results, and at the same time provide a check on the premise that size drives capability maturity, not age. (Refer to Figure 11.)

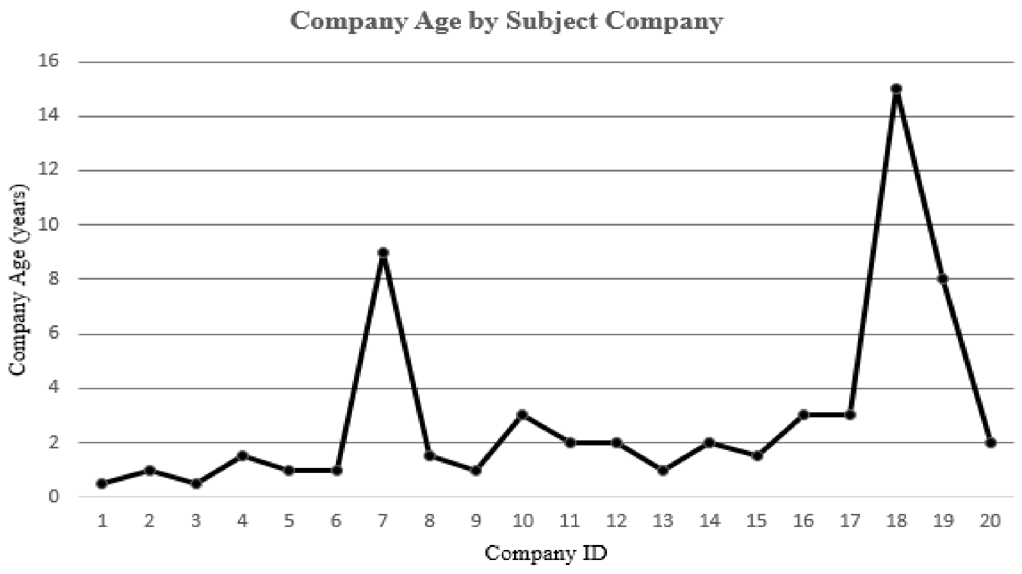


Figure 11. Graph of Subject Companies' Years in Business

Follow-up calls and email were utilized for most companies to confirm information and interpretations. The companies interviewed early in the research required some follow-up to complete questions that were added later in the research. With six companies it was possible to perform onsite interview with more in-depth assessments of capability maturity. A longitudinal perspective was possible with five companies that were interviewed early in the research and then re-interviewed ten-to-twelve months later.

Founders were sought for the interviews because they possessed the greatest knowledge of their companies' history, strategy, and operations. Interviews were performed with founders of the subject companies in all but one case. The founder of one company felt he was no longer intimately involved in operations, thus referred me to interview a long-time senior engineer.

Semi-Structured - Questions Part 1

Interviews were segmented into two parts. The first part was comprised of questions to gain information on each company's background, operating conditions, software platform, expected growth or change, and to gain insight into complexity factors. Each initial question typically lead to one or more follow-up questions.

Semi-Structured - Questions Part 2

The second part of the interviews pertained to completing the Capability Maturity Appraisal, described in Chapter 2. The structure of questions poised for each Appraisal Factor followed an approach of asking the questions, "Do you have" (for artifacts) or "Do you do" (for processes). For example, the question would be asked, "Do you have a Change Control Log?" The subject's answer would then be followed up with a probing open-ended question.

While the appraisal itself is structured, the answers received were highly variable. It was not unusual for a subject to require additional information about one or more survey questions. For example, the “Work Breakdown Structure” is a term common in waterfall project management, but the term is not typically used in Agile software development. Therefore, to accurately determine the capability maturity level of each Appraisal Factor, several questions may be asked.

Between subjects, the survey questions for which a subject would require additional information were highly variable. It was a risk that any question asked could be misunderstood. As a result of subjects’ unfamiliarity of terms, often an open discussion about the question topic preceded a question’s answer. There is a risk that further descriptions of the questions produced altered results. However, often the explorations surrounding explanations produced a richness in detail on the topic.

Complexity Interviews

Separate interviews were undertaken to explore complexity factors. As the interviews with small software companies progressed, the lack of complexity, or at least the lack of concern for complexity by most companies did not match expectations. Rather than disregard the concept of complexity factors as moderators of required capability maturity, effort was undertaken to explore the topic in a greater depth.

Five additional interviews were performed with well-established companies. The premise of undertaking these additional interviews is that complexity in younger startup companies may be overlooked for a time. However in time, companies must respond to complexity with higher capability maturity levels if they are to sustain their operations. The five companies were comprised of three large software companies (>1,000), one mid-size company (>500 employee) and the IT staff managing a 911 emergency system.

The Data

Company names have been changed for confidentiality. The aliases created are the names of colors, each drawn from a different letter of the alphabet, applied to the subject companies progressing alphabetically from small to larger companies.

The purpose of the first segment of the semi-structured interviews was to obtain general information about the company and its people. The following table presents the topics of information captured in each case, along with brief notes regarding the data collection of each topic.

Data Description of Interview Segment Part 1

Data Field	**Data Description**
Core Employees	Counting those involved in product development or service offering. This includes more than just engineers, but specially excludes accounting, sales, and customer service personnel.
Years in Business	Rounded to whole numbers except in the first two years in which rounding is to half years.
Founders' Experience	This field highlights “Experienced” founders who have prior experience with startups, along with at least a Masters level of education. Other founders are listed as “Limited” experience.
Profitable	“Yes” or “No” is obtained.
Funding	The categories are; Bootstrapping, Seed, Series A, Series B, or Profit-Sustaining.

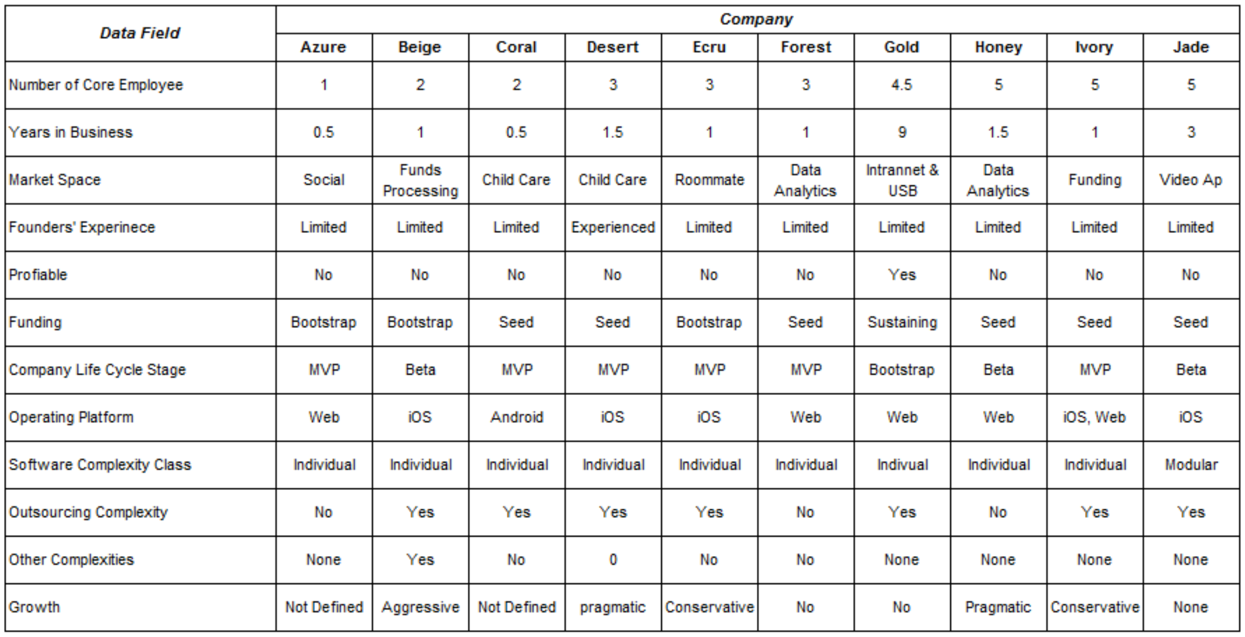
Data Description of Interview Segment Part 1 (Continued)

Data Field	**Data Description**
Product Life Cycle Stage	Using a four category model (Introduction, Growth, Maturity, decline), with “Introduction further subdivided into Beta, MVP, and Full-Featured.
Operating Platform	Web, Mobile (web), iOS, Android, and USB
Software Complexity Class	Defined in the Software Complexity Model in Chapter 3.
Outsourcing Complexity	The question is whether all or part of software engineering is outsourced.
Other Complexities	Notes of other types of complexity

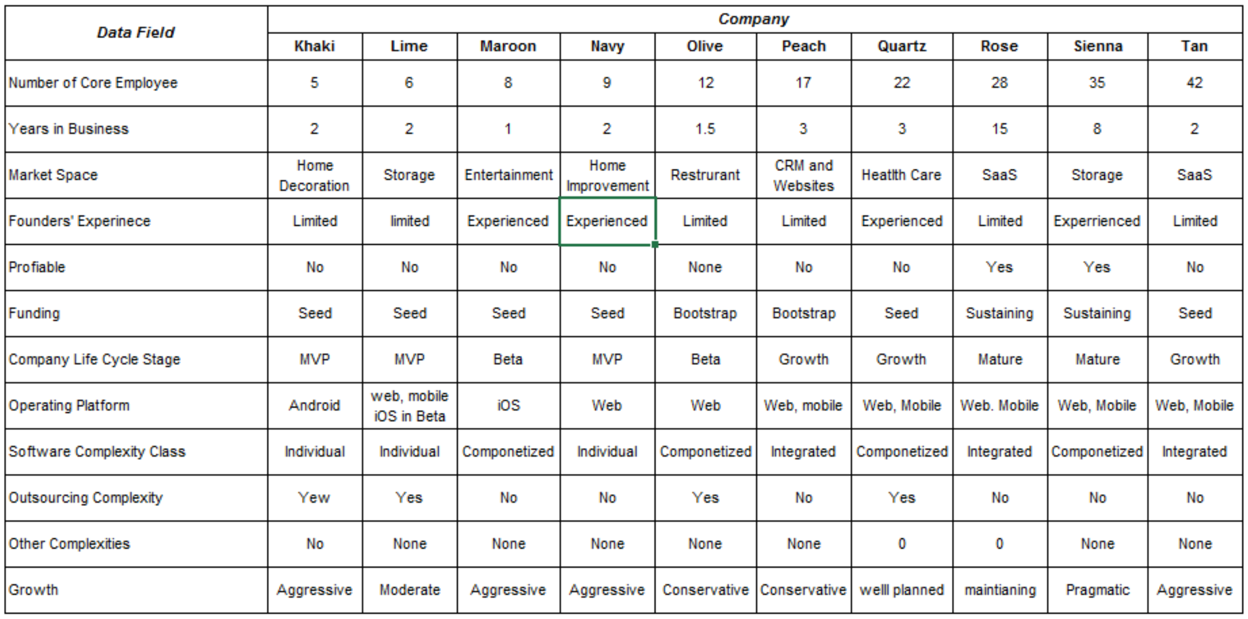
Table 13. Data Description of Part 1 of Semi-Structured Interviews

The general information collected on each subject company during the research interviews is presented in Table 14.

General Information - Each Subject Company



Data Field	***Company***									
	Azure	**Beige**	**Coral**	**Desert**	**Ecru**	**Forest**	**Gold**	**Honey**	**Ivory**	**Jade**
Number of Core Employee	1	2	2	3	3	3	4.5	5	5	5
Years in Business	0.5	1	0.5	1.5	1	1	9	1.5	1	3
Market Space	Social	Funds Processing	Child Care	Child Care	Roommate	Data Analytics	Intrannet & USB	Data Analytics	Funding	Video Ap
Founders' Experinece	Limited	Limited	Limited	Experienced	Limited	Limited	Limited	Limited	Limited	Limited
Profiable	No	No	No	No	No	No	Yes	No	No	No
Funding	Bootstrap	Bootstrap	Seed	Seed	Bootstrap	Seed	Sustaining	Seed	Seed	Seed
Company Life Cycle Stage	MVP	Beta	MVP	MVP	MVP	MVP	Bootstrap	Beta	MVP	Beta
Operating Platform	Web	iOS	Android	iOS	iOS	Web	Web	Web	iOS, Web	iOS
Software Complexity Class	Individual	Individual	Individual	Individual	Individual	Individual	Indivual	Individual	Individual	Modular
Outsourcing Complexity	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes
Other Complexities	None	Yes	No	0	No	No	None	None	None	None
Growth	Not Defined	Aggressive	Not Defined	pragmatic	Conservative	No	No	Pragmatic	Conservative	None



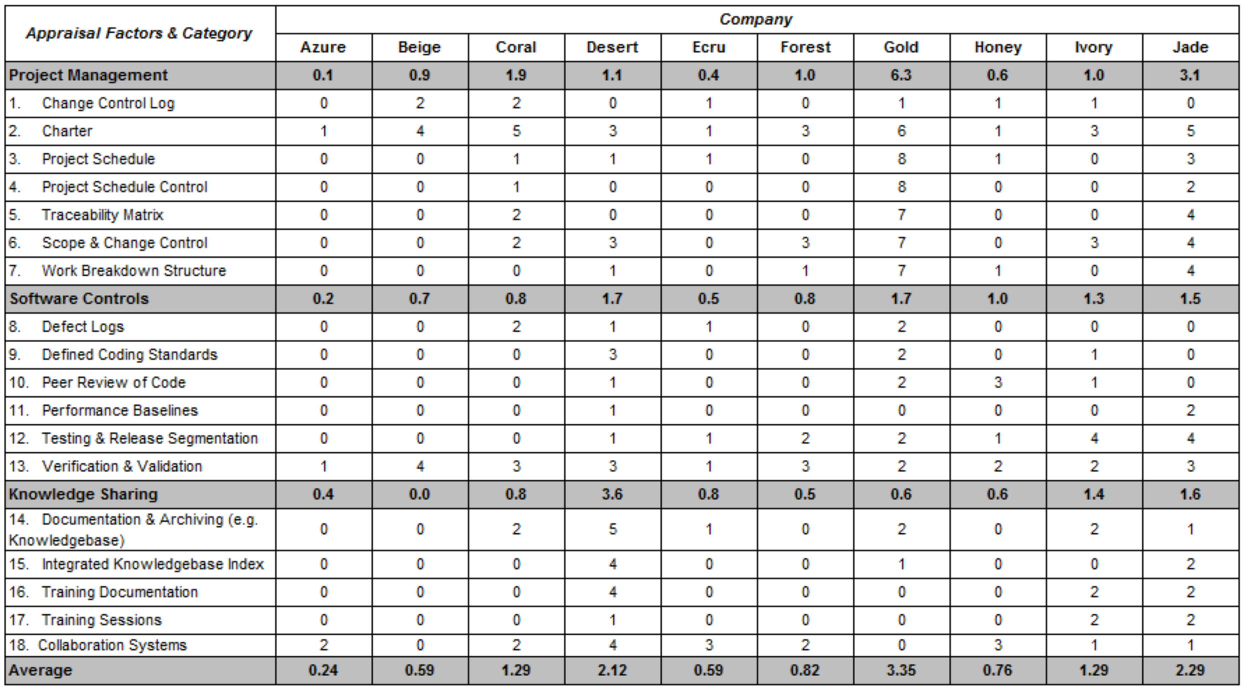
Data Field	***Company***									
	Khaki	**Lime**	**Maroon**	**Navy**	**Olive**	**Peach**	**Quartz**	**Rose**	**Sienna**	**Tan**
Number of Core Employee	5	6	8	9	12	17	22	28	35	42
Years in Business	2	2	1	2	1.5	3	3	15	8	2
Market Space	Home Decoration	Storage	Entertainment	Home Improvement	Restrurant	CRM and Websites	Heatlth Care	SaaS	Storage	SaaS
Founders' Experinece	Limited	limited	Experienced	Experienced	Limited	Limited	Experienced	Limited	Experrienced	Limited
Profiable	No	No	No	No	None	No	No	Yes	Yes	No
Funding	Seed	Seed	Seed	Seed	Bootstrap	Bootstrap	Seed	Sustaining	Sustaining	Seed
Company Life Cycle Stage	MVP	MVP	Beta	MVP	Beta	Growth	Growth	Mature	Mature	Growth
Operating Platform	Android	web, mobile iOS in Beta	iOS	Web	Web	Web, mobile	Web, Mobile	Web. Mobile	Web, Mobile	Web, Mobile
Software Complexity Class	Individual	Individual	Componetized	Individual	Componetized	Integrated	Componetized	Integrated	Componetized	Integrated
Outsourcing Complexity	Yew	Yes	No	No	Yes	No	Yes	No	No	No
Other Complexities	No	None	None	None	None	None	0	0	None	None
Growth	Aggressive	Moderate	Aggressive	Aggressive	Conservative	Conservative	welll planned	maintianing	Pragmatic	Aggressive

Table 14. General Information Collected on Each Subject Company

In the second segment of the semi-structured interviews, capability maturity is appraisal using the CMA-Instrument (Chapter 2). The results of the appraisal are shown in Table 15.

Complexity Factors (Chapter 3) are explored when their existence appears possible. In additional to the Complexity Factors, other moderators of the appraiser results were actively sought when the capability maturity appeared outside the typical results that might be indicated in the Optimal Maturity Model.

Measured Capability Maturity (Companies 1 through 10)



Appraisal Factors & Category	*Company*									
	Azure	**Beige**	**Coral**	**Desert**	**Ecru**	**Forest**	**Gold**	**Honey**	**Ivory**	**Jade**
Project Management	**0.1**	**0.9**	**1.9**	**1.1**	**0.4**	**1.0**	**6.3**	**0.6**	**1.0**	**3.1**
1. Change Control Log	0	2	2	0	1	0	1	1	1	0
2. Charter	1	4	5	3	1	3	6	1	3	5
3. Project Schedule	0	0	1	1	1	0	8	1	0	3
4. Project Schedule Control	0	0	1	0	0	0	8	0	0	2
5. Traceability Matrix	0	0	2	0	0	0	7	0	0	4
6. Scope & Change Control	0	0	2	3	0	3	7	0	3	4
7. Work Breakdown Structure	0	0	0	1	0	1	7	1	0	4
Software Controls	**0.2**	**0.7**	**0.8**	**1.7**	**0.5**	**0.8**	**1.7**	**1.0**	**1.3**	**1.5**
8. Defect Logs	0	0	2	1	1	0	2	0	0	0
9. Defined Coding Standards	0	0	0	3	0	0	2	0	1	0
10. Peer Review of Code	0	0	0	1	0	0	2	3	1	0
11. Performance Baselines	0	0	0	1	0	0	0	0	0	2
12. Testing & Release Segmentation	0	0	0	1	1	2	2	1	4	4
13. Verification & Validation	1	4	3	3	1	3	2	2	2	3
Knowledge Sharing	**0.4**	**0.0**	**0.8**	**3.6**	**0.8**	**0.5**	**0.6**	**0.6**	**1.4**	**1.6**
14. Documentation & Archiving (e.g. Knowledgebase)	0	0	2	5	1	0	2	0	2	1
15. Integrated Knowledgebase Index	0	0	0	4	0	0	1	0	0	2
16. Training Documentation	0	0	0	4	0	0	0	0	2	2
17. Training Sessions	0	0	0	1	0	0	0	0	2	2
18. Collaboration Systems	2	0	2	4	3	2	0	3	1	1
Average	**0.24**	**0.59**	**1.29**	**2.12**	**0.59**	**0.82**	**3.35**	**0.76**	**1.29**	**2.29**

Measured Capability Maturity (Continued, Companies 11 through 20)

Appraisal Factors & Category	*Company*									
	Khaki	**Lime**	**Maroon**	**Navy**	**Olive**	**Peach**	**Quartz**	**Rose**	**Sienna**	**Tan**
Project Management	**2.7**	**1.0**	**3.6**	**2.9**	**3.1**	**5.4**	**8.1**	**5.1**	**6.6**	**6.9**
1. Change Control Log	3	0	3	1	0	4	6	6	7	8
2. Charter	4	2	5	4	5	6	9	4	7	8
3. Project Schedule	2	1	5	3	3	6	7	7	6	6
4. Project Schedule Control	3	0	5	4	2	6	8	4	5	5
5. Traceability Matrix	2	0	2	2	4	3	9	5	6	8
6. Scope & Change Control	3	3	3	2	4	7	9	3	7	5
7. Work Breakdown Structure	2	1	2	4	4	6	9	7	8	8
Software Controls	**2.0**	**1.8**	**2.2**	**2.2**	**1.5**	**5.2**	**7.2**	**5.8**	**7.0**	**7.7**
8. Defect Logs	3	3	2	1	0	5	7	7	7	7
9. Defined Coding Standards	1	0	1	0	0	4	8	3	7	7
10. Peer Review of Code	2	0	1	0	0	5	8	6	7	7
11. Performance Baselines	0	3	0	5	2	5	5	7	7	9
12. Testing & Release Segmentation	2	3	4	4	4	6	7	7	7	9
13. Verification & Validation	4	2	5	3	3	6	8	5	7	7
Knowledge Sharing	**1.6**	**3.2**	**2.6**	**0.6**	**1.6**	**4.0**	**7.0**	**3.0**	**7.2**	**6.8**
14. Documentation & Archiving (e.g. Knowledgebase)	3	4	5	2	1	5	8	4	9	8
15. Integrated Knowledgebase Index	2	3	4	0	2	3	7	4	9	8
16. Training Documentation	0	4	0	0	2	5	6	1	8	5
17. Training Sessions	0	0	0	0	2	4	5	1	6	6
18. Collaboration Systems	3	5	4	1	1	3	9	5	4	7
Average	**2.29**	**2.00**	**3.00**	**2.12**	**2.29**	**5.24**	**7.94**	**5.06**	**7.29**	**7.53**

Table 15. Measured Capability Maturity via the CMA-Instrument

CHAPTER 6. ANALYSIS AND FINDINGS

Software Company Growth-Stage Model (Chapter 1)

Correlation of Capability Maturity to Core Employees

A statistical analysis of the 360 data points using Pearson's Correlation shows a correlation of the relationship to be 0.7352. The correlation indicates a strong positive relationship between core employees and capability maturity. This relationship is presented in the following chart showing the positive relationship of the average capability maturity for each subject company.

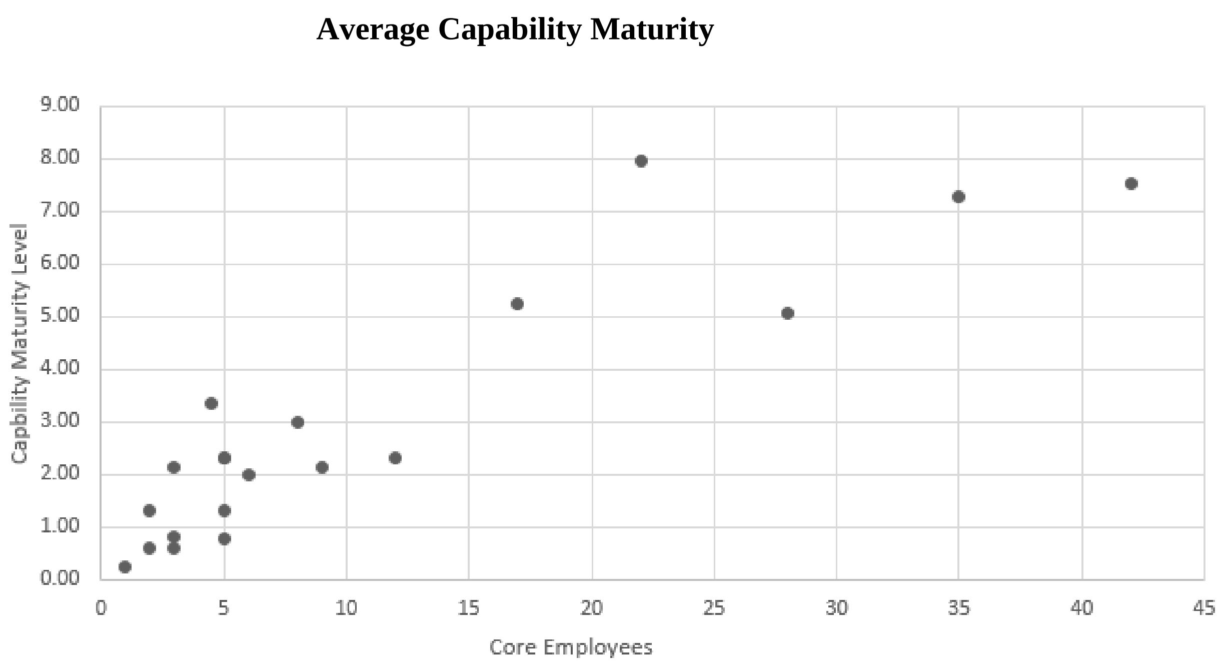


Figure 12. Average Capability Maturity Appraised via CMA-Instrument

Growth-Stages

As illustrated in Figure 13, there is preliminary evidence of distinct growth stages. As noted in this paper's section on Growth Stage Theory Development, growth stages are highly fluid, varying according to the conditions within individual companies. In this chart below, once five outliers are filtered, the capability maturity groupings match the transition points of the proposed Growth-Stage Model.

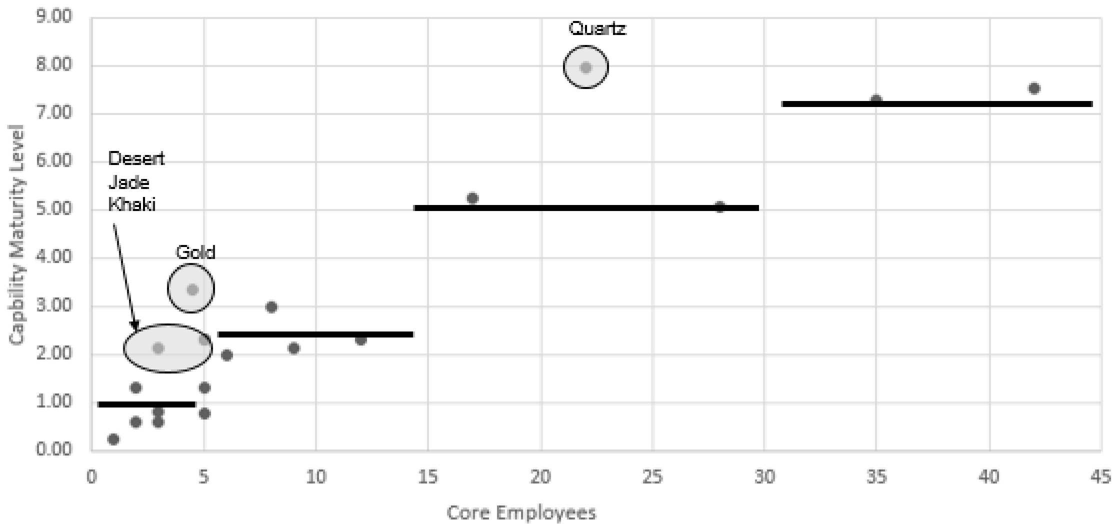


Figure 13. Illustrated Growth-Stages with Outliers

Variability in four of the five outliers is explained. Three outliers contribute their increased capability maturity to their company founders. The experienced founders of company Quartz created the company with the strategy to achieve conceived high levels of capability maturity as part of their business plan for funding and growth. Consistent with the business plan, company Quartz has doubled in size during the past fourteen months it has participated in this study. The company is currently adding another dozen employees. Thus, the company is effectively growing into its preconceived maturity.

Desert and Jade have founders that are also experienced, injecting capability maturity into their companies in anticipation of growth. Gold is one of the three older companies in the study. Gold's higher level of capability maturity is contributed to its response the complexity of outsourcing software development. There is no explanation in the research to explain Khaki as an outlier.

Terms and Distinction of Growth-Stages

The stages advanced in the Software Company Growth-Stage Model have been overlaid on the chart of Average Capability Maturity (Figure 14). While the growth-stages are fluid, once five outliers are filtered, the chart illustrates the Growth-Stage Models terms and employees numbers align quite nicely.

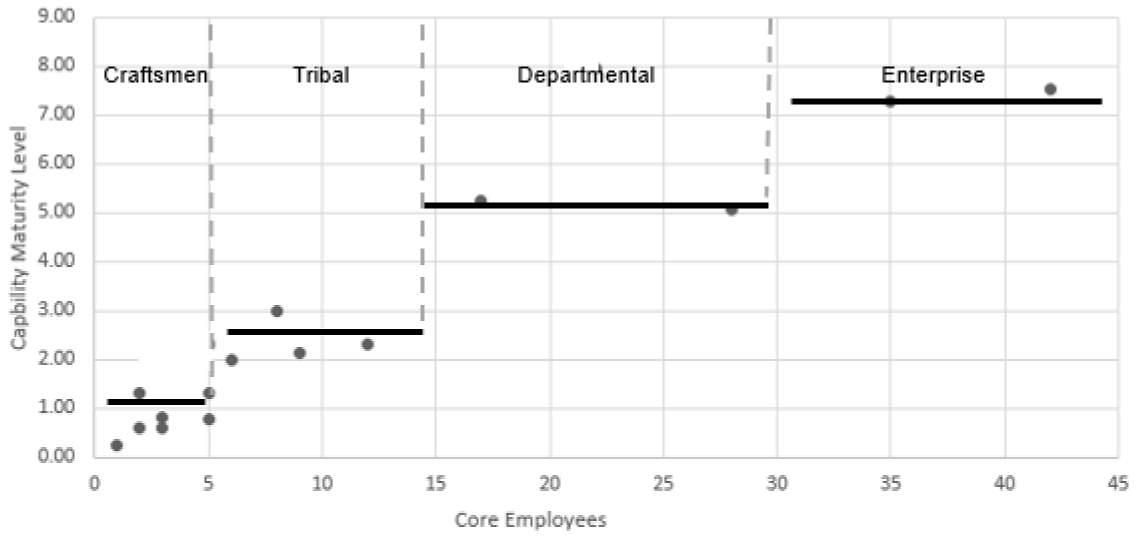


Figure 14. Growth-Stage Terms and Distinction of Stages

The proposition being advanced is not that there is absolute evidence of distinct points for stage differentiation. The evidence is that there is an interesting visual correlation that may be worth exploring is future research. In future research, the approach to consider would be to determine whether individual companies experience defined stages, whether those growth stages mirror the characteristics in the Growth-Stage Model, and whether those growth stages between companies correlate to each other.

Knowledge Sharing as a Determinant of Growth Stages

Knowledge sharing (dependent variable) correlates to core employees (independent variable) in the CMA-Instrument maturity appraisal. The studies 100 capability measures of Knowledge Sharing (n = 100) have a Pearson's Coefficient of .7372, similar to that of the entire data set. Figure 15 illustrates the average Knowledge Sharing measures for each company, with a clear positive trending correlation. However, there is no indication that knowledge sharing as measured by the CMA-Instrument, is a more of a driver of capability maturity than other measures of the CMA-Instrument.

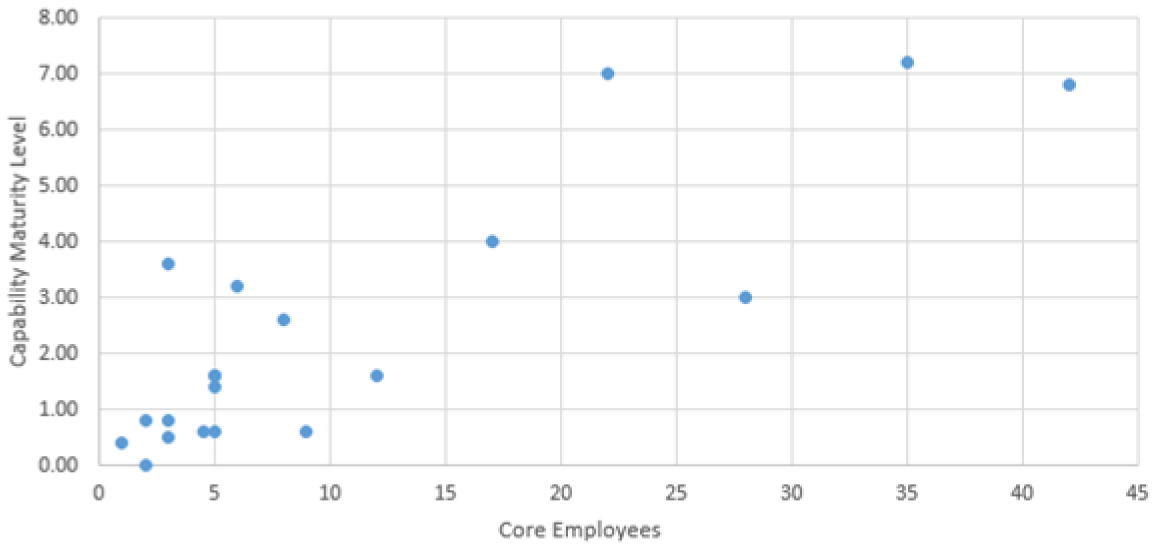


Figure 15. Average Capability Maturity of Knowledge Sharing Measure Appraised via CMA-Instrument

Support for Model

Each subject was presented with the Growth Stage Model and asked for their response. While it could superficially there was universal support for the model, the interviewer perceived that to state there was unqualified support would be an overstatement. Based on the impression of the interviewer, while the statements were positive, the subjects from those in smaller companies were interpreted to be less than enthusiastic. It is speculated many many of those interviewed have not experienced growth. Thus, there is little to which the many subject could relate.

It was not until reaching the size of company Peach that the level of exhibited interests was perceived to reached unqualified support. The exception was company Desert. Although a very small company, Desert has a founder that is an outlier in experience and education. Although Desert has only three core employees, the founder strongly endorsed the model as coalescing of her Fortune 500 experience and her ivy league MBA.

In interviews with the larger and older subject companies, each subject felt that they had gone through stages and each subject felt that the nature of knowledge-sharing changed with those stages. The companies Rose and Tan each spoke of stage-growth crises they had faced, and continue to face. The co-founder of Quartz spoke of stage-growth crises in a prior startup that shaped the capability maturity strategy for Quartz.

Capability Maturity Appraisal Instrument (from Chapter 2)

CMMI Class 'C' Appraisal

The structure of the Capability Maturity Appraisal Instrument is created with the CCMI Class C structured in mind. The CMA-Instruments following the documentation by CMMI

Institute and utilized professional expertise. The validity of the CMA-Instrument is not dependent on conformance to CMMI. But the believed conformance to the CMMI appraisal is an additional benefit of drawing from many years and thousands of appraisals performed by CMMI Institute approved professionals.

To further test for conformance of the CMA-Instrument to the CMMI structure, in future research one could pursue surveying CMMI certified professionals as to their opinions of conformity. Additionally, it would be beneficial to perform a parallel CMA-Instrument appraisal of a company being professionally appraised for a CMMI Class C appraisal. Results of the two instruments could then be compared.

Differences between the CMA-Instrument and CMMI

There are differences in the CMA-Instrument compared to a CMMI Class C Appraisal. First, the CMA-Instrument subdivides CMMI levels 1, 2 and 3 to generate eleven levels (counting zero), not 6. Second, the CMA Instrument adopts the Continuous Representation. The Continuous Representation omits level 4 and 5. Therefore, CMA-Instrument takes latitude by adopting the approach of both the CMMI Staged and Continuous Representations. These differences are not believed to be in conflict with a CMMI Class C Appraisal. These differences provide greater detail, thus enhancing what would be a CMMI Class C Appraisal.

Reasonably Accurate Measures

For the purpose of this research, the results were reasonably consistent between similar companies. Through the CMA-Instrument's assessments, similar companies were found to produce similarly measured levels of capability maturity. Companies that appeared as outliers typically were found to have reasons for variability.

In post-interview discussions, appraisal results were discussed with the interview subjects to gain their opinion on the reasonable assessment of their companies. The opinions of interview subjects were mainly supportive, given the descriptions provided with the appraisal ratings.

Several of smallest companies voiced concern that the capability maturity interview questions exposed their failure to sufficiently advance their companies. One company, Ecru, even stopped the interview mid-way through to voice his concern for the exposed state of his company's lack of maturity. The response to those concerned subjects was assured that his/her company, "was at maturity levels in keeping with companies of their respective size." Their maturity levels were "optimal" for their size.

A few companies exhibited higher levels of maturity than was typical. Desert was inordinately high in Knowledge Sharing. When questioned, the subject revealed her ivy-league MBA resulted in elevated personal value in Knowledge Sharing that is shaping her company's culture. Company Ivory had a higher measure in "baselines." When questioned, the subject revealed he and his co-founder possessed backgrounds in math and analytics background.

In summary, through the CMA-Instruments assessments, similar companies were found to produce comparable measured levels of capability maturity. Companies that appeared as outliers were typically found to have reasons for variability.

In future research to assess the validity of consistent results, subject companies could be interviewed more than once by multiple interviewers. The measured capability results could them compared for consistency.

Inherent Limitations

Both the CMA-Instrument and the CMMI Class ‘C’ Appraisal are thumbnail assessments. Thumbnail meaning that the assessments are surficial. In-depth observations and analyses are not performed on processes, procedures, and documentation. The level of assessment detail is fundamentally limited to questions of “do you have” and “do you do.” Therefore, there are a few unavoidable risks.

The risk of misunderstanding terms used in questions was anticipated. But, the degree of concern increased as the interviews progressed. No subject was familiar with all the terms in the appraisal questions. All the subjects required additional descriptions for multiple question terms. Subjects were most likely to be unfamiliar with project management terms that were are common in the industry. Because of the interactive nature that resulted in additional descriptions of the questions, concern arose that the descriptions could sway subjects’ answers.

The risk of misinterpreting how well or how complete a process or procedure was implemented was a sensitivity during the interviews. While the subjects may state that a measure is performed at a stated level, it is difficult to validate whether the measure is fully implemented. It was also difficult to determine how consistent a measure is performed or fully utilized.

The risk of obfuscation by software adds difficulty to the appraisal. Software for project management, software control, and collaboration may be in place within a company. However, it is difficult to ascertain; 1) the full breadth of software features that exists, 2) what features are utilized, 3) or the extent the software is fully embraced. For example, Jira, which was used by several subject companies, is a software package that is broad and deep in its support of Scrum software processes. However, the existence of a capability may not translate to effective implementation of a capability. Similarly, Slack was the most used collaboration software package, but without detailed observation of use by individuals, an assured accurate assessment is difficult.

Self-Assessment

It is believed that the CMA-Instrument has potential as a self-assessment tool by professionals for their own companies. During the course of the research, it became evident that if the CMA-Instrument where to progress to become a self-assessment tool, supportive guidelines, descriptions, and training materials would be required. The variability of understanding was high in what was initially believed to be broadly understood industry terms.

Does Project Management Advance First?

Early results of the interviews showed unexpectedly high measures in Project Managemen. The question surfaced, “does capability maturity advance first in project management?” As it turned out, there were some outliers in the early interviews. In particular, company Gold screwed the results with high project management measure. Removing Gold as an outlier, the average capability maturity levels of eleven subject companies with five or less employees for the three maturity categories of Project Management, Software Control, and Knowledge Sharing were respectively 1.3, 1.4.and 1.3. Gold measure of 6.3 in project management capability maturity was the result of long-term management of outsourced development (i.e. a moderator of maturity due to complexity).

Complexity: Moderators of Required Capability Maturity (from Chapter 3)

Software Platform Complexity as a Moderator

Interviews produced support for the conceptual Software Complexity Model. The Model provided a clear and concise means to discuss with subjects the type of software platforms from which they currently operated and the direction in which their software platforms where headed. Company Rose and company Quartz are companies that enthusiastically endorsed the model, stating that the Software Complexity Model was a clear presentation of one of the greatest challenges facing either company. Company Gold endorsed the model as an illustration of its strategy to control of software platform complexity.

Company Rose has a mature software platform that is classified as an “Integrated Platform.” The software platform of company Rose is a result of several years of organic development creating layers of interrelated code. The weight of platform maintenance is one of the company’s greatest concerns. Attempts to componentize the platform have failed, primarily due to client demands for immediate improvement to the old code, thus attempts to update the existing foundational code have been indefinitely delayed to address ongoing needs. Management of software decay has consumed an increasingly high percentage of company Rose’s resources, while software updates pose increasingly high risks of bugs.

Company Quartz is proactively addressing software complexity. Based on experiences with prior startups, the founders of company Quartz place a high emphasis on complexity countermeasures. Creep of product features is strictly controlled through a highly defined scope control process. The company’s “Componentized Product” is continually updated through refactored (rebuilding) of its software component modules.

Gold, a small but older company of 4.5 full-time-equivalent core employees, is able to maintain lower maturity levels in software controls. The founder knowingly controls its software platform by an approach of “Componentized Product” for the main platform modules, combined with layering “Individual Projects” for each client. The approach allows company Gold to effectively counteract outsourcing complexity with a reduction in software platform complexity.

Company Tan endorsed the Software Platform Complexity Model. However, exposure to the model does not appear to have aided the company in its challenges. Company Tan participated in this research for just over one year. At the time of the first interview, Tan intended to release a beta product within a couple months. Research results speculated the company lacked the capability maturity to effectively manage an “Integrated” platform with complexity beyond that of a typical beta version. Tan’s beta version has been delayed nine month due to complications. The most recent capability maturity measures indicate continued deficits in Software Control.

Other Complexity Factors as Moderators

While complexity factors were mentioned to varying degrees by some companies as moderators of required capability maturity, the most significant finding is the lack of response to complexity factors by the majority of subject companies. In general, far fewer complexity factors surfaced than expected. It was speculated that the absence of such factors was due to the young age and small size of most of the subject companies. Because of the lack of factors noted through the interviews, the second set of interviews was initiated with five large organization. Interviews, specifically to explore complexity factors, took place with five large multinational software companies, plus with an IT representative of a 911 system for a large city. It is through these interviews that the majority of the complexity factors were identified that are displayed in Figure 18 of Chapter 3.

Differed Capability Maturity

During the interview process, it became evident that most startups were not addressing the long-term implications and risks associated with complexity. Most evident was the absence of concern by companies for the complexity and risks resulting from outsourcing of software development. There was little to no outsourcing concern by companies with under ten employees. Of the nine companies with six or fewer core employees that outsourced development, only Gold appeared to respond to outsourcing complexity. Note that Gold is seven years in business, while the other eight have been in business three years or less. Of the eight younger companies, all either did not recognize the topic as a concern or in the case of three companies, stated (paraphrased) that “this would be the job of a CTO (when hired).” For the most part, the subjected companies trusted the outsourced developers/companies to meet schedules and provide quality software. Subjects also assumed that the outsourced software would easily transition to in-house developers when and if that step was taken.

The evidence that these subject startups did not respond to complexity would provide anecdotal evidence in support of the S-Curve of Optimal Maturity. The startups are differing capability maturity requirements at this early size of six or fewer core employees, thus avoiding consumption of resources at this early growth-stage.

In future research into complexity factors, in is recommended to study older and larger companies. Larger and older companies appear most likely to produce evidence of a broader range of complexity factors and responses to those factors.

The Optimal Maturity Model (from Chapter 4)

S-Curve of Optimal Maturity

The concept of the optimal maturity S-Curve is based on two premises. First, it is contended that capability maturity can be less complex in smaller size companies because knowledge sharing naturally occurs when employee numbers are small, and those employees work next one another. Second, it is contended when a company is quite small; capability maturity can differ for some time. Small, young company can more easily increase low levels of capability maturity because low levels of capability maturity are less onerous, young companies have little or no pre-existing systems to burden implementations, and it is easier to implement process and procedures with fewer people.

The evidence in this research indicated that smaller companies avoided or completely ignored many aspects of capability maturity and complexity. If we define Optimal Capability Maturity as the average maturity of a given size company, adjusting for complexity, then it is clear that the average company of ten employees or less is differing capability maturity. Because of this capability maturity deferment, we conceptually recognize an initial relatively flat line in capability maturity that will increase in slope as the company recovers from the capability maturity different. Based on the anecdotal evidence, it is believed that the recovery and stepping slope off the Optimal Maturity curve occur somewhere in the range of 15 to 40 core employees. This theory would benefit from future research focused on companies in the size range.

Note that the CMA-Instrument is not designed to produce an S-Curve. The instrument subdivides the lower maturity level. The result is a straightening of the curve and small incremental improvement at the low capability maturity levels are recognized.

Optimal Maturity as a Diagnostic Tool

An example of an application of the Optimal Maturity Model as a diagnostic tool would be an investor determining if a company is operating at a capability deficit, thus investment might better proceed with greater caution. Another example would be if company is taking outlying measures driven by the founders' bias, when such capability is not optimally justified by the size of the company.

The Optimal Maturity Model should be of great value to investors in startups. Seventeen of the twenty subject companies are seeking funding. An investor in a startup is buying a portion of the company with the expectation of a future return. While a startup may promote an appealing opportunity, one question of investors should be whether the startup can fulfill the expectations of building a company. The diagnostic function of the Optimal Maturity Model would provide investors with this needed insight.

The Optimal Maturity Model would give an investor insight into whether a company's operations can support the company's vision. Of the subject companies that have been in business less than two years, when asked about plans for growth, universally the plans for growth were stated regarding position that would be hired. No subject spoke about what systems would be advanced, or what the newly hired employees would contribute to business operations. Founders, business managers, and investors could use the Optimal Maturity Model to project what system (capability maturity) advancement are needed to support the newly hired employees.

A note from the interviewer is that it was surprising and disconcerting that almost universally that companies with less than ten core employees had such a high degree of differed capability maturity. Most of these companies appeared ill-prepared for growth, thus ill-suited to receive investment capital that they sought. Very low capability maturity for a very small size is a fundamental concept of the S-Curve. Interview observations were in keeping with the concept of the S-Curve..

Optimal Maturity as a Predictive Guide

Most subject companies disclosed in their interviews that they anticipate growth. However, most of those subjects had only a limited idea of what actions they needed to take to support growth. The model would provide insight as to where resources should be devoted to advance capability maturity that supports pending growth. The positive result would be to aid in counteracting "growing pains."

To draw an analogy, in architecture there is a phase of "form follows function."   That phrase could apply to organization development. The approach would be to know what operational advances need to take place for a given stage of growth (i.e. form) and then hire those with skills to advance those needs (i.e. functions). The Optimal Maturity Model provides a company with tools to gain insight into the functional improvements needed at each stage so that they can then create the form of the company to optimally fulfill those needs.

Driving Capability Maturity via Prospects of Funding

A premise of this paper is that the number of core employees will drive capability maturity. However, this premise might be modified to state that it is the number of anticipated employees that drive capability maturity. In fact, one of the proposed uses of the Optimal Maturity Model is that it is a predictive tool to provide insight into levels of capability maturity that would be required at higher employment numbers.

Two companies, Desert and Quartz, each described their efforts to advance their capability maturity in anticipation of investor funding. Both companies were actively engaged in hiring, even though investor funding had not been secured. It can be further highlighted that Quartz pursued high levels of capability maturity as part of its funding plan, which only indirectly pertained to the number of employee.

Given the importance in investor funding, a relationship to explore specifically for startup companies would be how strong the relationship is between investor funding or anticipated investor funding and capability maturity.

Driving Capability Maturity via Customers/Client

Sienna is a company that contributed advancement of many of its systems to the needs to support the high number of total employees, in particular a large number of call center employees (>100), who had less professional skills and incurred greater turnover. While there was only a minimal relationship between software systems and the call systems, the company had a culture of process engineering, documentation, and training that pervaded the software teams. This observation raises the question that it may not be sufficient to count only core employees engaged in software development. Total employees may affect the independent variable of core employees and the corresponding capability maturity.

Driving Capability Maturity via Founder Experience

Founders experience may significantly affect a firm's capability maturity. The firms Desert, Navy, and Quartz each mentioned prior experience and a determinant in their emphasis in creating systems, with the future operations of their companies in mind.

The MVP (Minimal Viable Product) Trap

The term MVP was mentioned by fourteen of the fifteen smallest companies. It was difficult to tell in isolation whether the term was being used as part of a product advancement strategy, or whether it was an excuse to launch a minimally solidified product that lacks operational capabilities and lacks a vision in the marketplace. As a cautionary note resulting from the research interviews, when the term MVP is presented, it may be warranted to look a little deeper at the organization's capability maturity and other business components.

Decay of Capability Maturity

Software code decay is a topic in the literature review of Chapter 3. However, similar research published on the decay of capability maturity was not found. While the term "capability maturity decay" was not used specifically by subject in interviews, it was evident that capability maturity decay has a significant impact on the operations of established organizations.

Of the subject companies, both Sienna and Rose spoke of the challenges of maintaining system once established. Each of the five companies in the additional complexity interviews mentioned challenges that would fall into the realm of capability maturity decay. Policies, procedures, and training materials start down a path of obsolescence from the moment there are created. Without vigilance, knowledge bases rapidly collect sediment (out of date documents). Various software applications, upon which an organization becomes dependent, will change or fails to serve the evolving needs of the organization. The best possesses and operating systems may adulterate or be ignored as alternative habits pervade employee behaviors.

During the Interview with Siena, the interviewee qualified statements in answers regarding capability maturity to the effect that “they were better in the past.” In the interview with Rose, several answers were of how they “once did things differently.”

The topic of capability maturity decay would appear to be an opportunity for future research.

CHAPTER 8. CONCLUSIONS

A strong relationship is evidenced in the data between the number of core employees as the independent variable, and capability maturity as the dependent variable. There is promising evidence that growth stages exist and that while fluid, the points of the stages are common between companies. The Software Company Growth-Stage Model illustrates the growth stages and advances what shows to be a possible point of differentiation between the stages. Further research could focus on these stage distinction points, and well as delving further into the drivers of differentiation between the stages, which may prove to be characteristics of knowledge sharing by group size.

The Capability Maturity Appraisal Instrument was shown to be an effective instrument for capability maturity assessments within the scope of this research. It is believed that the instrument shows promise for use in future research, and the instrument shows promise as a self-assessment tool for practitioners.

The absence of recognized complexity in smaller companies as a moderator of required capability maturity supports the concept of the S-Curve in optimal capability maturity. Anecdotal evidence in the large subject companies, as well as anecdotal evidence in the interview of the five large organizations indicate the importance of complexity as a modifier of required capability maturity as companies grow and age.

Capability maturity data from the research substantiates the validity of the Optimal Maturity Model. The number of core employees is shown to be a driver of required capability maturity. The relationship allows the model to be utilized as a diagnostic tool for capability maturity that a company should typically possess. The relationship also allows the model to be utilized as a predictive guide so that a company may anticipate potentiL growth stresses that they are likely to encounter.

REFERENCES CITED

Achtenhagen, L., Naldi, L., & Melin, L. (2010). Business Growth - Do Practitioners and Scholars Really Talk About the Same Thing? *Entrepreneurship Theory and Practice*, 34(2), 289-316.

Anacleto, A., von Wangenheim, C. G., Salviano, C. F., & Savi, R. (2004, April). Experiences Gained from Applying ISO/IEC 15504 to Small Software Companies in Brazil. In *4th International SPICE Conference on Process Assessment and Improvement, Lisbon, Portugal* (pp. 33-37).

Banker, R. D., Davis, G. B., & Slaughter, S. A. (1998). Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study. *Management Science*, *44*(4), 433–450.

Basili, V. R., & Perricone, B. T. (1984). Software Errors and Complexity: An Empirical Investigation. *Communications of the ACM*, *27*(1), 42-52.

Blanchette, S. and Keeler, K. (2005). Self-Assessment and the CMMI-AM – A Guide for Government Program Managers (CMU/SEI-2005-TN-004). *Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA.*

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P. & Zazworka, N. (2010, November). Managing Technical Debt in Software-Reliant Systems. *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research* (pp. 47-52). ACM.

CMMI Institute (2015) Maturity Profile, 1 January 2007 – 31 December 2015. *CMMI Institute.* Retrieved from http://partners.cmmiinstitute.com/wp-content/uploads/2016/03/Maturity-Profile-Ending-Dec-31-2015-Quality-20150301.pdf

CMMI Product Team (2002). Capability Maturity Model® Integration (CMMI), Version 1.1--Continuous Representation. Software Engineering Institute, Carnegie Mellon University Pittsburg, PA.

CMMI Product Team. (2010). CMMI for Development, Version 1.3 (CMU/SEI-2010-TR-033). *Software Engineering Institute, Carnegie Mellon University Pittsburg, PA.*

Dedrick, J., Gurbaxani, V., & Kraemer, K. L. (2003). Information Technology and Economic Performance: A Critical Review of the Empirical Evidence. *ACM Computing Surveys (CSUR)*, *35*(1), 1-28.

De Silva, L., & Balasubramaniam, D. (2012). Controlling Software Architecture Erosion: A Survey. *Journal of Systems and Software*, *85*(1), 132-151.

Eick, S. G., Graves, T. L., Karr, A. F., Marron, J. S., & Mockus, A. (2001). Does Code Decay? Assessing the Evidence from Change Management Data. *Software Engineering, IEEE Transactions on*, *27*(1), 1-12.

Fayad, M. E., & Laitnen, M. (1997). Process Assessment Considered Wasteful. *Communications of the ACM*, *40*(11), 125-128.

Fowler, M., & Highsmith, J. (2001). The Agile Manifesto. *Software Development*. 9(8), 28-35.

Gaibraith, Jay. (1982). The Stages of Growth. *Journal of Business Strategy,* 3(1), 70-79.

Greiner, L. E. (1972). Evolution and Revolution as Organizations Grow. *Harvard Business Review,* Reprint May-June, 1998.

Halstead, Maurice H. (1977). *Elements of Software Science (Operating and Programming Systems Series)*. Elsevier Science Inc.

Hansen, B., & Hamilton, R. T. (2011). Factors Distinguishing Small Firm Growers and Non-Growers. *International Small Business Journal*, 29(3), 278-294.

Harrison, W., Magel, K., Kluczny, R., & DeKock, A. (1982). Applying Software Complexity Metrics to Program Maintenance. *Computer*, *9*(15), 65-79.

Henry, S., & Kafura, D. (1981). Software Structure Metrics Based on Information Flow. *Software Engineering, IEEE Transactions on*, (5), 510-518.

Holweg, M (2007). The Genealogy of Lean Production. *Journal of Operations Management. 25*(2), 420-437.

Humphrey, W. S. (1988). Characterizing the Software Process: A Maturity Framework. *Software, IEEE*, *5*(2), 73-79.

Hwang, S. M. (2009). Process Quality Levels of ISO/IEC 15504, CMMI and K-model. *International Journal of Software Engineering and Its Applications*, *3*(1), 33-42.

Jiang, J. J., Klein, G., Hwang, H. G., Huang, J., & Hung, S. Y. (2004). An Exploration of the Relationship Between Software Development Process Maturity and Project Performance. *Information & Management*, *41*(3), 279-288.

Khurshid, N., Bannerman, P. L., & Staples, M. (2009). Overcoming the First Hurdle: Why Organizations Do Not Adopt CMMI. *Trustworthy Software Development Processes* (pp. 38-49). Springer Berlin Heidelberg.

Kushnir, Khrystyna. (2006, Jun 16). How Do Economies Define Micro, Small and Medium Enterprises (MSMEs)? *International Finance Corporation – World Bank Group.* Retrieved from http://www.ifc.org/wps/wcm/connect/624b8f804a17abc5b4acfddd29332b51/msme-ci-note.pdf?mod=ajperes

Levie, J., & Lichtenstein, B. B. (2010). A Terminal Assessment of Stages Theory: Introducing a Dynamic States Approach to Entrepreneurship. *Entrepreneurship Theory and Practice*, 34(2), 317-350.

Marcal, A. S. C., de Freitas, B. C. C., Furtado Soares, F. S., & Belchior, A. D. (2007, March). Mapping CMMI Project Management Process Areas to SCRUM Practices. *Software Engineering Workshop, 2007. SEW 2007. 31st IEEE*, 13-22.

Martin, J. (1991). *Rapid Application Development* (Vol. 8). New York: Macmillan.

McCabe, T. J. (1976). A Complexity Measure. *Software Engineering, IEEE Transactions on*, (4), 308-320.

McClure, C. L. (1978, May). A Model for Program Complexity Analysis. *Proceedings of the 3rd International Conference on Software Engineering* (pp. 149-157). IEEE Press.

Parnas, D. L. (1972). On the Criteria to be Used in Decomposing Systems Into Modules. *Communications of the ACM*, *15*(12), 1053-1058.

Parnas, D. L. (1994, May). Software Aging. *In Proceedings of the 16th International Conference on Software Engineering* (pp. 279-287). IEEE Computer Society Press.

Paulk, M., Weber, C., Curtis, B., and Chrissis, M., (1993) Capability Maturity Model for Software (Version 1.1). Technical Report CMU/SEI-93-TR-024 ESC-TR-93-177. *Software Engineering Institute, Carnegie Mellon University. Pittsburgh, PA.*

Peldzius, S., & Ragaisis, S. (2011, January). Comparison of Maturity Levels in CMMI-DEV and ISO/IEC 15504. *Proceedings of 5th WSEAS International Conference on Computer Engineering and Applications,* 29-31.

Penrose, E. T. (1959). *The Theory of the Growth of the Firm*. Oxford University Press. Cited 23,344

Pino, F. J., Baldassarre, M. T., Piattini, M., & Visaggio, G. (2010). Harmonizing maturity levels from CMMI-DEV and ISO/IEC 15504. *Journal of Software Maintenance and Evolution: Research and Practice*, *22*(4), 279-296.

Poppendieck, M. (2007, May). Lean Software Development. *Companion to the proceedings of the 29th International Conference on Software Engineering* (165-166). IEEE Computer Society.

Project Management Institute (2013a). A Guide to the Project Management Body of Knowledge (PMBOK Guide). Fifth Edition. *Project Management Institute, Pennsylvania*.

Project Management Institute (2013b). Organizational Project Management Maturity Model (OPM3). Third Edition. *Project Management Institute, Pennsylvania*.

Project Management Institute. (2014) *PMI Today*. *July*: 4.

Project Management Institute (2015, November 10). *PMI OPM3 Frequently Asked Questions*. Retrieved from http://www.pmi.org/~/media/PDF/Business-Solutions/OPM3%20FAQ.ashx

Quinn, Robert E., and Kim Cameron. (1983) Organizational Life Cycles and Shifting Criteria of Effectiveness: Some Preliminary Evidence. *Management Science* 29.1: 33-51.

Ries, E. (2011). *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Books.

Rubin, H. J., & Rubin, I. S. (2011). Qualitative Interviewing: The Art of Hearing Data. *Sage Publications.*

Rutherford, M. W., Buller, P. F., & McMullen, P. R. (2003). Human Resource Management Problems Over The Life Cycle of Small to Medium-Sized Firms. *Human Resource Management*, 42(4), 321-335.

SCAMPI Upgrade Team. (2011). Appraisal Requirements for CMMI Version 1.3 (ARC, V1.3) (CMU/SEI-2011-TR-006). *Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA.*

Scott, M., & Bruce, R. (1987). Five Stages of Growth in Small Business. *Long Range Planning*, 20(3), 45-52.

Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., & Murphy, R. (2007). An Exploratory Study of Why Organizations Do Not Adopt CMMI. *Journal of Systems and Software*, *80*(6), 883-895.

Staples, M., & Niazi, M. (2008). Systematic Review of Organizational Motivations for Adopting CMM-Based SPI. *Information and Software Technology*,*50*(7), 605-620.

Steinmetz, L. L. (1969). Critical Stages of Small Business Growth: When They Occur and How to Survive Them. *Business Horizons*, 12(1), 29-36.

Sugimori, Y., Kusunoki, K., Cho, F., & Uchikawa, S. (1977). Toyota Production System and Kanban System Materialization of Just-In-Time and Respect-For-Human System. *The International Journal of Production Research*, *15*(6), 553-564.

U.S. Small Business Administration. (2014, July 14). Table of Small Business Size Standards Matched to North American Industry Classification System Codes. *Small Business Size Standards*. Retrieved from https://www.sba.gov/sites/default/files/files/Size_Standards_Table.pdf

Weinzimmer, L. G., Nystrom, P. C., & Freeman, S. J. (1998). Measuring Organizational Growth: Issues, Consequences and Guidelines. *Journal of Management*, *24*(2), 235-262.

Womack, J. P., Jones, D. T., & Roos, D. (1990). *Machine that Changed the World*. Simon and Schuster.

Yin, R. K. (2013). *Case Study Research: Design and Methods*. Sage Publications.

APPENDICES

Appendix ‘A’ –CMMI Maturity Self-Evaluation

The following table is the full set of survey questions for a CMMI-AM Self Evaluation, published by the Software Engineering Institute (Blanchette and Keeler, 2005).

The survey uses a Likert Scale from 1 to 10. In the table below, the left column defines a “1”, and the right column defines a “10.”

	1 2 3 4 5 6	7 8 9 10
1	Estimates are based on wild guesses or dictated from above.	Estimates of project planning parameters (i.e., scope, task attributes, lifecycle, cost, effort) are established and maintained.
2	Plans are rarely written down nor do they reflect current project activities.	A project plan is established and maintained as the basis for managing the project.
3	We rarely seek commitments from those affected by the project plan.	Commitments to the project plan are established and maintained.
4	We track progress based on personality and an arbitrary baseline.	Actual performance and progress of the project are monitored against the project plan.
5	It is difficult to know when the project has deviated from the plan based on the data we review.	Corrective actions are managed to closure when the project’s performance or results deviate significantly from the plan.
6	There are no organizational assets available to assist in conducting the project	The project is conducted using a defined process that is tailored from the organization’s set of standard processes
7	Relevant stakeholders for our project are avoided or unknown	Coordination and collaboration of the project with relevant stakeholders are conducted.
8	Project team members do not share a common vision of success.	The project is conducted using the project’s shared vision
9	Our integrated teams are ad hoc and ill-defined.	The integrated teams needed to execute the project are identified, defined, structured, and tasked
10	Our program lacks a coherent risk management strategy, roles are ill-defined, and my responsibilities for participation in the process is not clear.	Preparation for risk management is conducted.
11	We deal with problems and issues, there’s no time to think proactively.	Risks are identified and analyzed to determine their relative importance.

12	Risk mitigation is ad hoc, and only dealt with in crisis mode.	Risks are handled and mitigated, where appropriate, to reduce adverse impacts on achieving objectives.
13	Our project scrambles to prepare for solicitation activities and has to “make it up” on the fly.	The project is prepared to conduct the solicitation.
14	Our suppliers are selected based on political whims.	Suppliers are selected based on the solicitation package.
15	Our contracts do not provide for the tasks, deliverables, and insight to meet our needs.	Contracts are issued based on the needs of the acquisition and the suppliers’ proposed approaches.
16	We get little or no insight into the processes used by our suppliers or their interim work products.	Work is coordinated with suppliers to ensure the contract is executed properly
17	Our project team has a hard time knowing what the requirements baseline really is	Requirements are managed and inconsistencies with project plans and work products are identified.
18	Our set of requirements for this project do not reflect the needs or expectations of the project's stakeholders.	Stakeholder needs, expectations, constraints, and interfaces are collected and translated into customer requirements.
19	Our requirements are at such a high level, you could drive a truck through them.	Customer requirements are refined and elaborated to develop product and product component requirements.
20	It’s hard to tell if our requirements will result in a useful system.	The requirements are analyzed and validated, and a definition of required functionality is developed.
21	Our verification activities are undefined	Preparation for verification is conducted.
22	We never review our own work before sending it out.	Peer reviews are performed on selected work products.
23	We rarely verify work products against their specified requirements	Selected work products are verified against their specified requirements.
24	Our validation activities are undefined.	Preparation for validation is conducted.
25	We never know if a product will be usable in its intended environment until it actually gets there.	The product or product components are validated to ensure that they are suitable for use in their intended operating environment.
26	We collect all kinds of data for no apparent reason.	Measurement objectives and activities are aligned with identified information needs and objectives.
27	I don’t get the data I need, and when I do get data, I don’t believe it.	Measurement results that address identified information needs and objectives are provided.

28	The boss makes all the decisions.	Decisions are based on an evaluation of alternatives using established criteria
29	We ignore operational issues and logistics—it just slows us down to listen to those guys.	Preparation for transition to operations and support is conducted.
30	We don't have time to think about transition criteria, we need to get our products to the field.	Transition decisions and actions are executed in accordance with transition criteria.

Appendix 'B' - CMMI Appraisal Detail

Requirements	**Class A**	**Class B**	**Class C**
Method Documentation			
4.1.1 – Documentation of method	Yes	Yes	Yes
4.1.2 – Identifying appraisal purpose and objectives	Yes	Yes	Yes
4.1.3 – Model scope	Yes	Yes	Yes
4.1.5 – Team member selection	Yes	Yes	Yes
4.1.7 – Size of team	Yes	Yes	Yes
4.1.8 – Sponsor, team leader, and team member roles and responsibilities	Yes	Yes	Yes
4.1.9 – Estimating appraisal resources	Yes	Yes	Yes
4.1.10 – Logistics	Yes	Yes	Yes
4.1.11 – Collecting and mapping data to appraisal reference model	Yes	Yes	Yes
4.1.12 – Creation of findings	Yes	Yes	Yes
4.1.13 – Assuring confidentiality and non-attribution	Yes	Yes	Yes
4.1.14 – Appraisal record yes partial	Yes	Partial	Partial
Planning and Preparing			
4.2.1 – Preparation of participants	Yes	Yes	Yes
4.2.2 – Development of appraisal plan	Yes	Yes	Yes
4.2.3 – Sponsor approval of appraisal plan	Yes	Yes	Yes
Data Collection, Consolidation, and Validation			
4.3.1 – Data from interviews	Yes	Yes	1+
4.3.2 – Data from documents	Yes	Yes	1+
4.3.3 – Consensus of team members	Yes	Yes	Yes
4.3.4 – Accuracy of findings	Yes	Yes	Yes

4.3.5 – Verification of findings	Yes	Yes	Opt.
4.3.6 – Corroboration of objective evidence	Yes	Opt.	Opt.
4.3.7 – Sufficiency of data	Yes	Yes	Yes
4.3.8 – Preliminary findings preparation	Yes	Opt.	Opt.
4.3.9 – Preliminary findings validation	Yes	Opt.	Opt.
Rating			
4.4.1 – Define a rating process	Yes	N/A	N/A
4.4.2 – Basis for maturity level and capability level rating	Yes	N/A	N/A
4.4.3 – Rules for goal rating	Yes	N/A	N/A
4.4.4 – Rules for process area rating	Yes	N/A	N/A
4.4.5 – Rules for maturity level rating	Yes	N/A	N/A
Reporting Results			
4.5.1 – Report results to sponsor and appraised organization	Yes	Yes	Yes
4.5.2 – Retention of appraisal record	Yes	Yes	Yes

Applicability of ARC Requirements to Appraisal Method Classes
Requirements for CMMI Version 1.3 (ARC, V1.3) (CMU/SEI-2011-TR-006). Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA.

Appendix 'C' - Chapter 2: End Notes

1. From website posts reviewed on 11-13-2015, costs for just a CMMI appraisal range from $20,000 to $100,000, with Level 5 appraisal potential being much higher.

- https://www.quora.com/Roughly-how-much-does-an-CMMI-evaluation-cost-for-a-small-business-less-than-100-developers
- http://cmmirocks.ning.com/forum/topics/estimated-cost-for-scampi

One company quoted a low rate of $15,000 plus travel for a Class C appraisal (http://www.bcgiso.com/cmmi/gap.html). Achieving a given CMMI Level is too widely varied, but several consultants stated that it is common for oversight fees to exceed $125,000 for a typical minimal one year period, with many projects being much higher. These costs are separate from the expenditures often associated to improve a company's maturity.

2. Examples of online CMMI appraisal tools are listed below. Each are websites for companies that sell training services.

- Clearmodel, LLC. http://www.assessyourcapability.com. Available at the cost of $199
- SmartMatix. http://www.smartmatix.com/home/cmmiquickselfassessment.aspx. Available upon a form request for access
- Griffith University. https://www.sqi.griffith.edu.au/AppraisalAssistant/about.html. Available upon registration and email confirmation.

3. CMMI reports may be downloaded at no costs from the Software Engineering Institute at Carnegie Melon University (http://www.sei.cmu.edu/).

- CMMI® for Development, Version 1.3, http://www.sei.cmu.edu/reports/10tr033.pdf
- Appraisal Requirements for CMMI® Version 1.3, http://resources.sei.cmu.edu/asset_files/technicalreport/2011_005_001_15383.pdf

The costs for the ten ISO/IEC 15504 report average about $200 each, for a total cost of approximately $2,000. (http://www.iso.org/iso/home.htm)

Appendix 'D' – CMA-Instrument Notes by Company

Twenty data tables follow, one for each subject company, which include raw interview notes pertaining to the capability assessment for the respective companies. "Level" is the level assessed for each of the capability factors, along with the notation of "A" or "P" indicating whether the factors is an artifact (physical or digital) or process.

Company: Azure

Factor	*Level*	*Notes*	*A/P*
Project Management	**0.1**		
1. Change Control Log	0	none	A
2. Charter	1	mostly verbal betwee cofounders	A
3. Project Schedule	0	none	A
4. Project Schedule Control Process	0	none	A
5. Requirement Document (Traceability Matrix)	0	none, division of responsibility between cofounders	A
6. Scope & Change Control Process	0	none	P
7. Work Breakdown Structure	0	none, division of responsibility between cofounders	A
Software Controls	**0.2**		
8. Defect Logs	0	none	A
9. Defined Coding Standards	0	none	A
10. Peer Review of Code	0	none	P
11. Performance Baselines	0	none	A
12. Testing & Release Segmentation	0	no preplanning	P
13. Verification & Validation	1	limted, division of responsibility between cofounders	P
Knowledge Sharing	**0.4**		
14. Documentation & Archiving (e.g. Knowledgebase)	0	none. Some information could be track down via Slack communication	A
15. Integrated Knowledgebase Index	0	none	A
16. Training Documentation	0	none	A
17. Training Sessions	0	none	P
18. Collaboration Systems	2	Using Slack	A

Company: Beige

Factor	*Level*	*Notes*	*A/P*
Project Management	**0.9**		
1. Change Control Log	2	ew changes to this point	A
2. Charter	4	Reasonable definition for contractors	A
3. Project Schedule	0	dependent on contract company	A
4. Project Schedule Control Process	0	dependent on contract company	A
5. Requirement Document (Traceability Matrix)	0	dependent on contract company	A
6. Scope & Change Control Process	0	dependent on contract company	P
7. Work Breakdown Structure	0	dependent on contract company	A
Software Controls	**0.7**		
8. Defect Logs	0	none	A
9. Defined Coding Standards	0	dependent on contract company	A
10. Peer Review of Code	0	dependent on contract company	P
11. Performance Baselines	0	none	A
12. Testing & Release Segmentation	0	dependent on contract company	P
13. Verification & Validation	4	Co-founders responsible	P
Knowledge Sharing	**0.0**		
14. Documentation & Archiving (e.g. Knowledgebase)	0	none	A
15. Integrated Knowledgebase Index	0	none	A
16. Training Documentation	0	none	A
17. Training Sessions	0	none	P
18. Collaboration Systems	0	cofounders work in same room	A

Company: Coral

Factor	*Level*	*Notes*	*A/P*
Project Management	**1.9**		
1. Change Control Log	2	Founder tracks. Scalability in question	A
2. Charter	5	Reasonably good documentation by Founder	A
3. Project Schedule	1	dependant on contracted company to provide	A
4. Project Schedule Control Process	1	dependant on contracted company to provide	A
5. Requirement Document (Traceability Matrix)	2	Founder tracks. Scalability in question	A
6. Scope & Change Control Process	2	Founder tracks. Scalability in question	P
7. Work Breakdown Structure	0	in hands of contracted company	A
Software Controls	**0.8**		
8. Defect Logs	2	Founder tracks. Scalability in question	A
9. Defined Coding Standards	0	entirely in hands of contracted company	A
10. Peer Review of Code	0	entirely in hands of contracted company	P
11. Performance Baselines	0	entirely in hands of contracted company	A
12. Testing & Release Segmentation	0	entirely in hands of contracted company	P
13. Verification & Validation	3	Process by Founder	P
Knowledge Sharing	**0.8**		
14. Documentation & Archiving (e.g. Knowledgebase)	2	Founder tracks. Scalability in question	A
15. Integrated Knowledgebase Index	0	none	A
16. Training Documentation	0	none	A
17. Training Sessions	0	none	P
18. Collaboration Systems	2	Active collaboration with Advisors	A

Company: Desert

Factor	*Level*	*Notes*	*A/P*
Project Management	**1.1**		
1. Change Control Log	0	none	A
2. Charter	3	defined by founder	A
3. Project Schedule	1	none	A
4. Project Schedule Control Process	0	controlled by founder	A
5. Requirement Document (Traceability Matrix)	0	dependent on contractors	A
6. Scope & Change Control Process	3	controlled by founder	P
7. Work Breakdown Structure	1	dependent on contractors	A
Software Controls	**1.7**		
8. Defect Logs	1	Using Twillio for all development	A
9. Defined Coding Standards	3	tools inherant in Twillio, but limited utilization of tools	A
10. Peer Review of Code	1	tools inherant in Twillio, but limited utilization of tools	P
11. Performance Baselines	1	tools inherant in Twillio, but limited utilization of tools	A
12. Testing & Release Segmentation	1	tools inherant in Twillio, but limited utilization of tools	P
13. Verification & Validation	3	founder and staff	P
Knowledge Sharing	**3.6**		
14. Documentation & Archiving (e.g. Knowledgebase)	5	active approach to knowledge accumilation	A
15. Integrated Knowledgebase Index	4	central document archiving, with limited scalability	A
16. Training Documentation	4	new employees document everything they learn	A
17. Training Sessions	1	new employees are supported and access documents	P
18. Collaboration Systems	4	Using Slack	A

Company: Ecru

Factor	*Level*	*Notes*	*A/P*
Project Management	**0.4**		
1. Change Control Log	1	dependency on founder, with paper records	A
2. Charter	1	limited graphic design info	A
3. Project Schedule	1	depent on contractors	A
4. Project Schedule Control Process	0	none, other than founder decisions	A
5. Requirement Document (Traceability Matrix)	0	none, other than founder's recolection	A
6. Scope & Change Control Process	0	depenant on founder	P
7. Work Breakdown Structure	0	wholly dependant on contractors	A
Software Controls	**0.5**		
8. Defect Logs	1	limited ffragmented records	A
9. Defined Coding Standards	0	dependent on contractors	A
10. Peer Review of Code	0	dependent on contractors	P
11. Performance Baselines	0	none	A
12. Testing & Release Segmentation	1	limitted	P
13. Verification & Validation	1	limitted	P
Knowledge Sharing	**0.8**		
14. Documentation & Archiving (e.g. Knowledgebase)	1	basic archiving of files	A
15. Integrated Knowledgebase Index	0	none	A
16. Training Documentation	0	none	A
17. Training Sessions	0	none	P
18. Collaboration Systems	3	Using Slack, Trello, & Google Hangouts	A

Company: Forest

Factor	Level	Notes	A/P
Project Management	**1.0**		
1. Change Control Log	0	none	A
2. Charter	3	defined plan to expore market	A
3. Project Schedule	0	limited to cofounder	A
4. Project Schedule Control Process	0	limited to cofounder	A
5. Requirement Document (Traceability Matrix)	0	limited to cofounder	A
6. Scope & Change Control Process	3	defined scope, that is not advancing by design	P
7. Work Breakdown Structure	1	co-founder's documents	A
Software Controls	**0.8**		
8. Defect Logs	0	limited to cofounder	A
9. Defined Coding Standards	0	limited to cofounder	A
10. Peer Review of Code	0	limited to cofounder	P
11. Performance Baselines	0	limited to cofounder	A
12. Testing & Release Segmentation	2	known expectation and testing of performance	P
13. Verification & Validation	3	testing Ap and perfromance model	P
Knowledge Sharing	**0.5**		
14. Documentation & Archiving (e.g. Knowledgebase)		retention of key information of model, technoilogy, perfromance, and market repsonse	A
15. Integrated Knowledgebase Index	0	none	A
16. Training Documentation	0	none	A
17. Training Sessions	0	none	P
18. Collaboration Systems	2	informal be highly integrated communication between cofounders and advisors	A

Company: Gold

Factor	Level	Notes	A/P
Project Management	**6.3**		
1. Change Control Log	1	no specific tracking, but could track records	A
2. Charter	6	#1, #5, and #5 part of same living document	A
3. Project Schedule	8	Define process to create and update	A
4. Project Schedule Control Process	8	Use Harbor Software + Excel + Dropbox + daily calls	A
5. Requirement Document (Traceability Matrix)	7	#1, #5, and #5 part of same living document	A
6. Scope & Change Control Process	7	Well defined process. Saclable w/growth?	P
7. Work Breakdown Structure	7	#1, #5, and #5 part of same living document	A
Software Controls	**1.7**		
8. Defect Logs	2	fix and move on. Retain records, but not referenced.	A
9. Defined Coding Standards	2	Rely on professionalism. Some tools enforce continuity	A
10. Peer Review of Code	2	Pair programming. Tribal. Not defined.	P
11. Performance Baselines	0	no tracking of current or past records	A
12. Testing & Release Segmentation	2	Testing based on project documents, founder knowledge, and customers	P
13. Verification & Validation	2	Testing based on project documents, founder knowledge, and customers	P
Knowledge Sharing	**0.6**		
14. Documentation & Archiving (e.g. Knowledgebase)	2	Tribal, document sharing on Dropbox, largely project related	A
15. Integrated Knowledgebase Index	1	Knowledgebase for projects only	A
16. Training Documentation	0	none	A
17. Training Sessions	0	none	P
18. Collaboration Systems	0	none	A

Company: Honey

Factor	*Level*	*Notes*	*A/P*
Project Management	**0.6**		
1. Change Control Log	1	complete dependency on CTO	A
2. Charter	1	little documentionm	A
3. Project Schedule	1	CTO has a "metal plan"	A
4. Project Schedule Control Process	0	none	A
5. Requirement Document (Traceability Matrix)	0	none, all CCTO	A
6. Scope & Change Control Process	0	none, other than CTO	P
7. Work Breakdown Structure	1	CTO conveys needs to contractors	A
Software Controls	**1.0**		
8. Defect Logs	0	none	A
9. Defined Coding Standards	0	none, dependent on CTO	A
10. Peer Review of Code	3	CTO review contractors' work'	P
11. Performance Baselines	0	none	A
12. Testing & Release Segmentation	1	dependennt on CTO	P
13. Verification & Validation	2	depemndent on Founder and CTO	P
Knowledge Sharing	**0.6**		
14. Documentation & Archiving (e.g. Knowledgebase)	0	none	A
15. Integrated Knowledgebase Index	0	none	A
16. Training Documentation	0	none	A
17. Training Sessions	0	none	P
18. Collaboration Systems	3	highly interative using common tools	A

Company: Ivory

Factor	*Level*	*Notes*	*A/P*
Project Management	**1.0**		
1. Change Control Log	1	limited history could be compiled	A
2. Charter	3	Founders draft project docs for contract company	A
3. Project Schedule	0	rely enterly on contracted company	A
4. Project Schedule Control Process	0	rely enterly on contracted company	A
5. Requirement Document (Traceability Matrix)	0	rely enterly on contracted company	A
6. Scope & Change Control Process	3		P
7. Work Breakdown Structure	0	rely enterly on contracted company	A
Software Controls	**1.3**		
8. Defect Logs	0	rely enterly on contracted company	A
9. Defined Coding Standards	1	new CTO beginning to gain oversight of work	A
10. Peer Review of Code	1	new CTO beginning to gain oversight of work	P
11. Performance Baselines	0	rely enterly on contracted company	A
12. Testing & Release Segmentation	4	Basis testing processes, along with individuals dependenies	P
13. Verification & Validation	2	Basis processes. But responsibility largely on founders	P
Knowledge Sharing	**1.4**		
14. Documentation & Archiving (e.g. Knowledgebase)	2	Efforts are underway	A
15. Integrated Knowledgebase Index	0		A
16. Training Documentation	2	Newer marketing people are now active at doumenting the product	A
17. Training Sessions	2	2 marketing people are preparing for new hires.	P
18. Collaboration Systems	1	Effort are just beginning	A

Company: Jade

Factor	*Level*	*Notes*	*A/P*
Project Management	**3.1**		
1. Change Control Log	0		A
2. Charter	5	started in tradition project managemment. Attempted to convert to Agile	A
3. Project Schedule	3	reliance on outsourced develpoment	A
4. Project Schedule Control Process	2	reliance on outsourced develpoment	A
5. Requirement Document (Traceability Matrix)	4	reasonable dettailed product requirements, but only at a highly level.	A
6. Scope & Change Control Process	4	Controlled to a functional protoype, that may have had the potential to scale with growth.	P
7. Work Breakdown Structure	4	reasonable detailed product requirements, but only at a highly level.	A
Software Controls	**1.5**		
8. Defect Logs	0	none	A
9. Defined Coding Standards	0	none	A
10. Peer Review of Code	0	none	P
11. Performance Baselines	2	Definition of expecations defined	A
12. Testing & Release Segmentation	4	no testing systems, but defined	P
13. Verification & Validation	3	no testing systems, but but reviewed	P
Knowledge Sharing	**1.6**		
14. Documentation & Archiving (e.g. Knowledgebase)	1	Project communication only	A
15. Integrated Knowledgebase Index	2	attempts at documenation, but limited results	A
16. Training Documentation	2	attempts at documenation, but limited results	A
17. Training Sessions	2	Business to Client product dictated training for all	P
18. Collaboration Systems	1	outsourced work not collaborative	A

Company: Khaki

Factor	*Level*	*Notes*	*A/P*
Project Management	**2.7**		
1. Change Control Log	3	Use of "Sketch & Zeplin" software"	A
2. Charter	4	Use of "Sketch & Zeplin" software"	A
3. Project Schedule	2	Agile, with some Scrum processes	A
4. Project Schedule Control Process	3	Generally 2 week sprints. But inconsistent	A
5. Requirement Document (Traceability Matrix)	2	Utilize "Zeplin" software. But, even with collaboration/dev software, engineer's detail is not surfaced.	A
6. Scope & Change Control Process	3	Process advancing using "Zeplin" software	P
7. Work Breakdown Structure	2	Utilize "Zeplin" software. But, even with collaboration/dev software, engineer's detail is not surfaced.	A
Software Controls	**2.0**		
8. Defect Logs	3	Defects tracked. Detail and intergration is limited'	A
9. Defined Coding Standards	1	Expected professional standards of each engineers. But, standards are not documented.	A
10. Peer Review of Code	2	Expected that engineers reviews each other's code. But not well defined and not verified.	P
11. Performance Baselines	0	data not racked	A
12. Testing & Release Segmentation	2	limited detail manintained in "Zeplin."	P
13. Verification & Validation	4	Full team involvement in testing verification	P
Knowledge Sharing	**1.6**		
14. Documentation & Archiving (e.g. Knowledgebase)	3	Integrated use of "Sketch & Zeplin" software"	A
15. Integrated Knowledgebase Index	2	limited indexing and searching within "Zeplin"	A
16. Training Documentation	0	none	A
17. Training Sessions	0	none	P
18. Collaboration Systems	3	Integrated use of "Sketch & Zeplin" software"	A

Company: Lime

Factor	Level	Notes	A/P
Project Management	**1.0**		
1. Change Control Log	0	none	A
2. Charter	2	mostly verbal between cofounders	A
3. Project Schedule	1	desired to be Agile, but largely ad hoc	A
4. Project Schedule Control Process	0	none	A
5. Requirement Document (Traceability Matrix)	0	none, division of responsibility between cofounders	A
6. Scope & Change Control Process	3	verbal between cofounders	P
7. Work Breakdown Structure	1	limited, division of responsibility between cofounders	A
Software Controls	**1.8**		
8. Defect Logs	3	2 engineers fully document, but with limited sharing and not verified	A
9. Defined Coding Standards	0	none	A
10. Peer Review of Code	0	none	P
11. Performance Baselines	3	automated testing, with ongoing concern for usibility	A
12. Testing & Release Segmentation	3	testing process with division of labor	P
13. Verification & Validation	2	cofounder check, but not well formalized	P
Knowledge Sharing	**3.2**		
14. Documentation & Archiving (e.g. Knowledgebase)	4	"Slack" comboined with Google Docs	A
15. Integrated Knowledgebase Index	3	references withing Slack topics	A
16. Training Documentation	4	training docs on product and engineering. Statred as UX review and for potential patent	A
17. Training Sessions	0	none	P
18. Collaboration Systems	5	Using Slack	A

Company: Maroon

Factor	Level	Notes	A/P
Project Management	**3.6**		
1. Change Control Log	3	Archive data in "Jira" software	A
2. Charter	5	Utilize "Jira" software	A
3. Project Schedule	5	1 week (Scrum) sprints. Utilize "Jira" software	A
4. Project Schedule Control Process	5	Agile (Scrum) development. Utilize "Jira" software	A
5. Requirement Document (Traceability Matrix)	2	Some detail within Jira. Primarily assumes founder as Product Owner. However,upon go-live, there may be a risk engineers' interpretation and discretion to adapt (revise).	A
6. Scope & Change Control Process	3	Founder is the control point with as the "Jira" software tool. However, process is not scalable.	P
7. Work Breakdown Structure	2	Verbal scrums are not well documented. Dependent on engineers' discretion.	A
Software Controls	**2.2**		
8. Defect Logs	2	Largely at hoc at this point, although some functionality in "Jira." Howe	A
9. Defined Coding Standards	1	newer staff with few controls	A
10. Peer Review of Code	1	newer staff with few controls	P
11. Performance Baselines	0	not at this time	A
12. Testing & Release Segmentation	4	Draft processes not yet fullly implemted	P
13. Verification & Validation	5	Sprint Reviews. Founder may not persist as sole Product Owner.	P
Knowledge Sharing	**2.6**		
14. Documentation & Archiving (e.g. Knowledgebase)	5	Basecamp is being well utilized	A
15. Integrated Knowledgebase Index	4	Indexing and searching within Basecamp	A
16. Training Documentation	0	none	A
17. Training Sessions	0	none	P
18. Collaboration Systems	4	While Basecamp (and Jira) are used.	A

Company: Navy

Factor	*Level*	*Notes*	*A/P*
Project Management	**2.9**		
1. Change Control Log	1	no, but past sprints documented	A
2. Charter	4	User stories	A
3. Project Schedule	3	Agile, not waterfall	A
4. Project Schedule Control Process	4	Defined Sprints, but limited advance planning	A
5. Requirement Document (Traceability Matrix)	2	limited	A
6. Scope & Change Control Process	2	limited	P
7. Work Breakdown Structure	4	Scrum user stories, documneted in "Joblet"	A
Software Controls	**2.2**		
8. Defect Logs	1	limited via recretation of some docuements	A
9. Defined Coding Standards	0	none	A
10. Peer Review of Code	0	dependent on contractors	P
11. Performance Baselines	5	good overall analytics, but very limited ability to segment functions for release and testing	A
12. Testing & Release Segmentation	4	QA team fiunction for all core staff	P
13. Verification & Validation	3	only within the scope of a cursory sprint revieew	P
Knowledge Sharing	**0.6**		
14. Documentation & Archiving (e.g. Knowledgebase)	2	Utilize Google Docs	A
15. Integrated Knowledgebase Index	0	none	A
16. Training Documentation	0	none	A
17. Training Sessions	0	none	P
18. Collaboration Systems	1	very limited	A

Company: Olive

Factor	*Level*	*Notes*	*A/P*
Project Management	**6.7**		
1. Change Control Log	1	prototype docuemented, but record management	A
2. Charter	8	highly documented for outsourcing	A
3. Project Schedule	8		A
4. Project Schedule Control Process	8		A
5. Requirement Document (Traceability Matrix)	8		A
6. Scope & Change Control Process	7	Highly defined, but dependant on Founder. Scalable?	P
7. Work Breakdown Structure	7		A
Software Controls	**1.5**		
8. Defect Logs	1	no logs, but have communicaation records	A
9. Defined Coding Standards	0	none	A
10. Peer Review of Code	0	none	P
11. Performance Baselines	2	Definition of expecations defined, but no plans to maintain	A
12. Testing & Release Segmentation	4	no testing systems, but defined prototype & customer testing	P
13. Verification & Validation	2	no testing systems, but documented functions expectations	P
Knowledge Sharing	**1.3**		
14. Documentation & Archiving (e.g. Knowledgebase)	1	Project communication only	A
15. Integrated Knowledgebase Index	0	none	A
16. Training Documentation	2	documents for customers, not develpoers	A
17. Training Sessions	2	plans with customers, not develpoers	P
18. Collaboration Systems	0	none	A

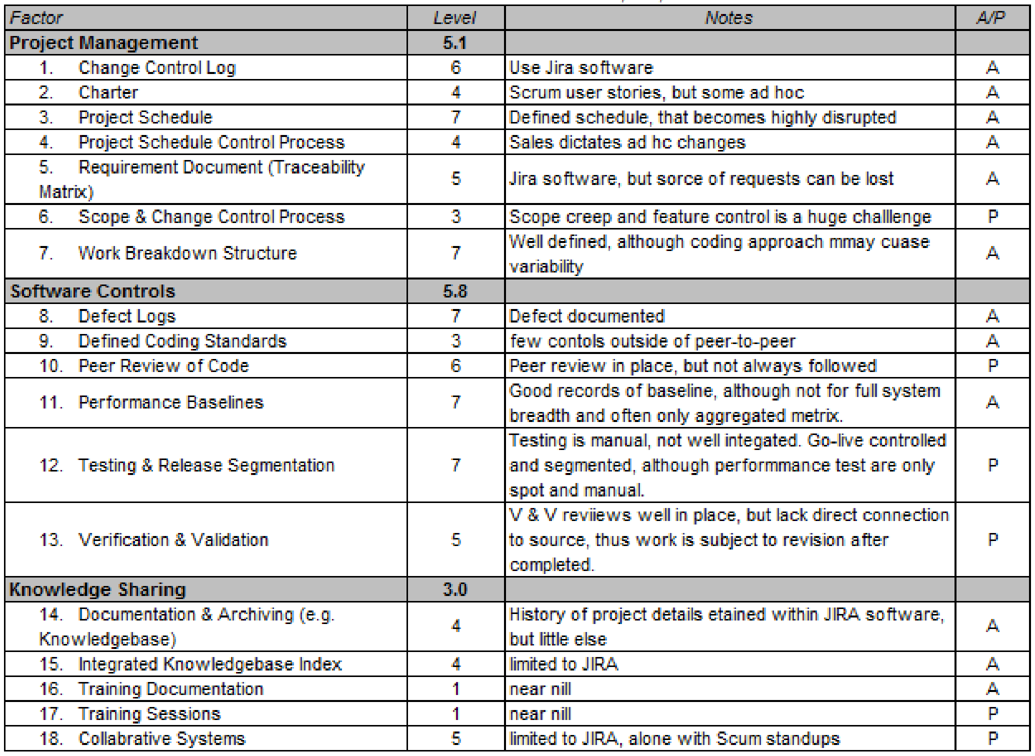
Company: Peach

Factor	Level	Notes	A/P
Project Management	**5.4**		
1. Change Control Log	4	Utilize "Jira" for archiving	A
2. Charter	6	All projects with harters, but absent of some traditional Chater components	A
3. Project Schedule	6	Projects managed within Jira and Portfolio in "Project" Gantt charts	A
4. Project Schedule Control Process	6	Sprint scheduling in place but with acknowledged problems	A
5. Requirement Document (Traceability Matrix)	3	Partially documented. But much origin information lost in user stories to sprints	A
6. Scope & Change Control Process	7	Well contriolled. Somewhat limited by other processes	P
7. Work Breakdown Structure	6	Well controlled. Challenge in individual engineers interpretation of work and need for greater support/oversight	A
Software Controls	**5.2**		
8. Defect Logs	5	Well documented, but not intergrated into proactive improvements	A
9. Defined Coding Standards	4	Stadards said to exist. Not varified and question as possible over rating	A
10. Peer Review of Code	5	Appears to occur within each dev team. But certainly not a defined process	P
11. Performance Baselines	5	utilize monitor tools and services. But data appears isolated from proactive use	A
12. Testing & Release Segmentation	6	Clarity of advancing processes	P
13. Verification & Validation	6	Clarity of advancing processes	P
Knowledge Sharing	**4.0**		
14. Documentation & Archiving (e.g. Knowledgebase)	5	Appears to be a wealth of documentation	A
15. Integrated Knowledgebase Index	3	Other than inherant tools in "Jira," docuementation seem siloed and not integrated into processes	A
16. Training Documentation	5	Documents exists. Completeness and integration to processes are questionable	A
17. Training Sessions	4	Considerable efforts to training, yet it is largely ad hoc	P
18. Collaboration Systems	3		A

Company: Quartz

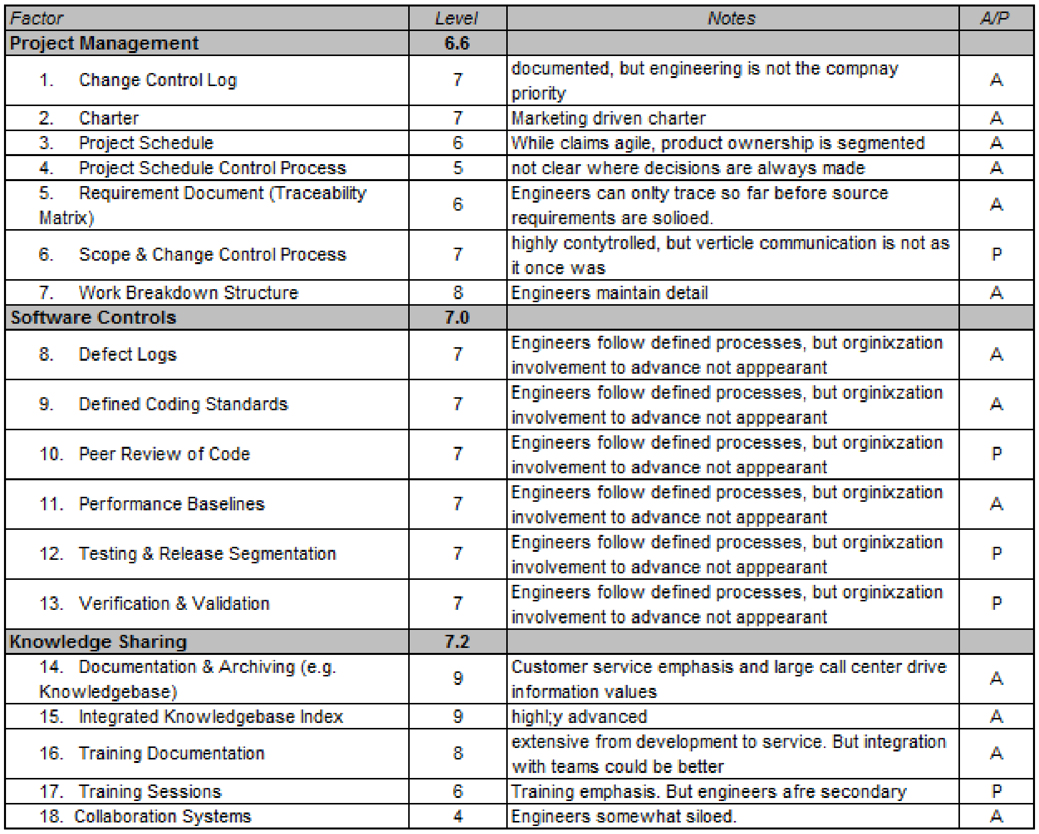
Factor	Level	Notes	A/P
Project Management	**8.1**		
1. Change Control Log	6	no project manager	A
2. Charter	9	User stories	A
3. Project Schedule	7	Agile, not waterfall	A
4. Project Schedule Control Process	8	No project manager, but highly controlled	A
5. Requirement Document (Traceability Matrix)	9	Scrum user stories versus	A
6. Scope & Change Control Process	9	Super tight process.	P
7. Work Breakdown Structure	9	Scrum user stories versus	A
Software Controls	**7.2**		
8. Defect Logs	7	Log gereates back into review for refactoring	A
9. Defined Coding Standards	8	Minimalist goals for full-stack team	A
10. Peer Review of Code	8	Full-stack team oversees each-others work	P
11. Performance Baselines	5	exists, but not a priority and not highly intergrated. Mmore driven by low everhead	A
12. Testing & Release Segmentation	7	Goal for continuos reaklease with individual segmentation, but packaging still occurs	P
13. Verification & Validation	8	emphasis on avoiding product feature creep, refactoring, and purging leagacy code and legacy features, as much as checking new features.	P
Knowledge Sharing	**7.0**		
14. Documentation & Archiving (e.g. Knowledgebase)	8	utilize collabortive software, difficult to confirm full depth of utilization	A
15. Integrated Knowledgebase Index	7	utilize collabortive software, difficult to confirm full depth of utilization	A
16. Training Documentation	6	documents for customers, not develpoers	A
17. Training Sessions	5	Daily Stand-up and real-time needs	P
18. Collaboration Systems	9	Collaboration is near reall-time throughout the company	A

Company: Rose



Factor	Level	Notes	A/P
Project Management	**5.1**		
1. Change Control Log	6	Use Jira software	A
2. Charter	4	Scrum user stories, but some ad hoc	A
3. Project Schedule	7	Defined schedule, that becomes highly disrupted	A
4. Project Schedule Control Process	4	Sales dictates ad hc changes	A
5. Requirement Document (Traceability Matrix)	5	Jira software, but sorce of requests can be lost	A
6. Scope & Change Control Process	3	Scope creep and feature control is a huge challlenge	P
7. Work Breakdown Structure	7	Well defined, although coding approach mmay cuase variability	A
Software Controls	**5.8**		
8. Defect Logs	7	Defect documented	A
9. Defined Coding Standards	3	few contols outside of peer-to-peer	A
10. Peer Review of Code	6	Peer review in place, but not always followed	P
11. Performance Baselines	7	Good records of baseline, although not for full system breadth and often only aggregated metrix.	A
12. Testing & Release Segmentation	7	Testing is manual, not well integated. Go-live controlled and segmented, although performmance test are only spot and manual.	P
13. Verification & Validation	5	V & V reviiews well in place, but lack direct connection to source, thus work is subject to revision after completed.	P
Knowledge Sharing	**3.0**		
14. Documentation & Archiving (e.g. Knowledgebase)	4	History of project details etained within JIRA software, but little else	A
15. Integrated Knowledgebase Index	4	limited to JIRA	A
16. Training Documentation	1	near nill	A
17. Training Sessions	1	near nill	P
18. Collabrative Systems	5	limited to JIRA, alone with Scum standups	P

Company: Sienna



Factor	Level	Notes	A/P
Project Management	**6.6**		
1. Change Control Log	7	documented, but engineering is not the compnay priority	A
2. Charter	7	Marketing driven charter	A
3. Project Schedule	6	While claims agile, product ownership is segmented	A
4. Project Schedule Control Process	5	not clear where decisions are always made	A
5. Requirement Document (Traceability Matrix)	6	Engineers can onlty trace so far before source requirements are solioed.	A
6. Scope & Change Control Process	7	highly contytrolled, but verticle communication is not as it once was	P
7. Work Breakdown Structure	8	Engineers maintain detail	A
Software Controls	**7.0**		
8. Defect Logs	7	Engineers follow defined processes, but orginixzation involvement to advance not apppearant	A
9. Defined Coding Standards	7	Engineers follow defined processes, but orginixzation involvement to advance not apppearant	A
10. Peer Review of Code	7	Engineers follow defined processes, but orginixzation involvement to advance not apppearant	P
11. Performance Baselines	7	Engineers follow defined processes, but orginixzation involvement to advance not apppearant	A
12. Testing & Release Segmentation	7	Engineers follow defined processes, but orginixzation involvement to advance not apppearant	P
13. Verification & Validation	7	Engineers follow defined processes, but orginixzation involvement to advance not apppearant	P
Knowledge Sharing	**7.2**		
14. Documentation & Archiving (e.g. Knowledgebase)	9	Customer service emphasis and large call center drive information values	A
15. Integrated Knowledgebase Index	9	highl;y advanced	A
16. Training Documentation	8	extensive from development to service. But integration with teams could be better	A
17. Training Sessions	6	Training emphasis. But engineers afre secondary	P
18. Collaboration Systems	4	Engineers somewhat siloed.	A

Company: Tan

Factor	*Level*	*Notes*	*A/P*
Project Management	**6.9**		
1. Change Control Log	8	Scrum environmrrnt using "Jira"	A
2. Charter	8	Scrum environmrrnt using "Jira"	A
3. Project Schedule	6	Projects follow Scrum. Portfolio is a criticism	A
4. Project Schedule Control Process	5	Periodic disrunption of process due to top-down changes in priopritie	A
5. Requirement Document (Traceability Matrix)	8	highly documentd, but not always clear who drive requirements (execs, clients, sales)	A
6. Scope & Change Control Process	5	Periodic disrunption of process due to top-down changes in priopritie	P
7. Work Breakdown Structure	8	highly documented	A
Software Controls	**7.7**		
8. Defect Logs	7	highly documented	A
9. Defined Coding Standards	7	defined, but can be varible managed	A
10. Peer Review of Code	7	turnover noted as disrupting code quality	P
11. Performance Baselines	9	monitiring is defined, largely automated, and quantitiative	A
12. Testing & Release Segmentation	9	process are defined, largely automated, and quantitiative	P
13. Verification & Validation	7	highly documented	P
Knowledge Sharing	**6.8**		
14. Documentation & Archiving (e.g. Knowledgebase)	8	Extensive docuements, but the amount of sediment may be causing bloating of data and dimishing use.	A
15. Integrated Knowledgebase Index	8	Controlled, but dilligence is a challlenge	A
16. Training Documentation	5	Increasing tendency to relay on verbal training, avoiding documents	A
17. Training Sessions	6	Comments indicate diminishing as of late	P
18. Collaboration Systems	7	Use of "Slack"	A