Chapter 10
Motivating Software Engineers Working in Virtual Teams Across the Globe

Sarah Beecham

Abstract The motivation of software engineers affects the quality of the software they produce. Motivation can be viewed in terms of needs. The key need for a software engineer is to 'identify with their task' which requires being given a task that is challenging and understanding the purpose and significance of the task in relation to the complete system being developed. Software engineers' needs are complex – they also require regular feedback, trust, appreciation, rewards, a career path, and sustainable working hours. Furthermore, amongst other fixed environmental factors, these motivators require sensitive tuning in line with a software engineer's personality and career stage. Creating this personality-job fit is not easy in a co-located environment, so how can project managers motivate teams of individuals distributed across the globe?

This chapter reflects on some of the motivational issues that managers of virtual teams may encounter. Some background theory is presented for a deeper understanding of how to manage team motivation. Recommendations are drawn from a case study where issues raised by practitioners working in virtual teams serve to highlight and magnify known motivational issues. Project managers play an important part in software engineer motivation. If they can create a working environment that motivates individuals in the team, they will find that team members are more likely to turn up to work, are less likely to look elsewhere for employment, will work harder to meet deadlines, will take more pride in their work, and will share their knowledge, concerns, and ideas for innovation.

S. Beecham (✉)
Department of Computer Science and Information Systems, Lero – The Irish Software Engineering Centre, University of Limerick, Limerick, Ireland
e-mail: sarah.beecham@lero.ie

G. Ruhe and C. Wohlin (eds.), *Software Project Management in a Changing World*,
DOI 10.1007/978-3-642-55035-5_10,

10.1 Introduction

This chapter explores how to motivate a software engineer working in a virtual team. To answer this question, some general motivation theories are introduced that are relevant to software engineer motivation in a global setting. Since motivating practitioners is likely to lead to improved quality of the software product (McConnell 1996; Verner et al. 2014) and development of software is increasingly a global effort (Chaps. 9 and 12), examining how to motivate software engineers working in globally distributed teams should be of interest to software development practitioners.

Although motivation is a well-researched area, existing theories have not kept pace with today's software engineering climate. The twenty-first century has seen radical changes in both the working environment and the demands made on the people employed to undertake the work. The move towards developing software globally has been rapid, requiring engineers to work in teams around the clock, with mixed values and cultural styles. It is clear that global software development (GSD) is here to stay, despite the risk it poses to motivation (Frey and Osterloh 2002).

GSD can require software engineers to work on sites hundreds or even thousands of miles away from their virtual team mates, where members of the same team may never meet in person. Engineers may also have to cope with time zone differences between sites that constrain the ability to communicate in real time and can lead to delays and frustration, or to working antisocial hours. Other barriers emerge such as cultural and linguistic differences between team members who may need to discuss complex technical issues (Noll et al. 2010; Shah et al. 2012; Monasor et al. 2013). Add to this mix a backdrop of tight deadlines, centralized budgetary controls, and requirements for high-quality software. It is clear from this short summary that GSD places unique and extreme demands on the engineer. While existing *theories* of motivation do not account for the complexities introduced when working in distributed teams, fortunately we have a wealth of empirical research we can draw on to help identify how, where and what motivation means in this context. This chapter draws on theories of motivation and maps these to empirical findings of work undertaken in GSD.

The growth of agile practices (see Chap. 11), shared responsibilities, and flat organisational hierarchies have all contributed to our understanding of how to foster motivation. For example, Beecham et al. (2008) and França et al. (2012) found that agile principles generally meet software engineers' motivational needs, with a few exceptions. However, recent work also suggests that too much freedom and ad hoc arrangements can work against software engineer motivation (Fernández-Sanz and Misra 2011). This seems to contradict the open source and inner source software development paradigms that are gaining in popularity and impetus (Chaps. 13 and 14 relate). The authors of Chap. 14 discuss how software engineers, when adopting OSS practices, are likely to be the recipients of many types of rewards—shown in this chapter to be important intrinsic motivators. In OSS, self-selecting volunteers come together to create their own communities and expend effort to produce high-

quality and useful software. It is not surprising that researchers look to these environments to learn about motivation since contributors appear to be motivated by some internal impetus not necessarily associated with financial reward (Ye and Kishida 2003; Roberts et al. 2004; Riehle 2007).

Having read through this chapter, the reader should come away with a basic understanding of some organisational and motivation theories (Sect. 10.2), a feel for whether software engineers are likely to have distinct personalities of their own (Sect. 10.3), and an understanding of how to motivate software engineers in a virtual team setting through a case study example presented in Sect. 10.4. Section 10.5 maps Global Teaming practices to motivational factors. Section 10.6 discusses the case study example in the context of some motivation theory. Section 10.7 concludes the chapter with a summary of software engineer motivation in GSD.

10.2 Motivation Theory

There are techniques managers can apply that will motivate employees to work harder, and increase their commitment to the organisation. Creating the right conditions can also stimulate innovation (Frey and Osterloh 2002). However, perhaps the reason that there are well over 100 theories of motivation (Petri and Govern 2012), is that no one theory truly reflects how people are motivated. At best, each theory provides new insights into what is a highly complex area (da Silva and França 2012).

Classic motivation theories can be broadly classified as either 'content' or 'process' theories. Content theories include Maslow's hierarchy of needs (1954), Herzberg et al.'s two-factor theory (1959), and McClelland's needs theory (1961). These content theories assume a "complex interaction between internal and external factors" and explore "the circumstances in which individuals respond to different types of internal and external stimuli" (Bassett-Jones and Lloyd 2005). Process theory, on the other hand, describes motivation as "a sequence or process of related activities" (Hall et al. 2009). Exponents of process theories include Adam's (1963) equity theory, Vroom's (1964) expectancy theory, Skinner's (1976) stimulus-response theory, Locke et al.'s (1968) goal setting theory, and Hackman and Oldham (1976) and Couger and Zawacki's (1980) task design theories. This chapter focuses on the process theories relating to work design and job characteristics (Hackman and Oldman 1976; Couger and Zawacki 1980), where task variables are explored in a GSD context. Also, Herzberg's (1959) two-factor content theory (motivation-hygiene theory) is discussed since the external environment (a hygiene factor) is an integral part of GSD.

Motivation theories try to explain the conscious or unconscious decisions people make to expend effort or energy on a particular activity. Handy (1993) encapsulates many of these process theories in his 'motivation calculus,' which expresses

A software engineer with a strong need to learn is given a problem-solving task that promises to expand his or her knowledge and skill-set. The ***effort*** *the engineer is prepared to expend on the task will depend on the degree to which he or she*

*a) believes that solving the problem will lead to increased knowledge and skills (****expectancy*** *that the need will be satisfied) and*

b) having completed the task, finds that the ***result****ing increased knowledge and skills satisfy the learning needs (****instrumentality*** *of the satisfied need).*

Fig. 10.1 Example scenario of motivational elements adapted from Handy (1993)

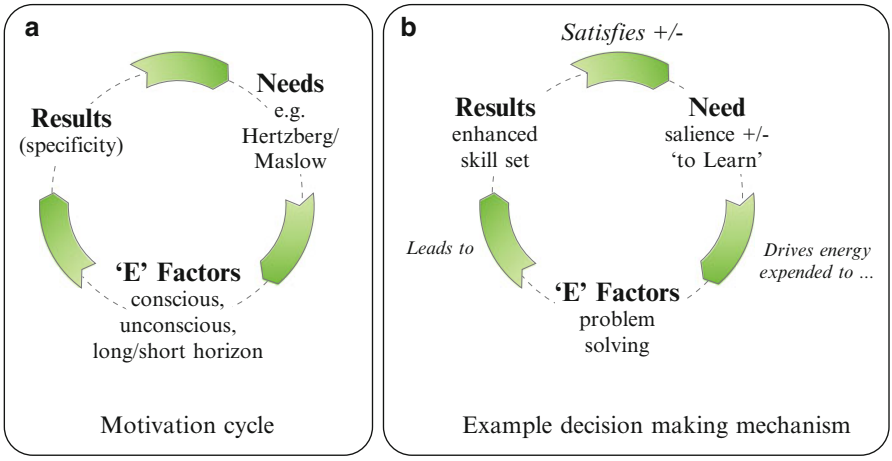

Fig. 10.2 Motivation calculus inspired by Handy (1993) (**a**) Motivation cycle (**b**) Example decision-making mechanism

motivation in terms of *needs*, *results* that satisfy needs, and *effort*[1] expended to achieve those results that satisfy the needs. The calculus demonstrates that the level of effort a person is willing to expend on a task is linked to how the person feels that effort will be rewarded; if the person values the expected reward, they will increase the effort accordingly. If the result fulfills a need and the experience is positive, it will feed into the person wanting to expend energy again on similar need-related tasks, and so the cycle continues. The re-enforcement cycle is explained in the example in Fig. 10.1 and summarised in Fig. 10.2.

Figure 10.2 indicates that it is tapping into the strength (or salience) of the need that will determine how to motivate the engineer. For example, looking at Fig. 10.2 (b),

[1] Effort is just one of the E's in the calculus; other E's include energy, excitement, enthusiasm, emotion, expenditure of time, expenditure of money, and expenditure of passion.

motivation can break down if the problem-solving task does not lead to enhancing the skill set as this will not satisfy the need to learn. Alternatively, the motivation cycle is broken if an enhanced skill set does not satisfy the need to learn. The need and how to satisfy that *need* will vary from person to person. Individual characteristics play an integral part in motivation theories.

It is the mapping of the individual need to the type of job that forms the basis for the 'job characteristics theory' (JCT) also discussed in this chapter. Section 10.3 considers the individual personality traits and characteristics of the software engineer—this is perhaps the Holy Grail, as if we can find common traits in people attracted to the software engineering profession, it will ease the task of motivating engineers.

10.2.1 Motivation as a Social Process

While Handy's approach is useful in capturing motivation, understanding can be broadened by viewing motivation as part of a *social* process. This complementary social process dimension relates strongly to the context of software engineers who must work together in teams and interact. Motivation as a social process defines how people join, remain part of, and perform adequately in a human organisation (Huczynski and Buchanan 1991). The global organisation is a social arrangement comprising members who are motivated to join, to stay, and to perform at acceptable levels. It is within a social context that teams working remotely are encouraged to work harder and more effectively. Some research suggests that social interaction itself can be motivating (Petri and Govern 2012). Self-motivation is just one factor that drives an individual to join an organisation, to stay, and to perform at acceptable levels. The other characteristics are discussed in Sect. 10.3. The increase in the use of social media in GSD (see Chap. 16) reflects the growing need for members of distributed teams to collaborate via informal channels.

10.2.2 Rational-Economic Needs

Scientific management research, conducted in the 1940s, asserted that dividing work into structural units and offering monetary incentives would motivate individuals to increase their productivity. Frederick Taylor (1947), a key proponent of the scientific management movement, introduced the rational-economic needs concepts of motivation that he believed would lead to work being more satisfying and profitable. Taylor hypothesised that workers would be motivated by the high wages that they earned by working in the most efficient and productive way. Taylor was preoccupied with finding the most efficient methods and procedures for coordination and control of work, a goal shared with today's global software development managers. Key principles of Taylor's approach include the division

of labour, the sharing of responsibility between management and workers, knowledge sharing, and carrying out work in a prescribed way. It appears that Taylor anticipated the need for a work environment suited to automation. Taylor's vision can also be mapped to GSD and other product management approaches since dividing up processes into discrete, unambiguous pieces eases task allocation and sharing among developers working in virtual teams. Chapter 9 discusses how to allocate tasks in distributed and global development to maximize team cohesion and minimize coupling.

Initially, the effect of the introduced changes raised productivity and employee wages by 60 % (Huczynski and Buchanan 1991). However, despite increased output and monetary rewards, there was a strong reaction against scientific management methods from employees (Mullins 1993). Taylor's design of fragmented tasks was boring and called for a low level of skill. Requiring low levels of skill allowed the management to treat people as pure resources that could easily be replaced. In line with this, managers could reduce wages and ignore the psychological needs of the employees who had little opportunity to give feedback, experiment or make changes. These factors resulted in Taylorism in its strictest sense becoming obsolete with the term "scientific management" falling out of favour fairly soon after its introduction.

Paradoxically, even though we are aware that Taylor's methods do not work in the long term, today's managers of distributed teams seem to be reintroducing some of these practices. In terms of division of labour, one model is that testing gets outsourced to low-labour cost countries, design is undertaken in the onshore office, and coding is performed in satellite locations. Also, decision making can be the province of the centralized managers, where the needs of those working remotely are not necessarily represented in the organisational process.

In summary, Taylor's approach reflected the spirit of the times in the 1940s, a time of industrial reorganisation, new forms of technology, and the emergence of large complex organisations. We now find ourselves again in a phase of industrial reorganisation, with even more complex work structures in the form of globally distributed software development. Work patterns have changed largely due to new forms of technology, especially concerning methods of communication. Perhaps for this reason elements of Taylor's approach have not died out, a sentiment shared by academics who proclaimed in the early 1990s that "Taylorism is alive today" (Huczynski and Buchanan 1991). More recently, researchers Kennedy and Nur (2012) note that prescriptive practices associated with Taylorism continue to rise. Engineering work is now highly controlled by procedures and the increased need for senior management approval. This has implications for motivation as "so long as the Taylorist paradigm persists, the organisational aspiration to create a high commitment culture is likely to prove elusive" (Bassett-Jones and Lloyd 2005). However, it could be that it is the high employee turnover and subsequent low knowledge retention that drive the need for regimented processes (Kennedy and Nur 2012).

10.2.3 Motivation Theories for Software Engineering

Given that motivation is important, and that looking at people as machines that will do repeated work ad infinitum (even for good pay) is not a panacea, we now move away from the scientific management view and go to an approach where it is the people within the organisation that matter most.

Organisational theory records several approaches and models of motivation, many of which have been applied in software engineering research (Hall et al. 2009; Sharp et al. 2009). This chapter focusses on two theories that stand out amongst this group as being particularly relevant to motivating people who work in global software development teams: firstly, the content theory expounded by Herzberg et al. (1959)—the *two-factor theory*- and secondly, the process theory according to Hackman and Oldham's (1974) *job characteristics theory* (JCT) and adapted by Couger and Zawacki (1980) for software engineers. A brief definition of these theories is given next, along with why they might, even after 40 plus years, support the management of virtual teams.

Herzberg's two-factor theory: In 1959, Herzberg and his collaborators isolated two different factors that influenced motivation and satisfaction at work. One set of factors, classified as 'demotivators' or hygiene factors, are those that, if absent, can reduce motivation; these extrinsic factors are concerned with the work environment. However, to motivate employees to give their best, the focus must move to a different set of factors, classified as 'motivators' or intrinsic factors relating to the task itself.

The work of Herzberg is pertinent to global software development, since motivation (and demotivation) factors are viewed in terms of external influences and internal influences, even though the theory was developed over 50 years ago. However, there is some controversy as to how factors are classified, "largely because of the assertion that there was a weak correlation between financial reward and job satisfaction" (Bassett-Jones and Lloyd 2005). Herzberg classifies financial rewards as a hygiene factor, suggesting that inadequate financial reward can demotivate—and that beyond a limited threshold, money cannot motivate. Although classifying factors as either *hygiene* or *motivators* can appear contrived, it is helpful for the purpose of identifying how GSD factors may demotivate. Also, it is helpful as there might be some hygiene factors that are outside the control of the project manager. Of note is that demotivators are not the opposite of motivators; demotivators and motivators are distinct groups of factors.

Job characteristics theory: According to Beecham et al.'s review of the motivation literature (2008), Hackman and Oldham's (1974, 1976) job characteristics theory (JCT) is the most applied theory in software engineering. This process theory views the work itself as the main motivator, where given a set of personal needs, a person will only be motivated if these needs are matched by the job. The JCT model reflects the relationship between job characteristics, psychological states, and personal work outcomes. The JCT was extended by Couger and Zawacki (1980) to fit the software engineering context. The associated data collection tool, the Job

Diagnostics Survey (JDS), was developed to quantify an individual's person-job fit. Once quantified, motivation levels can be compared across individuals, across organisations, and across studies. As shown in the investigation into Herzberg's hygiene factors, global software development can involve environmental factors outside the control of the project manager. The importance, therefore, of the person-job fit seems particularly pertinent to our solution. The person-job fit for GSD is considered later in this chapter.

10.3 Characteristics of a Software Engineer

As noted in the Section on motivational theory (Sect. 10.2), there is no one-size-fits-all solution to motivating software engineers since motivation depends on individual needs and personality. Findings are derived from an in-depth systematic literature review of software engineer motivation (Beecham et al. 2008). A section of the motivation review was dedicated to studies that looked specifically at types of people attracted to the software engineering profession as opposed to what motivates them to stay in the profession, to do better work, etc. As a result, some characteristics appear similar to motivators, even though they came from a different strand of research. The original rationale for conducting the research into the characteristics was to assess whether software engineers are somehow different to professionals in other domains. Because, if there is no difference, we could argue that we do not need a separate study and model of motivation for software engineers; we could draw on the existing models and theories of motivating people in the workplace, some of which have been discussed in the previous section. On balance, 73 % of studies of software engineers indicated that software engineers do form a distinct identifiable occupational group (Beecham et al. 2008). This finding indicates that studying motivation for software engineers as a separate profession could benefit the managers of software engineers and researchers.

Figure 10.3 shows how controls and mediators will shape a software engineer's characteristics.

A systematic literature review of 92 separate studies relating to software engineer motivation (Beecham et al. 2008) observed that a software engineer's characteristics are formed by two factors: their internal make-up (termed 'control factors') and external factors (termed 'moderators'). Control factors define innate personality. Although personality will have an influence on the characteristics of a software engineer, defining personality types goes beyond the scope of this chapter. For readers interested in knowing more about this dimension, Chap. 4 gives an in-depth description of personality and how to assess different personality types. Moderators, on the other hand, are discussed briefly since they are understood to change the strength of certain characteristics.

The moderators identified in Beecham et al. (2008), listed in Table 10.1, have particular significance in a global, virtual team setting. Although they can be outside the control of the manager, they still need to be acknowledged as important

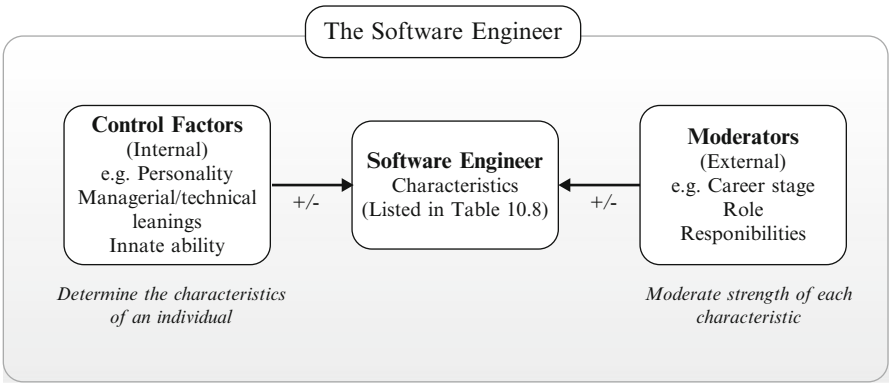

Fig. 10.3 Model of software engineer characteristics adapted from Beecham et al. (2008)

to the individual. For example, although an individual's career stage, age, and the state of the IT profession are fixed, they may all influence how to motivate the individual.

Also, the culture of the individual or given setting has been identified as problematic in many GSD studies, for example, Olson and Olson (2004), and is often labeled as a barrier to successful communication (Noll et al. 2010). However, managers can take advantage of a mixed cultural team for enhanced creativity, innovation, and holiday cover (Deshpande et al. 2010).

Promotion prospects are also seen as moderating a software engineer's characteristics. For example, Johnson et al. (2010) found a significant positive correlation between an employee's promotion focus and affective commitment. Promotion prospects are a poorly studied area in GSD research and may as a result be overlooked as an important motivator. Translating this moderator into a GSD context highlights the need for software engineers working remotely to have a clear career path and promotion opportunities. All factors listed in Table 10.1 are likely to moderate the strength of a software engineer's *characteristics*.

Of the many software engineer characteristics identified in the literature (considered in relation to GSD later in this chapter in Table 10.8), growth-oriented, introverted, and need for independence were the most cited. However, some characteristics contradict each other, such as 'introverted' with a low need for social interaction, and 'need to be sociable and identify with a group or organisation'. The view that software engineers are introverted reflects findings from the many studies coming from Couger and colleagues who measured the social needs strength of engineers (Couger and Zawacki 1980) in their Job Diagnostics Survey (for a full list of sources, see Beecham et al. 2008). This view is not universal, as seen in the body of more recent research that identified software engineers as sociable people (Beecham et al. 2008). Certainly the need for software engineers to communicate and relate to others is crucial in a GSD context. The new software engineer profile may therefore reflect the changing demands of the role.

Table 10.1 Software engineer moderators

1.	Career stage (age and experience)
2.	Culture (relating to national culture)
3.	Job type/role/occupational level
4.	State of IT profession (snapshot of evolutionary process)
5.	Type of organisation (e.g., promotion opportunities/rules)

Figure 10.4 shows the relationship between characteristics, controls and moderators (given in detail in Fig. 10.3), and motivators and outcomes. The level to which the needs (defined by a software engineer's characteristics) are met by the motivators will impact on tangible outcomes. For example, Hall et al. (2008a, b) found a positive correlation between software engineer motivation and employee turnover, and Verner et al. (2010) found a positive correlation between motivation and software engineering/management agreements on project success.

In summary, this section reflected on the characteristics of software engineers and whether they have any common characteristics. Understanding that engineers are likely to have some distinct traits should help managers to motivate their software development teams. The studies included in the motivation review (Beecham et al. 2008) did not consider the extra complexity of working in a distributed environment. The characteristics in a GSD context are considered in Sect. 10.6. The next section places all the 22 motivating factors identified in the review in a GSD context through an empirical mapping study.

10.4 Software Engineer Motivation in GSD—A Case Study

This section examines how software engineers' needs are likely to be affected by GSD. A case study conducted during 2010–2011 is used as an example of how virtual team behaviour may inhibit or strengthen software engineer motivation.

The case study is based on a GSD organisation that distributes its software development activities across several sites and countries. The organisation has its central office in Ireland and develops bespoke software for the financial services sector. It is a medium-sized organisation with approximately 200 employees. For reasons of confidentiality and anonymity, this organisation is referred to as "GSD Corp." GSD Corp offshore much of their development activities, but maintain a large team of practitioners in the central office who work mainly from 9 A.M. to 5 P.M., 5 days a week. This team develops their core product, manages the off-shore teams, and tests the bespoke software. The offshore teams are more focussed on requirements gathering and product deployment. Offshoring is undertaken for three key reasons: firstly, to extend their customer base; secondly, to work closely with their customers; and thirdly, to hire excellent technical talent in low-cost countries. All development comes under the organisational control of GSD Corp. The purpose of conducting the case study was to examine the processes used by GSD Corp to develop their software.

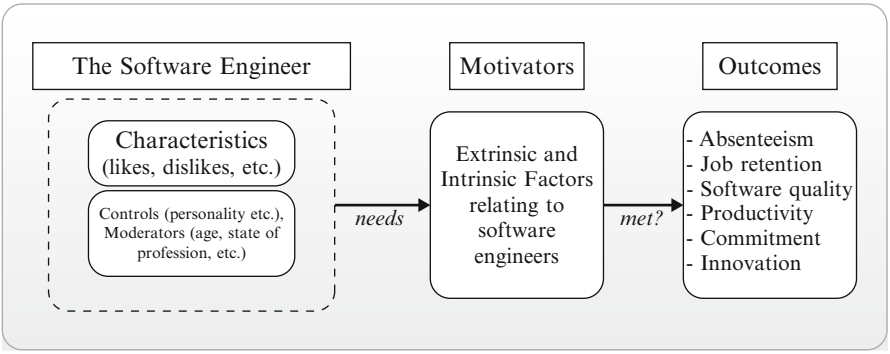

Fig. 10.4 Model of software engineer motivation adapted from Beecham et al. (2008)

Onshore and offshore opinions and lessons learned were solicited through a series of in-depth interviews that provided insights into where GSD processes might be improved. To gain a full picture of day-to-day work patterns, two projects were studied, one of which had just been completed, and the other was still in progress. Twenty-four employees performing various roles in the development process based in four different countries (Ireland, USA, South Africa, and Australia) were interviewed using the same set of semistructured questions. Each employee was interviewed for 1–2 hours, on a one-to-one, confidential basis. All interviewees worked in a distributed environment. Detailed notes were taken during the interviews, which were also recorded and later transcribed verbatim. A cross section of roles was interviewed in the sample, including technical and business consultants, quality assurance, project managers, project leads, solution architects, technical and business stream leads, and programme manager.

The detailed notes and analysis of interview transcripts presented a full picture of how GSD Corp operates across its several sites. The findings presented in this section are drawn from a subset of responses to direct and indirect interview questions that related to motivation. Direct questions included "how motivated were you in your project?", "was your project a success or highly challenged?", "how do you define project success?". Indirect questions tackled the problems and challenges the interviewees experienced in conducting their day-to-day tasks working in a virtual team, as well as discussing what excited them about their work. Using a qualitative, content analysis approach (Krippendorff 1980) similar to that used in Noll et al. (2011), responses were categorised according to whether they highlighted a challenge or a solution to a given problem, as well as advantages and disadvantages of working in virtual teams.

Table 10.2 Extrinsic motivators enhanced by virtual team practices

Extrinsic factor	Virtual team practice (drawn from case study)
Working in a successful organisation	Spreading the business into new markets was seen as a good business model. Operating in different countries was linked to organisational success.
Job security/stable environment	None of the employees interviewed felt insecure. They knew that good engineers where in short supply and did not feel that their jobs were under threat, despite the financial climate.

10.4.1 Case Study Results

The preliminary results shown in Tables 10.2, 10.3, 10.4, and 10.5 map case study findings (such as advantages and disadvantages of working in a virtual team) to motivational factors drawn from the literature (Beecham et al. 2008). Extrinsic factors are presented separately as it is likely that they will require a different management approach to the intrinsic motivators that are concerned with the job itself.

In the case study, two extrinsic factors were enabled by GSD: job security (no interviewee felt their job was under threat) and the need to work for a successful company (see Table 10.2). Some factors appear independent of whether employees work in a co-located or virtual environment, such as having a feedback mechanism in place. Factors that are likely to be equally difficult to achieve in virtual and co-located teams are not covered in these tables.

However, the case study did reveal several motivational factors that appear to be challenged as a result of working in a distributed team. Extrinsic factors (see Table 10.3) such as poor working conditions (disruptive) and a poor work-life balance (e.g., unpredictable working hours, extensive travel, and long commute) can demotivate software engineers. GSD Corp's poorly defined virtual team roles weakened practitioner empowerment and feeling of responsibility. Ambiguous roles and responsibilities can be problematic in a co-located environment, and working remotely magnifies the problem. For example, a team lead working at a remote customer site was undermined by senior management (working from a different country and time zone) who decided to discuss on-site matters directly with the customer (ignoring the team lead working in a predominantly customer-facing role). Not only was this undermining for the on-site team member, it caused confusion as to who is responsible for handling customer issues.

Considering issues directly associated with the job itself (shown in Tables 10.4 and 10.5), there were perceived differences in how employees were treated based on where they were located. For example, a practitioner working offshore reported that they sometimes missed out on training opportunities. Also, those based remotely felt they did not have the same promotion prospects as the onshore team since the remotely based senior management tended to see problems rather than when the employee was doing a good job and there were fewer options in a small organisation for internal promotion. Offshore teams felt that they worked longer

Table 10.3 Extrinsic motivation challenged by virtual team practices

Extrinsic factor	Virtual team practice (drawn from case study)
Rewards and incentives (e.g., scope for increased pay and benefits linked to performance)	Requires objective measurement, and as such is independent of location—however, making sure that rewards are given to each employee fairly across different locations may not be achievable, e.g., some remote workers were not able to take time off in lieu for working long hours and overtime.
Good management (senior management support, team-building, good communication).	Becomes even more important when working remotely—extra pressures, extra layer of complexity requires experienced and confident managers to deal with unforeseen problems. A recurring theme was that remote projects required experienced managers that can communicate well with both customers and all team.
Sense of belonging/supportive relationships	Difficult to feel supported when your counterpart might be sleeping during your core working hours. However, the organisation had a strong corporate culture, clearly communicated in all interviews.
Work/life balance (flexibility in work times, caring manager/employer, work location)	Extremely difficult to achieve, when there is a lot of travel, working away from home (and family), and keeping work hours down to core times seems impossible. It was rare to hear any reports of people working sustainable hours when working remotely.
Employee participation/involvement/working with others	Some experienced managers working remotely did not want to participate with the wider organisation; finding interference from higher management to be a negative influence. They tended to want to be left alone to sort out their customer facing issues. A fine balance needs to be struck between participation and a top-down style of management that imposes the processes.
Appropriate working conditions/environment/equipment/tools/physical space/quiet	Working conditions specially affected remote workers. For example, when working onsite with customers they often did not have any influence on where they work, or how and sometimes, when. They were not able to separate themselves from being on call to the customer: there was a tension between dealing with customer demands and their tangible deliverables.
Sufficient resources	Resources were scarce in terms of people (individuals were stretched to fill the gaps).

Table 10.4 Intrinsic motivation challenged by virtual team practices

Intrinsic factor	Virtual team practice (from case study)
Development needs addressed (e.g. training opportunities to widen skills)	Formal training, though offered centrally, would not always be extended to those employees working remotely. Many employees would have liked to attend training programmes that were only made known to them after they occurred, or when all places were filled. Also, working remotely, they were not able to take the time out for training, which was not built into the development schedule.
Technically challenging work	The technical work may be less challenging if task is highly specified, with reduced dependencies. However, balancing the many responsibilities and demands on time, and keeping both customer and management happy was very challenging.
Identify with the task (clear goals, how task fits in with whole)	Working in a distributed fashion in some cases resulted in developers not seeing the bigger picture and how their part of the work fitted in with the overall delivered product.
Employee participation/involvement/working with others	Members of the team may find it difficult to work with others if they are in different time zones. They can become disenfranchised or alienated. Difficult to make their opinion heard if not working physically together.
Career path (opportunity for advancement, promotion prospect, career planning)	There was a perceived lack of opportunity for advancement within the organisation—especially in an offshore role. Also in this SME, there were a limited number of roles available. Individual career plans to gain as much experience as possible to improve their marketability with external employers were met. Yet, even if employee work experience and skills were increased over time, they may leave the organisation to reap the benefits of increased skills.
Making a contribution/task significance (degree to which job impacts on others).	Working in a piecemeal fashion, e.g. just doing the coding, or part of the coding, prevented the developer from recognising how his/her part will make a difference. Some were uncertain as to whether software they are developing was used/implemented.
Recognition (for a high-quality, good job done	Universal recognition for a job well done is difficult to achieve when working remotely, where the main reason for making contact with head office might be to escalate a problem, or to check back when some action needs to be granted permission, or expenses need to be paid. If all is going well, then the practitioner 'doing a good job' may just be invisible.

(continued)

Table 10.4 (continued)

Intrinsic factor	Virtual team practice (from case study)
Trust/respect	Trust is a recognised problem in GSD and can cause barriers to the development. Engendering trust is difficult when teams may never have met face to face, may not share a common language, and may have different cultures. This however did not pose a problem in the case study with their strong corporate and friendly culture.
Equity	Fair treatment of all individuals working in virtual teams is difficult to achieve in GSD. For example, teams working in both the UK and the USA may feel that the employee in the other county is working less hours (e.g., a US employees developer may not be able to contact their European counterpart after 11 A.M. US time), also the US tend to take fewer days leave than their European counterparts.
Empowerment/responsibility	Responsibility by role is ambiguous—when roles are not well defined, it is difficult to know just how much authority you have to make changes and make decisions. Decision making is key to motivation (Handy 1993), therefore responsibilities need to be clear.

Table 10.5 Intrinsic motivation enhanced by virtual team practices

Intrinsic factor	Virtual team practice (from case study)
Variety of work (e.g., making good use of skills, being stretched)	The individual can find themselves fulfilling several roles, even if not trained or experienced in the role. When working remotely, there might be no-one to delegate to. Employees made excellent use of their skills. However, there became an over-reliance on the employee, who at times experienced unsustainable working hours.
Autonomy	Autonomy is usually not a problem when working remotely; a prerequisite for remote working is the ability to work independently. However, individuals can be undermined if head office is heavy handed, and interferes with communication, say with on-site customers, or if their work is monitored too stringently. For developers working under the spotlight of the customer, autonomy can be problematic.

hours and had less leave than those based in the head office. These factors threaten motivators such as career path, trust, recognition, good management, respect, rewards, and equity.

10.5 Motivational Factors and GSD Guidelines

GSD research is rich in frameworks, guidelines, and recommendations that aim to overcome challenges that arise when software is developed by teams that are separated by geographic distance, that span multiple time zones, have different first languages and represent multiple cultures. However, none of these guidelines are expressly connected to motivation. This section therefore addresses the question, "How do GSD recommended practices support software engineer motivation?" For brevity, consideration is given to guidelines developed specifically for global teams. These are presented in the global teaming model (GTM) according to Richardson et al. (2012) and encompass 20 detailed GSD practices drawn from empirical research and the GSD literature. Where possible the guideline is mapped to the motivational factors identified in Tables 10.2, 10.3, 10.4, and 10.5. Tables 10.6 and 10.7 present a mapping of extrinsic and intrinsic motivational factors to GTM practice guidelines. The motivational factors included in Beecham et al. (2008) are considered, and the case study findings have been used to explore how the Global Teaming Model (GTM) guidelines support motivation.

The mappings shown in Tables 10.6 and 10.7 indicate that if a project manager follows the guidelines offered by the Global Teaming Model, they will in turn also address many of the software engineer's motivational needs (labeled 'need directly addressed'). Those areas not supported by the guidelines (labeled 'need not addressed') tend to be environmental, and outside the scope of a process model such as the Global Teaming Model (Richardson et al. 2012). However, many of the solutions are indirectly addressed (labelled 'implied/partial support of need') and will require careful implementation to ensure the practice fully meets the motivational needs of the virtual engineer. Those motivators labeled 'need directly addressed' will also need further investigation, since as already discussed in this chapter, each guideline needs to be tailored to the individual requirements of the software engineer. (The GTM (Richardson et al. 2012) contains more detailed descriptions of the guidelines.)

Although the Global Teaming Model takes a management or organisation view, it does reflect the needs of the employee, as shown in this practice: "Ensure that the supervision, support and information needs of all team members are met regardless of location." However, although the guidelines for global teaming reflect best practice, they are no substitute for highly experienced project managers (Hall et al. 2008a, b; Beecham et al. 2013; Monasor et al. 2013).

10.6 Theory and Practice of GSD Motivation

Šteinberga and Šmite (2011) conducted a complementary study of motivation in distributed software development teams. They mapped motivators to the GSD literature in order to assess the impact of GSD on software engineer motivation.

Table 10.6 GSD guideline support for intrinsic motivation

SW engineer intrinsic motivators	Global teaming guidelines (Richardson et al. 2012)
	NEED DIRECTLY ADDRESSED
Development needs addressed (e.g., training opportunities to widen skills)	"Training should be tailored to team member's specific needs and location." "Undertake training onsite and face-to-face so team members can be directly assessed and training provision tailored to their specific requirements."
Identify with task (how task fits in with whole)	"Project goals and objectives should be communicated, understood and agreed across all team members regardless of location."
Making a contribution/task significance	"Let team members know their input to process development and ownership is valued."
	IMPLIED/PARTIAL SUPPORT OF NEED
Variety of work (e.g., making good use of skills, being stretched)	"Gather all information relating to the technical . . . experience of potential and existing team members. When teams are in place and project details reported project managers should understand . individual's skills and knowledge."
Technically challenging work	"Gather all information relating to the technical and professional experience of potential and existing team members."
Employee participation	". . . individuals visit locations for extended periods. . ."
Autonomy	Modularisation is one approach to development where work is partitioned into modules that have a well-defined functional whole.
Recognition (for a high-quality, good job done -different to rewards/incentives)	"The global team is viewed as an entity in its own right, regardless of the location of its team members and its performance should be judged and rewarded accordingly." "Acknowledging team success may require tailoring rewards to the needs of different cultures."
Trust/respect	"Structure global team and monitor operation to minimize fear and alienation in teams." "Set up a strategy to handle, monitor and anticipate where conflict between remote locations may occur."
Equity	"Be aware of problems with unbalanced team sizes . . ." "Ensure supervision, support and information needs of all team are met regardless of location."
Empowerment/responsibility	Information of individual's role within the team and specific areas of responsibility [should be recorded].
	NEED NOT ADDRESSED
Career path (opportunity for advancement, promotion prospect) hierarchy; state of economy.	Outside scope of practice model: depends on organisation size;

Table 10.7 GSD guideline support for extrinsic motivation

SW engineer extrinsic factors	Global teaming model guidelines (Richardson et al. 2012)
	NEED DIRECTLY ADDRESSED
Rewards and incentives	"Identify common goals, objectives and rewards." Consider "cultural issues, economic situation and income tax laws when planning rewards."
Employee participation/ involvement	"Face-to-face meetings are recommended when and where possible"
	IMPLIED/PARTIAL SUPPORT OF NEED
Good management	"Ensure that the supervision, support and information needs of all team members are met regardless of location."
Feedback	"Strategies need to be put in place to encourage formal and informal reporting... Seek and encourage input from team members at all locations."
Sense of belonging/supportive relationships	"Provide training to give all team members an opportunity to learn and understand about each other's culture."
Work-life balance	"Achievable milestones should be planned and agreed." "Project manager should allocate tasks and timescales that are realistic."
Appropriate working conditions/environment	Many solutions relate to this under practice "Determine team and organisational structure between locations."
	NEED NOT ADDRESSED
Working in successful company	Environmental/not based on practice
Job security stable environment/	Environmental/not based on practice
Sufficient resources	Environmental/not based on practice

Their assessments, however, do not totally agree with the empirical results reported in this section. These differences indicate that the findings reported in this chapter are preliminary, context specific, and non-generalizable. Šteinberga and Šmite (2011) hypothesized that many motivators challenged by working in a distributed manner would be supported by agile methods. For example, feedback, recognition, and trust are likely to be promoted by iterations and small releases that enable early and frequent feedback, and recognition for a job well-done. Some of these ideas are supported by Beecham et al.'s (2007) empirical study of motivation of teams applying XP development methods. Specifically XP had a positive effect on motivation by clearly monitoring project progress, promoting knowledge sharing and learning on the job, working independently as a team, and communicating good and bad news through positive feedback without punitive repercussions. However, individual recognition was a problem (as the focus was on the team, or pairs of programmers) that could have a negative influence on promotion prospects and rewards. Also, the work tended to be repetitive and therefore did not meet the need for variety. Although agile methods were originally designed for co-located teams, research has shown that distributed teams can apply many of the practices

effectively (Jalali and Wohlin 2012; Hillegersberg et al. 2011). For more information on general agile project management, see Chap. 11.

Returning to motivation theory—how can it help us understand how to manage motivation in virtual teams? The job characteristics theory emphasizes the need to match the person to the role to ensure their growth needs and social needs are met. We found in our analysis of software engineer characteristics that software engineers vary in their profiles. For example, they are not necessarily introverted or motivated by financial rewards (although some might be). Placing these ideas in the context of GSD points to the importance of global project managers identifying which practices can be adapted to meet the needs of the engineer and which practices are fixed. Where the global manager is powerless to change a practice or environmental factor associated with a given task, an option will be to select a person whose characteristics are most suited to the role required to complete that task. If the software engineer enjoys the task to the extent that environmental factors do not detract, or whose growth needs and social needs match those offered by the task, their motivation level should remain high.

An analysis of the impact of GSD on the motivation of practitioners interviewed in the GSD Corp case study listed areas that were 'challenged' by distributed development. Applying GSD best practice in the form of Global Teaming recommendations indicates that good management could, in many cases, counteract these vulnerable areas. Challenged areas in GSD motivation include rewards and incentives, staff development, work-life balance, and promotion opportunities. However, of more concern are those factors that, due to the environment, would be extremely difficult, or even impossible, to change or control. The only way to support practitioners involved in GSD exposed to these fixed factors is by having a clear knowledge of their characteristics. For example, an engineer with a high need to work with people face to face would be unsuited to working in a virtual team. Enjoyment of travel and ability to communicate with people from different cultures is also a prerequisite in many distributed projects.

Tables 10.8 and 10.9 list the characteristics, moderators, and controls associated with software engineers' suitability to working in a GSD environment according to case study findings. Managers can use these tables as a starting point to identify those practitioners suited to working in virtual teams either because of their characteristics, or moderators and controls of those characteristics. For example in Table 10.8, some engineer characteristics (for the sake of the example labeled 'low' suitability for GSD) require high geographic stability (suggesting a dislike of travel), and an introverted personality. A manager may decide that the role demands a lot of travel and interaction with other team members, and therefore this profile could be deemed unsuited to the role. A more suitable set of characteristics for GSD (labeled high GSD compatibility) is likely to be that the software engineer is technically competent, growth oriented, independent, creative, etc.

Sharp et al.'s (2007) empirical research considered the motivation of software engineers in terms of the role they play, thus creating a more pragmatic model of motivation than considering each member of a large team individually. This view reflects Maslow's theory of motivation (Maslow 1954), where, for example,

Table 10.8 Software Engineer Characteristics and GSD Role

	Characteristic	GSD compatibility
Ch.1	Need for stability (organisational stability)	Low
Ch.2	Technically competent	High
Ch.3	Achievement orientated (e.g., seeks promotion)	Medium
Ch.4	Growth orientated (e.g., challenge, learn new skills)	High
Ch.5	Need for competent supervising	Medium
Ch.6	Introverted (low need for social interaction)	Low
Ch.7	Need for involvement in personal goal setting	Medium
Ch.8	Need for feedback (needs recognition)	Medium
Ch.9	Need for geographic stability	Very low
Ch.10	Need to make a contribution (worthwhile/meaningful job)	High
Ch.11	Autonomous (need for independence)	High
Ch.12	Need for variety	High
Ch.13	Marketable	High
Ch.14	Need for challenge	High
Ch.15	Creative	High
Ch.16	Need to be sociable/identify with group	High

Table 10.9 Software engineer moderators and controls and GSD compatibility

	Moderators and controls	GSD compatible
Mod.1	Career stage	At stage that allows flexible working hours and travel
Mod.2	Culture	Open, interested in and tolerant of other cultures
Mod.3	Job type/role/occupational level	Applies to all development roles and levels, though inexperienced levels may not be suited to GSD
Mod.4	State of IT profession	Ideally, buoyant to support feeling of security
Mod.5	Type of organisation	Offers promotion opportunities, e.g., management, customer facing, domain specific, technical roles
Cont.1	Personality traits	Good communicator; not too introverted
Cont.2	Career paths (managerial/technical)	Fixed in a person: either type likely to be compatible with most GSD organisations
Cont.3	Competencies	Ability (both technical and managerial)

developers may be at a different stage in their careers than project managers. Sharp et al. (2007) found a difference in how software developers and project managers were motivated, and suggest that project managers should recognize these differences. Management approaches applied by project managers who assume that developers are motivated in a similar way to themselves are in danger of being ineffective or even detrimental. However, this finding does assume that roles are well defined. In GSD, this is not always the case (Richardson et al. 2012). Furthermore, the literature is divided as to whether clear role definition is a good thing. For example, Kennedy and Nur (2012) found that clear role differentiation can hinder effective project management. Motivation by *group* therefore will be difficult to achieve where group is defined by the role a practitioner plays in the development, if that role is not well defined.

While role ambiguity can be stimulating for the practitioner—and far from the Taylor approach of narrow and specialised work, it can be difficult to manage. The case study in this chapter highlights some advantages and disadvantages associated with role ambiguity: the individual may enjoy up skilling, developing a healthy CV with a broad skill set to market. However, if one member plays several roles, they can become over-stretched in terms of holding key knowledge, and having demands on their time from customers, sales force, head office, development team, etc. Also, the hours they are required to work can be unsustainable. Therefore, despite the challenging work, the individual may leave their employment if no time is allowed for their own needs and for a work-life balance.

10.6.1 Herzberg's Two-Factor Theory and the 'Crowding Out' Effect

Software engineers are motivated by internal factors such as challenging and varied work, fairness, participation, trust, respect, and social interaction (discussed earlier; see Tables 10.4 and 10.5). It is the careful management of these intrinsic factors that will result in a software engineer's increased commitment to the task. External factors such as rewards and salary also need to be managed in order to match the software engineer's expectations. However, internal and external needs must be finely balanced. For example, advantages gained from intrinsic motivation can be *crowded out* by placing too much emphasis on extrinsic motivators (Frey and Osterloh 2002). *Crowding out* can be explained as follows: An employee finds their job interesting and challenging, feels they are treated fairly, and feel that they are part of a team and have something specific to contribute. However, the motivation engendered by these intrinsic factors can be *crowded out*—"obscured by shifting the excitement connected with the job towards monetary reward." According to Frey and Osterloh (2002), "Offering extrinsic motivators, such as salary can actually switch someone's enjoyment and fulfillment from the job itself to doing the job for financial reward. That reward is often short lived." However, the authors add that extrinsic motivators cannot be ignored and that under certain circumstances they are indispensable (Frey and Osterloh 2002).

10.6.2 People, Process, and Creativity

We know from software engineer motivation research that the profession attracts people who are technical and creative with a very high need for challenging work (Beecham et al. 2008). This is reflected in, for example, the enjoyment derived from problem solving or learning a new programming language. The idea of working in the same small area, in piecemeal fashion, is anathema to the software practitioner.

As much as managers might strive to put processes in the place of dependence on people, this can never work in software development, since the task itself is so dependent on skilled people finding solutions to new problems and it is unlikely that two pieces of software will be identical when viewed as a whole.

The very job itself is difficult, hard to estimate, and challenging to get right in terms of meeting customer requirements, since each piece of software is in some way going to be different from other software written in the past. But that is the attraction, the challenge, and perhaps a key reason software engineers are attracted to this profession in the first place.

There is a tension between some management practices and software engineer creativity and job satisfaction. For example, it is difficult to work across sites operating in different time zones without structure: "While synchronous groups can often vary the degree and type of structure dynamically as needed, this is more difficult for distributed asynchronous groups that are dependent on both structure and process rules for coordination" (Ocker et al. 1995). Being dependent on structure fits Taylor's scientific management view that the way to maximize output was to discourage free thought and expect employees to follow prescribed steps (Kennedy and Nur 2012). It appears that controls and processes are orthogonal to creativity. According to Ocker et al., "too much structure, or the wrong structure, can limit the creative process." This is supported by Van de Walle et al., who note that the absence of structured task support led to greater satisfaction in their study of distributed teams (Van de Walle et al. 2007). To allow practitioners the freedom to be creative, the project manager must therefore aim for a correct balance between structure (seen as defined processes) and flexibility.

10.6.3 Returning to the Rational-Economic Model in GSD

Working remotely, managers are encouraged to keep the need for interaction among remote locations to a minimum (e.g., Richardson et al. 2012). It is, after all, these communications that can introduce difficulties such as misunderstandings, stress, and delays into the process, especially when working in different time zones, etc. Effective partitioning and allocation of work across the GSD team is something all managers need to plan for at the start of any project. There are several options to task allocation (Carmel 1999) and, according to Parnas (1972), managers can choose one or more of the following approaches: modularisation, phase-based, or integrated. Partitioning can be component based (Kotlarsky et al. 2007) or lifecycle based (Šmite 2007). Strategic partitioning of the development task can reduce the need for communication across teams.

However, when we consider the individual, the perceived productivity gains of working discretely are likely to be short-lived. Developing software is essentially a human intellectual and social activity (Ferratt and Short 1986; Burn et al. 1992; Jordan and Whiteley 1994; Garza et al. 2003; Sumner et al. 2005). If the work is viewed as repetitive, boring, and fragmented, then the individual may not feel part

of the overall organization and may perceive their work to be meaningless. It is important for engineers' motivation that they perceive that their contributions matter (Ferratt and Short 1986; Crepeau et al. 1992; Garza et al. 2003). Research shows that monotony creates apathy, dissatisfaction, and carelessness (Crepeau et al. 1992; Peters 2003; Sumner et al. 2005; Ituma 2006), particularly when an individual does not develop new skills. However, another issue with task allocation by site is career advancement. For example, if a programmer desires to become a software architect, he or she needs to see a career path and be given an opportunity to learn related new skills. Working remotely can mean that the individual either does not have the scope to advance up the career ladder, or that they may be overlooked.

The concepts and theories relating motivation to a GSD context have been drawn from the literature, and the case study has been used to provide real-world examples of how these theories can be applied in practice. There are limitations to using one case study, and findings are used merely as indicators of where practices can help or hinder motivation. For example, some engineers might be highly motivated by salary, and provided they are well paid, they will continue to produce high-quality work despite many other motivators being challenged.

10.7 Summary and Conclusions

This chapter explored motivation theories and used a case study as an example of how to motivate software engineers working in virtual teams. Theories of motivation suggest that people, in whatever sphere, will have their own specific needs, and that it is the strength of those needs and the likelihood that they will be met by a given task that will determine the energy and enthusiasm the individual will expend on that task. However, despite the bespoke nature of motivation, the research does point to areas that need to be considered in every case. In every situation, the manager needs to balance three things: the *task*, the *environment*, and the software engineer's *characteristics*. If any of these are mismatched, no amount of stimulus from the job will result in sustained motivation.

Software development teams cannot be treated as a homogeneous group with similar characteristics. As it appears that engineers' needs differ according to the role they play, a way to manage engineers' motivation is by role. However, roles are an area that can be blurred, particularly in a global setting. The research and findings from the case study reported in this chapter have identified this as a problem. The demands placed on engineers working remotely can mean that they are encouraged to take on many roles to ensure project success. While keeping roles and responsibilities fluid might suit upper management, and even meet the software engineer's need for varied and challenging work, it can place unrealistic pressures on an individual's time.

This chapter listed factors known to motivate software engineers. Twenty-two different factors that motivate software engineers to produce high-quality software

have been divided into intrinsic and extrinsic motivators. Case study findings, drawn from a company engaged in distributed software development, were mapped to each motivation factor. In this way, preliminary results were presented relating to areas that particularly threaten software engineers' motivation in a global setting.

Taking a needs-theory approach, managers must first consider which motivation factors they can control and which are outside their control. For example, the negative influence of extrinsic factors can be reduced by ensuring that employees are given adequate pay, a feeling of security, and good management. As global software development stems from the 'environment' it makes sense to view external factors as a separate threat or enhancement to motivation. Some environmental factors are outside the control of the manager. For example, no amount of best practice can change the culture of a country, the security of a job in a volatile economy, or a limited career path in a small organization.

The job characteristics theory (Hackman and Oldham 1974) emphasizes the importance of person-job fit. The GSD engineer needs to have certain characteristics in place that are resilient to the environment, leaving them free to be motivated by the intrinsic aspects of the work. Ideally, a software engineer recruited to work in a virtual team will be at a career stage that allows them flexibility to travel, flexibility in their place of work, and the hours they work. Also, they need to be open, tolerant, and interested in different cultures. They need a good level of confidence in their own ability, to know their own limitations, to be good and clear communicators, and on the extrovert end of the personality spectrum.

So, once the person-job fit has been matched, and extrinsic factors controlled for where possible, managers can turn their attention to the intrinsic factors that relate to the job itself. It is getting these factors right that will motivate software engineers to do the best job they can. Software engineers need technically challenging work, variety of tasks, and evidence that their efforts will result in a useful contribution. Engineers also require developmental training, to feel involved, recognition and rewards for doing a good job, and to be treated fairly, regardless of their location. Finally, they need trust and respect, responsibility, autonomy, and empowerment.

This chapter looked at how motivation theory can help to solve some GSD organisational problems such as low employee commitment and high turnover. It also examined how motivation might promote software quality and inspire innovation. However, an assumption associated with recommended management practices is that the employee stays in the organisation long enough to benefit from any motivation program. We have seen that heavy-weight processes can stifle innovation and that enforced compliance will demotivate the technical employee. A way forward could be to apply agile development methodologies and empower employees by creating an environment that allows a worker to develop a sense of ownership and pride of accomplishment (Kennedy and Nur 2012).

There is still more work required in this area. We need ways to measure the person-job fit for GSD. Also, although agile methods appear to address many software engineer motivation needs in co-located settings, we still need to know how to implement agile methods in a distributed setting so that motivation is positively affected.

Finally, when managers are allocating tasks to engineers, they would do well to heed the advice given by Herzberg:

If you want someone to do a good job, give them a good job to do.

Acknowledgments Thanks are extended to the many practitioners who gave their time and made this study possible. Also, thanks are given to the reviewers and colleagues who commented and proofread early versions of this chapter. This work was supported in part by Science Foundation Ireland grant 10/CE/I1855 to Lero – the Irish Software Engineering Research Centre (www.lero.ie).

References

Adams JS (1963) Toward an understanding of inequity. J Abnorm Social Psychol 67:422–436

Bassett-Jones N, Lloyd GC (2005) Does Herzberg's motivation theory have staying power? J Manage Develop 24(10):929–943

Beecham S, Baddoo N, Hall T, Robinson H, Sharp H (2008) Motivation in software engineering: a systematic literature review. Info Softw Technol (IST), Elsevier 50(9–10):860–878

Beecham S, O'Leary P, Baker S, Richardson I, Noll J (2013) Who are we doing global software development research for? In: 8th IEEE international conference on global software engineering (ICGSE'13), Bari, Italy

Beecham S, Sharp H, Baddoo N, Hall T, Robinson H (2007) Does the XP environment meet the motivational needs of the software developer? An empirical study. Agile 2007 conference. Washington, DC

Burn JM, Couger JD, Ma L (1992) Motivating IT professionals. The Hong Kong challenge. Info Manage 22(5):269–280

Carmel E (1999) Global software teams: collaboration across borders and time zones. Prentice Hall, Saddle River, NJ

Couger JD, Zawacki RA (1980) Motivating and managing computer personnel. Wiley, New York

Crepeau RG, Crook CW, Goslar MD, McMurtrey ME (1992) Career anchors of information systems personnel. J Manage Info Syst 9(2):145–160

da Silva FQ, França ACC (2012) Towards understanding the underlying structure of motivational factors for software engineers to guide the definition of motivational programs. J Syst Softw 85 (2):216–226

Deshpande S, Richardson I, Casey V, Beecham S (2010) Culture in global software development - a weakness or strength? In: IEEE international conferences on global software engineering (ICGSE 2010), Princeton, NJ

Fernández-Sanz L, Misra S (2011) Influence of human factors in software quality and productivity. In: Murgante B, Gervasi O, Iglesias A, Taniar D, Apduhan B (eds) Computational science and its applications - ICCSA 2011, vol 6786. Springer, Berlin, pp 257–269

Ferratt TW, Short LE (1986) Are information systems people different: an investigation of motivational differences. Manage Info Syst (MIS) Q 10(4):377–387

França ACC, Carneiro DE, da Silva FQ (2012) Towards an explanatory theory of motivation in software engineering: a qualitative case study of a small software company. 26th IEEE Brazilian Symposium on Software Engineering (SBES)

Frey BS, Osterloh M (2002) Successful management by motivation: balancing intrinsic and extrinsic incentives. Springer, Berlin

Garza AI, Lunce SE, Maniam B (2003) Career anchors of Hispanic information systems professionals. Proceedings - annual meeting of the decision sciences institute, pp 1067–1072

Hackman JR, Oldman GR (1976) Motivation through the design of work: test of a theory. Academic Press, New York

Hackman RJ, Oldham GR (1974) The job diagnostic survey: an instrument for the diagnosis of jobs and the evaluation of job redesign projects. Office of Naval research manpower administration: NCIS national technical information service, US Department of Commerce

Hall T, Beecham S, Baddoo N, Sharp H, Robinson H (2009) A systematic review of theory use in studies investigating the motivations of software engineers. ACM Transac Softw Eng Methodol (TOSEM) 18(3):10

Hall T, Beecham S, Verner J, Wilson D (2008a) The impact of staff turnover on software projects: the importance of understanding what makes software practitioners tick (Refilling the pipeline: meeting the renewed demand for information technology workers). In: ACM-SIGMIS CPR'08 Conference, Charlottesville, VA, April 3–5

Hall T, Sharp H, Beecham S, Baddoo N, Robinson H (2008b) What do we know about developer motivation? IEEE Softw 25(4):92–94

Handy C (1993) Understanding organisations, 4th edn. England Penguin Books Ltd, Middlesex

Herzberg F, Mausner B, Snyderman BB (1959) Motivation to work, 2nd edn. Wiley, New York

Hillegersberg Jv, Ligtenberg G, Aydin MN (2011) Getting agile methods to work for Cordys global software product development. In: Fifth global sourcing workshop, Courchevel 1850, France

Huczynski AA, Buchanan DA (1991) Organizational behaviour: an introductory text, 2nd edn. Prentice Hall, London

Ituma A (2006) The internal career: an explorative study of the career anchors of information technology workers in Nigeria Proceedings of the 2006 ACM SIGMIS CPR conference on computer personnel research: forty four years of computer personnel research: achievements, challenges & the future. Claremont, CA, pp 205–212

Jalali S, Wohlin C (2012) Global software engineering and agile practices: a systematic review. J Softw Evol Proces 24(6):643–659

Johnson RE, Chang C-HD, Yang L-Q (2010) Commitment and motivation at work: the relevance of employee identity and regulatory focus. Acad Manage Rev 35(2):226–245

Jordan E, Whiteley AM (1994) HRM practices in information technology management. In: Computer personnel research conference (SIGCPR) on Reinventing IS: managing information technology in changing organizations. ACM Press, Alexandria, VA

Kennedy D, Nur M (2012) The rise of taylorism in knowledge management. In: Proceedings of PICMET'12: technology management for emerging technologies (PICMET)

Kotlarsky J, Oshri I, von Hillegersberg J (2007) Globally distributed component-based software development: an exploratory study of knowledge management and work division. J Info Technol 22:161–173

Krippendorff K (1980) Content analysis an introduction to its methodology. Sage, Beverly Hills, CA

Locke EA (1968) Toward a theory of task motivation and incentives. Organ Behav Human Perfor 3:157–189

Maslow A (1954) Motivation and personality. Harper & Row, New York

McClelland DC (1961) The achieving society. Van Nostrand, Princeton, NJ

McConnell S (1996) Avoiding classic mistakes [software engineering]. IEEE Softw 13(5):111–112

Monasor MJ, Vizcaíno A, Piattini M, Noll J and Beecham S (2013) Towards a global software development community web: identifying patterns and scenarios. In: PARIS Workshop, International Conference on global software development (ICGSE), Bari, Italy

Mullins LJ (1993) Management and organisational behaviour. Pitman Publishing, London

Noll J, Beecham S, Richardson I (2010) Global software development and collaboration: barriers and solutions. ACM SIGCSE bulletin - special section on global intercultural collaboration (September)

Noll J, Beecham S, Seichter D (2011) A qualitative study of open source software development: the OpenEMR project. In: IEEE empirical software engineering and measurement conference – ESEM 2011, Banff, Canada, September, 19–23

Ocker R, Hiltz SR, Turoff M, Fjermestad J (1995) The effects of distributed group support and process structuring on software requirements development teams: results on creativity and quality. J Manage Info Syst 12(3):127–153

Olson JS, Olson GM (2004) Culture surprises in remote software development teams. ACM Q, Nova Iorque 1(9):52–59
Parnas D (1972) On the criteria to be used in decomposing systems into modules. Commun ACM 15(12):1053–1058
Peters L (2003) Managing software professionals. IEMC'03 proceedings. managing technologically driven organizations: the human side of innovation and change (IEEE Cat. No.03CH37502). IEEE, Albany, NY, pp 61–66
Petri HL, Govern JM (2012) Motivation: theory, research, and application, 6th edn. Wadsworth Publishing, Belmont, CA
Richardson I, Casey V, McCaffery F, Burton J, Beecham S (2012) A process framework for global software engineering teams. Info Softw Technol 54(11):1175–1191
Riehle D (2007) The economic motivation of open source: stakeholder perspectives. IEEE Comput 40(4):25–32
Roberts J, Hann I, Slaughter S (2004) Understanding the motivations, participation and performance of open source software developers: a longitudinal study of the Apache projects. Carnegie Mellon University Working Paper
Shah H, Nersessian NJ, Harrold MJ, Newstetter W (2012) Studying the influence of culture in global software engineering: thinking in terms of cultural models In: ACM proceedings of the 4th international conference on intercultural collaboration, Bengaluru, India
Sharp H, Baddoo N, Beecham S, Hall T, Robinson H (2009) Models of motivation in software engineering. Info Softw Technol 51(1):219–233
Sharp H, Hall T, Baddoo N, Beecham S (2007) Exploring motivational differences between software developers and project managers. In: The 6th joint meeting of the european software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering (ESEC/FSE07), Dubrovnik, Croatia
Skinner BF (1976) Walden two. Macmillan, New York
Šmite D (2007) Global software development improvement. PhD Thesis, Riga Information Technology Institute, University of Latvia
Šteinberga L, Šmite D (2011) Towards a contemporary understanding of motivation in distributed software projects: solution proposal, vol 770. University of Latvia, Computer Science and Information Technologies, pp 15–26
Sumner M, Yager S, Franke D (2005) Career orientation and organizational commitment of IT personnel. ACM SIGMIS CPR conference on computer personnel research (Atlanta, Georgia, USA, April 14 16, 2005) pp 75–80
Taylor FW (1947) Scientific management. Harper & Row, New York
Van de Walle B, Campbell C, Deek FP (2007) The impact of task structure and negotiation sequence on distributed requirements negotiation activity, conflict, and satisfaction, published by LNCS
Verner J, Beecham S, Cerpa N (2010) Stakeholder dissonance: disagreements on project outcome and its impact on team motivation across three countries. In: ACM SIGMIS CPR'10, Vancouver, Canada
Verner J, Ali-Barbar M, Cerpa N, Hall T, Beecham S (2014) Factors that motivate software engineering teams: a four country empirical study. J Syst Softw 95:115–127
Vroom VH (1964) Work and motivation. Wiley, New York
Ye Y, Kishida J (2003) Toward an understanding of the motivation of open source software developers. Proceedings - International conference on software engineering, pp 419–429

Biography Sarah Beecham holds the position of research fellow in the Process Quality Group in Lero—the Irish Software Engineering Research Centre. Sarah's research interests include software fault prediction, effort estimation, evidence-based software engineering, requirements engineering, and software process improvement. She has published widely in the area of software engineer motivation.