Association for Information Systems
AIS Electronic Library (AISeL)

SAIS 2012 Proceedings | Southern (SAIS)

2012

A Theory of Software Development Methodologies

Adarsh Kumar Kakar
University of Alabama, askakar@crimson.ua.edu

A THEORY OF SOFTWARE DEVELOPMENT METHODOLOGIES

Adarsh Kumar Kakar
University of Alabama
askakar@crimson.ua.edu

ABSTRACT

In this study, it is proposed that software development methodologies (SDMs), when looked at from the perspective of job design, offers interesting and useful insights. The increasing popularity of agile methods over plan-driven methods in the 2000's mirror the increasing popularity of non-Taylorist job designs over Taylorist job designs in the 1980's, when jobs were redesigned by adopting self-managed teams and work groups, and creating employee programs like quality circles with salutary results. This study finds the widely accepted (JCM) Job Characteristic Model (Hackman and Qldham, 1976) for job design relevant in providing a theoretical foundation for the atheoretical domain of SDMs. JCM provides a structural framework for practice to understand what they are doing right and what needs to change by diagnosing the characteristics of the software development processes in need of improvement and making recommendations for tailoring SDMs for superior work outcomes.

Keywords

Software development methodologies, job characteristic model, agile methods, plan-driven methods

INTRODUCTION

This study is an attempt to balance theory and empiricism in the area of SDMs. The aim is to provide a theoretical framework for a deeper understanding of SDMs, a sound basis for their comparison and make tailoring recommendations to practice. While there is substantial literature discussing the merits and demerits of SDMs, there is a need for a structural framework from which these assessments could be looked back. It is suggested that a theory based approach would allow researchers to make sense of the empirical findings and trends in the area of SDMs as well as provide a benchmark for practice to determine what they are doing right and where they are going wrong.

The literature on job design contrasts "Taylorist" jobs to the "Enriched" jobs. Fredrick Taylor (1947) viewed job design as a scientific optimization problem, where industrial engineers study the production process and devise the most efficient way to break that process into individual, precisely defined tasks. Typically, a Taylorist job is highly specialized, and workers are not encouraged to experiment, innovate, or otherwise vary the way that tasks are performed.

In the 1970's, academics such as Richard Hackman, Edward Lawler and Greg Oldham started to argue that Taylorist job design is sub-optimal (Hackman and Qldham, 1976; Hackman and Lawler, 1971; Lawler, 1973; Porter, Lawler and Hackman, 1975). Enriched jobs, by encouraging workers to learn and innovate at work, increase the motivating potential of work. Motivated workers perform tasks more accurately and are more likely to find productivity innovations that engineers overlook. In the 1980's, firms put the theory into practice by redesigning jobs, adopting self-managed teams and work groups, and creating employee participation programs like quality circles.

A similar trend was witnessed in the relatively new discipline of software engineering. With the proclamation of the Agile manifesto in 2001(http://agilemanifesto.org), the last decade saw a rapid increase in the popularity of the Agile methods as compared with the Taylorist methods such as the waterfall model and its variances. Although this trend in the software domain favoring agile development methodologies came in much later than the trend in favor of non-Taylorist or "enriched" job design in other domains, there is a strong equivalence between them. The agile manifesto has been welcomed by many in the software development community who often perceived formal processes as management generated inefficiency that gets in the way of productivity (Anderson, 2005). Agile development proponents question the assumption that change and uncertainty can be controlled through a high degree of formalization and have discovered inadequacies in formal design that follow systematic procedures dictated by rigid processes (Nerur and Balijepally, 2007).

However, the Agile manifesto principles are insufficiently grounded in theory (Conboy and Fitzgerald, 2004). This study suggests that Agile methodologies in essence represent a generic category of alternative job design with people focus rather than the formal process focused job design of Taylorist plan-driven methods. Non-Taylorist or "enriched" job designs are agile. They are designed to rapidly respond to changing situations through built-in autonomy, skill variety and rapid feedback due to close and frequent interaction amongst teams. "Enriched" job designs are not weighed down by the formal plan-driven and process oriented approach of Taylorist jobs. The aim of this study is to examine the relevance of adopting a job-design

perspective in providing a credible theoretical framework to the atheoretical domain of SDMs. It investigates the appropriateness of JCM, *one of the most elaborate and widely accepted theories of job design* (Kiggundu, 1981), in explaining the emerging trends and empirical observations in the domain of SDMs.

THE JOB CHARACTERISTICS MODEL

Job design is an attempt to influence motivation through work. In Herzberg's motivation- hygiene theory (Herzberg, 1966), a distinction is made between factors that are motivators and hygiene factors. Motivators are thought to be work related factors such as challenging work, achievement, responsibility and personal competence. Hygiene fa**C**tors are external factors such as company policies, supervisory practices, pay and working conditions. They are not part of the work itself and have no power to motivate the employee but their absence can lead to employee dissatisfaction. Herzberg proposed that jobs enriched to include motivators to enhance work motivation. Job enrichment is a type of job redesign intended to reverse the effects of tasks that are repetitive requiring little autonomy. The underlying principle is to expand the scope of the job with a greater variety of tasks, vertical in nature, that require self-sufficiency.

Hertzberg's work on job enrichment was further refined in 1975 by Hackman and Oldham using what they called as the Job Characteristics model (See Figure 1).

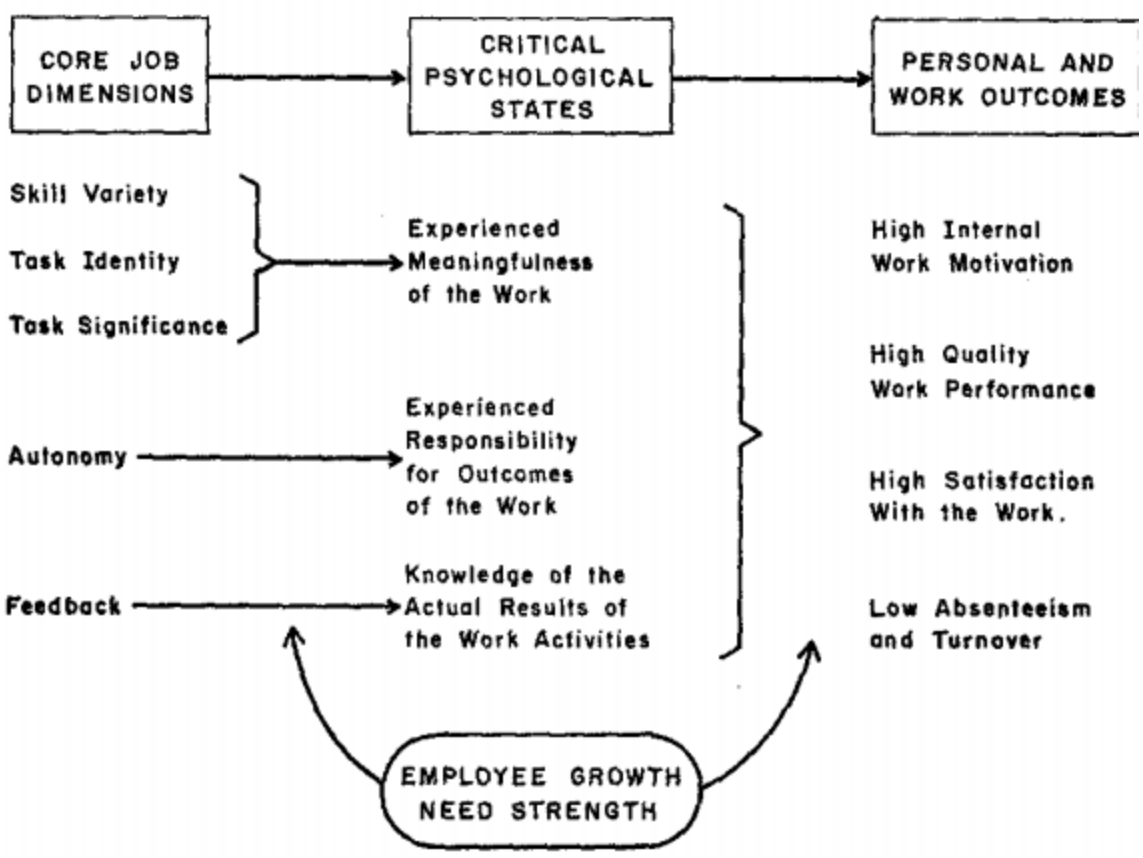

Figure 1. Job Characteristics Model (Hackman and Oldham, 1976)

Job characteristics theory claims that an individual will be motivated to work when jobs are designed to satisfy three critical psychological states. These include:

1. the need for meaningful work
2. the need to be responsible for work outcomes
3. the need for performance feedback.

These critical psychological states are affected by the five characteristics of job which describe:

1. Skill Variety. The degree to which a job requires a variety of different activities in carrying out the work, which involve the use of a number of different skills and talents of the person.
2. Task Identity. The degree to which the job requires completion of a "whole" and identifiable piece of work; that is, doing a job from beginning to end with a visible outcome.
3. Task Significance. The degree to which the job has a substantial impact on the lives or work of other people, whether in the immediate organization or in the external environment.
4. Autonomy. The degree to which the job provides substantial freedom, independence, and discretion to the individual in scheduling the work and in determining the procedures to be used in carrying it out.
5. Feedback. The degree to which carrying out the work activities required by the job results in the individual obtaining direct and clear information about the effectiveness of his or her performance.

The critical psychological states in turn affect the motivation, job satisfaction and productive behaviors of employees.

When applied in the context of SDMs, the five characteristics of the model can be used to characterize a typical Agile and Taylorist method of software development (Figure 2).

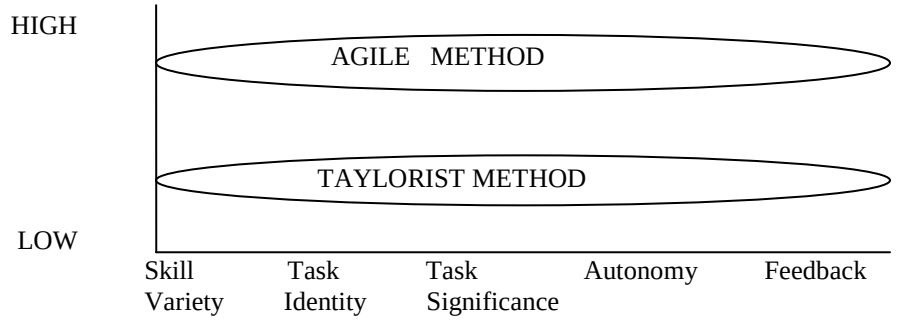

Figure 2. A Comparison of Agile and Taylorist Methods

Taylorist software development methods deploy specialized role based teams, with individual team members requiring less skill variety to accomplish jobs. Detailed planning is done of entire software development lifecycle activities including requirements gathering, design, construction, testing and project coordination and management activities and specialized people handle each of these tasks. The allocation of work specifies "not only what is to be done but how it is to be done and the time allowed for doing it" (Chau, Maurer and Melnik, 2003). This reduces the autonomy of employees and shifts the focus from individuals and their creative abilities to the processes themselves.

On the other hand agile methods emphasize and value individuals and interactions over processes. Agile methods are people-centric, recognizing the value competent people and their relationships bring to software development (Nerur, Mahapatra, and Mangalaraj, 2005). People issues are at the heart of the agile movement (Boehm and Turner, 2005). The agile team works by placing people physically closer, replacing documents with talking in person and at whiteboards, improving the team's amicability and its sense of community (Cockburn and Highsmith, 2001). Tasks are not specialized to the degree of plan-driven methods. For example agile methods typically do not have separate coders and designers.

Agile methods move away from a deterministic/ mechanistic view of problem solving to a dynamic process characterized by iterative cycles and the active involvement of all stakeholders. Unlike the Taylorist methods, where the cycle time between requirements gathering and product release is typically very long, the gaps between customer requirements and implementation into the product in agile projects are narrowed in rapid cycles. The focus on developing working products rather than paper artifacts and components enhances task identity and task significance. Big upfront design plans and extensive documentation are of little value to practitioners of agile methods (Nerur and Balijepally, 2007). Important features of this approach include evolutionary delivery through short iterative cycles – of planning, action, reflection – intense collaboration, self-organizing teams, and a high degree of developer discretion, providing the team members autonomy as well as quick feedback on the work accomplished. The agile paradigm empowers individuals through a focus on developing working products, ownership and shorter feedback cycles (Boehm and Turner, 2005), satisfying the three psychological states of the job characteristics model, the need for meaningful work, the need to be responsible for work outcomes, and the need

for performance feedback. This increases the motivating potential of work as given by the formula (Hackman and Oldham, 1976):

$$\text{MPS} = \frac{\text{Skill Variety} + \text{Task Identity} + \text{Task Significance}}{3} \times \text{Autonomy} \times \text{Job Feedback}$$

resulting in higher team member morale, satisfaction and productivity.

EXPLANATORY POWER OF JCM

JCM is able to explain the relevance of various best practices. For example the benefits of Paired programming can be explained by the rapid feedback it provides to the developers. The benefits of developing working products in rapid iterative cycles is due to the enhanced significance of the task completed and getting early feedback from the users of the product. In addition developing whole, meaningful and working products makes it easy for developers to identify with the tasks that are fulfilled. This in turn increases the motivating potential of team members and the resulting work outcomes. This is in contrast to the work where developers are given a specification for parts of the solutions, and do not have full picture of the product this is being developed.

EMPIRICAL SUPPORT

In a review of research investigating team motivation of software development projects, this study did not come across a single article that concluded that team members who worked in plan-driven projects demonstrated higher morale compared to those that followed agile approaches. In contrast, numerous studies have found that the morale and motivation of agile team members is higher compared to those using Taylorist plan-driven methods. These observations are in line with the theory of JCM and its predictions about the psychological and behavioral outcomes of the Taylorist and non-Taylorist job design.

The results of a survey (Cockburn and Highsmith, 2001) of 200 people from a wide range of organizations in North America, Europe, Australia, India, and other locations assessing agile and rigorous methodologies demonstrated that agile methodologies scored better than rigorous methodologies in terms of employee morale. In another survey (Mannaro, Melis, and Marchesi, 2004) of job satisfaction among 122 employees of software companies that used XP (Extreme Programming) and companies that did not use agile development methods, 95% of the employees who used XP answered that they would like their company to continue using their current development process, while the number for the employees in companies that did not use agile development methods was 40%. The employees in the companies that used XP were significantly more willing to use the development process in the future than the employees in companies that did not use XP. In addition the results showed that members of the agile project team experience greater job satisfaction, feel that the job environment is more comfortable, and believe that their productivity is higher compared to those of non-agile methods.

A study (Mann and Maurer, 2005) found all developers recommended the use of Scrum in future projects. The developers were more satisfied with the product, and saw that the Scrum process fostered more customer involvement and communication. In a comparative analysis by Melnik and Maurer (2006) of the way agile teams and general IT professionals in the industry perceive their work environments revealed significantly higher rates of satisfaction by agile team members. In addition, it was found that not only workers but also managers of agile teams are overwhelmingly satisfied with their jobs. In another study (Layman, Williams and Cunningham, 2004) that conducted semi-structured interviews of software developers by asking mixture of open-ended and specific questions, several team members stated that they enjoyed their jobs and enjoyed the XP methodology more than the waterfall method. A detailed review of agile development literature (Dyba and Dinsoyr, 2008) concluded that most developers were found satisfied with agile methods and suggested that agile methods can improve job satisfaction and productivity.

RECOMMENDATIONS OF JCM

Although SDMs are broadly classified into two categories, the Agile methods and the Taylorist methods, within each category there are many different methods each with their own principles and practices making comparisons between them confusing. For example, there are many Agile methods currently in use such as Extreme programming, Scrum, Crystal methodologies, Dynamic Software development method (DSDM), Feature Driven Development (FDD) and Lean Software Development Method (LSDM) with each focusing heavily on some of the principles of the agile manifesto and completely ignores others making it impossible to reach any conclusions on agile methods and their use (Conboy and Fitzgerald, 2004).

JCM suggests that in order to improve psychological and behavioral outcomes at work, all the five core job characteristics should be developed. The degree of each of the five job characteristics can be measured using an assessment tool, the (JDS) Job Diagnostic Survey (Hackman & Oldham, 1974), thus providing a uniform basis for diagnosing the motivating potential

of SDMs. The JDS also provides for the measurement of the three critical psychological states and work outcomes. If the psychological and work outcomes of a software development project do not meet defined goals, then interventions can be planned based on the JCM model to achieve desired results.

The model does not impose any constraint on how the development process should be designed or which software development is used. Depending on the context the process may be people-centric or process-centric or a suitable combination of both. Whatever be the method adopted for software development, JCM can be used as a conceptual basis for the diagnosis and identification of those specific job characteristics that are most in need of improvement to improve work outcomes. In addition to providing guidance for tailoring SDMs, JCM can also be used as a framework for assessing and interpreting measurements collected to evaluate the effects of changes that have been carried out, to determine which job dimensions did and did not change, to assess the impact of the changes on the affective and motivational responses of employees and the resulting work outcomes.

If the development context, such as a large project size or outsourcing, demands that the waterfall method be adopted, then JCM offers ways of mitigating the risks inherent in Taylorist methods and improving team member morale through manipulation of the five characteristics. Skill variety can be enhanced in a software development project through job rotation by involving developers in requirement gathering activities as well as quality assurance. A shared vision for the project can be evolved to help increase task identity and significance for the individual team members. This will enable the team members see the big picture (task identity) and make them realize the grand purpose (task significance) of the specialized tasks which they perform. This in turn will increase the meaningfulness of work and sustain high team member morale. The need for performance feedback and autonomy can be fulfilled through periodic goal setting, securing team members' commitment to plan and frequent project team meetings to track progress. These recommendations of the JCM also happen to be the best practices for plan-driven Taylorist software development approaches. The guidance emanating from the JCM model is thus well aligned with known best practices of software development, generating added confidence in the relevance of this model.

CONCLUSION

A methodology is a systematic way of performing a task or doing work. Therefore it is logical to look at SDMs from the perspective of job design. It opens up avenues to vast existing literature on job design and makes them available to a newer discipline of software engineering. This study finds the well accepted and rigorously tested JCM well suited to provide the theoretical underpinning for SDMs. Currently there is no theory of SDMs. "Without theory we are just groping in chaos" as building knowledge involves systematic revision and extension of theory based on comparison of prediction with observation (Deming, 1986). In the absence of theory improvement efforts will be adhoc and their effects uncertain.

A good theory is one which both explains and predicts. JCM is able to credibly explain the higher motivation of project teams using Agile practices and account for their increasing popularity. In addition JCM provides a handy and useful framework for understanding the implications of using different SDMs and for recommending corrective actions to achieve desired project outcomes. Although further confirmatory empirical studies may be necessary, the alignment of these recommendations with the well-known best practices in software development provides reasonable assurance about the relevance of JCM as a theoretical framework for SDMs.

REFERENCES

1. Anderson, D. J. (2005) Stretching Agile to fit CMMI Level 3 - the story of creating MSF for CMMI ® Process Improvement at Microsoft Corporation, presented at Agile 2005 Conference, http://agile2005.org/, 2005. (1.3.2006).
2. Boehm, B. and Turner, R. (2005) Management challenges to implementing agile processes in traditional development organizations, *IEEE Software*, 22(5), 30–39.
3. Cockburn, A. and Highsmith, J. (2001) Agile software development: The people factor, *Computer*, 131-133.
4. Conboy, K. and Fitzgerald, B. (2004) Toward a conceptual framework of agile methods: a study of agility in different disciplines, in: Proceedings of XP/Agile Universe, Springer Verlag.
5. Chau, T., Maurer, F. and Melnik, G. (2003) Knowledge sharing: Agile methods vs. Tayloristic methods, IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), Linz, Austria, 302-308.
6. Deming, W.E. (1986) Out of the crisis, Cambridge, MA, MIT Press.
7. Dyba, T. and Dinsoyr, T. (2008) Empirical studies of agile software development: A systematic review, Information and Software Technology, 50(9-10), 833-859.

8. Hackman J.R, Oldham G.R. (1974) The Job diagnostic survey: An instrument for the diagnosis of jobs and the evaluation of job redesign projects, JSAS Catalog of Selected Documents in Psychology (Ms. No. 810), 4, 148.
9. Hackman, J.R, and Qldham, G,R. (1976) Motivation through the design of work: Test of a theory, *Organizational Behavior and Human Resources*, 16(2), 250-279.
10. Hackman, J.R., and Lawler, E.E. (1971) Employee reactions to job characteristics, *Journal of Applied Psychology Monograph*, 55, 259-286.
11. Herzberg, F. (1966) Work and nature of man, World Publishing, Cleveland, OH.
12. Kiggundu, M. N. (1981) Task interdependence and the theory of job design, *Academy of Management Review*, 6, 499 – 508.
13. Lawler, E. E. (1973) Motivation in work organizations. Monterey, Calif.: Brooks/Cole.
14. Layman, L., Williams L. and Cunningham L. (2004) Exploring extreme programming in context: An industrial case study, Agile Development Conference.
15. Mann, C., Maurer, F. (2005) A case study on the impact of scrum on overtime and customer satisfaction, Agile Development Conference.
16. Mannaro, K., Melis, M. and Marchesi, M. (2004) Empirical analysis on the satisfaction of IT employees comparing XP practices with other software development methodologies, in: Extreme Programming and Agile.
17. Melnik, G. and Maurer, F. (2006) Comparative analysis of job satisfaction in agile and non-agile software development teams, XP2006.
18. Nerur, S., Mahapatra, R. and Mangalaraj, G. (2005) Challenges of migrating to agile methodologies, *Communications of the ACM*, 72–78.
19. Nerur, S. and Balijepally, V. (2007) Theoretical reflections on agile development methodologies, *Communications of the ACM*, 50 (3), 79–83.
20. Porter, L. W., Lawler, E. E., and Hackman, J. R. 1975. Behavior in organizations. New York: McGraw-Hill.
21. Taylor, F. W. (1947) Shop management (published as part of Scientific Management). New York: Harper.