Toward an Understanding of Job Satisfaction on Agile Teams: Agile Development as Work Redesign

John F. Tripp
Baylor University
john_tripp@baylor.edu

Cynthia K. Riemenschneider
Baylor University
c_riemenschneider@baylor.edu

Abstract

Agile methodologists have claimed that a key value proposition for the adoption of agile methods is that the methods' practices, processes, and philosophy make people more motivated and satisfied with their jobs. However, while several studies have found evidence for this impact, there has not been extensive theoretical support to explain why. In this study, we use the lens of Hackman & Oldham's job characteristics model to motivate a theory of motivation and satisfaction amongst agile development teams. We propose that agile teams are, in fact, redesigning work in the very way that Hackman & Oldham propose will increase job perceptions, and lead to greater job satisfaction. We report the initial results of a research-in-progress study. Using a quantitative survey of 104 software professionals, we test the theory and find preliminary support for our model and hypotheses.

1. Introduction

Over the past decade, agile software development methods have become extensively used and, according to a recent Forrester report, have now been adopted to some extent by a majority of companies [24]. The proponents of agile methods have made two particular claims about the impacts of their use. First, they claim that the methods produce better software. This claim has been researched to a great extent, and support has been found as to the impact of some agile practices on project success. The second key claim of agile practitioners is that people who work on agile teams are more motivated and satisfied. Specifically, they claim that when agile methods are used, people "want to work there" [11]. While initial research has been performed on the impacts of particular agile practices on motivation [e.g., 13] , research on this claim of agile impacts is still in its infancy.

To explore this issue, we utilize the Job Characteristics Model [JCM; 10] as a lens through which to view the impacts of agile software development method use on individuals' perceptions and attitudes they develop about their jobs. The JCM proposes that the characteristics of a job influence a person's perceptions of the job, and their attitudes about it. The JCM has been used as a lens to study the impact of job design on job satisfaction [e.g., 10], turnover intention [e.g., 1], work exhaustion [14] and more. While there is substantial empirical support on the impacts of job characteristics on job attitudes in many IS contexts, far less research has been performed on how the design of work in IS teams may impact the perceptions of job characteristics. Given that agile practitioners have made particular claims about the fact that their methods produce the by-product of higher job satisfaction amongst the team members, it is reasonable to assume that the use of the methods may impact job perceptions.

The previous research on motivation and job satisfaction on agile teams has been largely executed using a case study approach. Tessem & Maurer [21] used a case study of a team using the Extreme Programming (XP) agile method. Their findings suggested that the JCM constructs were observable, and while some interviewees stated that they were satisfied, their data couldn't support testing this. McHugh et al. [13] investigated the impact of the use of agile practices on agile team motivation. Using Beecham et al.'s [4] factors of IS worker motivation, they observed and identified the impact of three feedback mechanisms of agile methods – iteration planning, daily stand up meeting, and the iteration retrospective as being associated with perceptions of JCM constructs. The presence of this initial evidence warrants further study of our research question:

How does agile method use impact job satisfaction?

Broadly, our research objective is to adapt and expand the JCM to the context of agile method use. Using the JCM as our starting point, we develop a model that proposes that the use of agile methods impacts job satisfaction, mediated by its impact on job perceptions. Further, we propose moderating effects of iteration speed and previous experience using non-agile methods. This model contributes to the previous IS literature by focusing on the antecedents of job perceptions, and by recognizing that ISD methods are, to a great extent, engines of work redesign. Further, this model contributes to the literature by providing a theoretical lens through which to explain the impacts of agile methods on job attitudes.

2. Theoretical Background

2.1. The Job Characteristics Model and Job Satisfaction

Hackman and Oldham's [10] Job Characteristics Model (JCM) is one of the most tested theoretical models in social science. It defines five job characteristic perceptions that impact a person's attitude about their job. They are: task significance, which is defined as the extent to which a person believes that their job has impact on the lives of people – either society at large or in an organization; task identity, which means the extent to which a job's tasks are "whole", or involve the completion of a complete and identifiable outcome; skill variety, which is defined as the extent to which a job is perceived as requiring the use of multiple skills, talents and experience; autonomy: the extent to which an employee is given discretion as to how to complete the work required, and set a schedule for completion; and finally, feedback, which is the extent to which the process of completing the work provides a person with information through which an employee can evaluate his performance.

Perceptions of job characteristics impact affective states such as job satisfaction and work exhaustion. These are well-tested consequents of job characteristic perceptions. Further, strong claims by agile proponents [e.g., 11], and some initial evidence of impacts [21] indicate that these constructs may be directly impacted by agile method use. Job satisfaction is an affective response that is associated with one's experiences at work [23], Work exhaustion is defined as the depletion of one's emotional, mental, and physical resources due to work experiences [14]. Work exhaustion has been shown to be associated with lower job satisfaction [5,18], and higher turnover [14].

2.2. Job Design Principles and their Impact on Job Characteristics

The influential relationship of job characteristics on job satisfaction and work exhaustion is well established [e.g., 1,2,14,15,18]. Hackman & Oldham [10] also propose that there are principles of work design that , if followed, can influence perceptions of job characteristics. They argue that by designing work according to these principles, job perceptions and therefore job attitudes can be improved. The five principles are 1) Combining Tasks, 2) Forming Natural Work Units, 3) Establishing Client Relationships, 4) Vertically Loading the Job, and 5) Opening Feedback Channels. We present a short explanation of each below.

Combining tasks. Hackman & Oldham argue that many of the issues with work design emerged as a result of the fractionalization of jobs that arose out of the principles of "scientific management". They argue that if job tasks are recombined, workers will develop and exercise a broader range of skills. In addition, they will better understand how their work relates to the completion of a "whole" product.

Software development can, in a certain sense, take a work fractionalization approach. By dividing the work of software development across teams who focus on particular "layers" of the software (database, business logic, user interface), the ability to see the results of tasks being related to a completed product is reduced.

Forming natural work units. Hackman & Oldham suggest that in a quest for efficiency, work has been divided without care for the impact on worker satisfaction. This may lead to seemingly unrelated and unnatural set of tasks that a worker may address in a given time period. This leads to lower task identity and task significance.

Hackman & Oldham argue that by creating natural work units, employees develop an "ownership" response. Rather than identifying their work as "data access layer coding", programmers in the above example should being to identify their work as "data access layer coding for the XYZ module". Developing this ownership mentality can increase the perceived meaningfulness and perceived value of the work. As such, forming natural work units is expected to increase task significance and task identity.

Establishing client relationships. When work is fractured and specialized, many workers do not have contact with the actual client for whom the work is completed. If employees lack direct contact with the customer, feedback is filtered through others, and employees lose a key source of information through which they can understand the impacts of their work, which reduces task significance. Further, direct client feedback is likely to provide the clearest information through which an employee can evaluate the quality of their work.

Besides the impact of direct communication on feedback, Hackman & Oldham argue that skill variety and autonomy also increase due to establishing client relationships.

In the traditional software development context, the potential division between the "analyst", "architect", "coder" and the "tester" often creates several layers of hierarchy between an employee and the actual client.

Vertical loading. Hackman et al. [9] consider this to be potentially the most crucial job design principle. Due to specialization, jobs responsibilities have been split between the doing and the planning and controlling of the work. This creates a gap between these components of the job. By vertically loading the job, Hackman and his co-authors propose to reduce the distance between the doing and the planning and controlling of a given piece of work. "When a job is vertically loaded, responsibilities and controls that formerly were reserved for higher levels of management are added to the job." [10:64]

In a traditional software development environment, decisions regarding the design and implementation of the system may be made by one group of employees, while the "doing" of the implementation may be completed by another set of employees, indicating a lack of vertical loading.

Finally, by *opening feedback channels,* employees are able to obtain additional information relevant to the planning, doing, and results of their work. As *feedback* is one of the JCM constructs, providing additional opportunities and avenues through which to obtain feedback should increase this perception. While establishing client relationships provides a source of client-supplied information, additional opportunities for feedback can be designed into the work itself. By adding the task of quality control to an employee's job of making a product or delivering a service, the employee can obtain direct feedback about their performance. Further, automated procedures can check the quality of a product, and provide feedback directly to the individual who created it.

The remainder of our study is based on the proposition that the philosophy and the widely used practices of agile methods address each of the job design principles presented above and, because of this, impacts job characteristic perceptions and attitudes such as job satisfaction and work exhaustion.

2.3. Agile Methods As Job Redesign

Agile methods prescribe a wide range of practices. In this study we focus on those most widely used and those where research has shown a potential impact on job perceptions. According to source data provided by VersionOne from the State of Agile Survey (2011), the eleven most used agile practices (in rank order) are:

1. Daily Stand Up Meeting
2. Iteration Planning
3. Unit Testing
4. Retrospectives
5. Burndown
6. Release Planning
7. Velocity
8. Automated Builds
9. Continuous Integration
10. Coding Standards
11. Refactoring

As the level of reported use drop significantly after these initial 11 practices, we focus primarily on these practices. In addition to these top 11 practices, we will consider the use of pair programming (rank order 16) due to evidence from the literature [e.g., 21] that suggests an impact on job characteristic perceptions. Further, due to the similarity and interrelatedness of release planning, iteration planning, and velocity, we combine these practices into the concept of "work planning" Definitions of each of these practices, as well as mappings to job design principles are provided in Table 1.

Daily Stand Up Meeting: The daily stand up meeting helps to both establish client relationships, and to open feedback channels. The team as a whole performs this practice each day. Each member of the team attends, and provides information to the team regarding the work performed the previous day, the work planned for the day, and any blocking or coordination issues he or she has encountered [19]. Agile methods also prescribe that, when possible, the business owner should attend this meeting daily.

Several job design principles are in play during the daily stand up meeting. By providing the opportunity for direct discussion of the current status of the project, each team member is able to directly interact with the customer, and with each other. Questions can be asked of the customer, establishing client relationships. Further, team members can bring up issues that may have been created due to the work previously performed by another team member. In this manner, feedback channels are opened, allowing team members to better judge the quality of their previous work.

Work Planning. Release planning, Iteration planning, and Velocity are congruent with three of the job design principles as noted in Table 1. Release planning defines at a high level the order in which features will be deployed. Iteration planning is performed before each work cycle, as the team and customer together define the features included in the next cycle, divide the features into tasks, and estimate the work to be performed.

Further, work planning supports establishing client relationships by including both the client representatives and team members in the planning.

According to both Scrum and XP, the client is responsible for choosing the priorities of tasks to include in the release and iteration, while the developers are responsible for estimating the work to complete those priorities. The team uses its defined velocity [3] in order to establish the amount of work that can fit into a work cycle.

Table 1: Job Design Principles and Agile Practices

Agile Practice	CT	FNWU	ECR	VL	OFC
Daily Standup Meeting			X		X
Work Planning: Release Planning Iteration Planning Velocity		X	X	X	
Unit Testing	X				X
Retrospectives				X	X
Burndown				X	X
Automated Builds	X			X	X
Continuous Integration	X			X	X
Coding Standards				X	
Refactoring	X			X	
Pair Programming	X			X	X

CT- Combining Tasks , FNWU – Form Natural Work Units, ECR – Establish Customer Relationships, VL – Vertical Loading the Job, OFC – Opening Feedback Channels

Finally, by including the tasks of planning and estimating in the jobs of the team members, agile team members' roles are expanded. Planning and estimating were often the purview of management or others outside the team. The inclusion of these tasks that are related to the management of the project vertically load agile team members' jobs.

Unit Testing. Unit testing refers to the process of writing a separate suite of code that is not part of the system that is designed to exercise and test system code [12]. This practice both combines tasks, and opens feedback channels.

Initial unit testing of software has always been a task that developers completed. However, most developers did not write persistent code suites to test software systems. With the emergence of unit testing frameworks, teams now write unit tests that are preserved for use by all members of the team. These tests combine tasks in two ways. First, the code is an example of documentation in practice, as the tests themselves provide guidance as to the functionality of the system. Further, these tests can be run at any time, allowing a developer to test the consequences of a code change across the entire system. In this way, unit tests combine tasks previously performed by the business analyst role and the tester role.

Further, feedback channels are opened due to the immediate information that is provided to a developer via a failed test. Many development teams require coders to run the full suite of tests before committing code changes to the team repository. By running all of the tests until they pass, developers can receive information that allows them to immediately validate the quality of the code they have written.

Retrospectives. Retrospective meetings occur at the end of a work cycle. This meeting is specifically intended to allow the team to reflect on the work process used in the previous iteration, and give the team an opportunity to propose and adopt modifications to the process in the next work cycle. This process allows the team to be self-directed, and vertically loads the team with the responsibility of defining its own development process. Further, it provides a feedback channel to evaluate the quality of work for the entire team.

Burndown. The burndown chart provides a graphical representation that compares the amount of work planned at a given point in a work cycle with the amount of work actually completed [20]. Providing this information to the entire team, rather than restricting it to a project manager, allows the team to take action to ensure that the project does not fall behind. By opening this feedback channel to the team, the team is empowered and their jobs are vertically loaded.

Automated Builds and Continuous Integration. Each of these practices combine tasks, vertically load, and open feedback channels. Automated builds refer to the process of creating scripts to generate a complete and deployable build [12]. This process can be executed as needed by the team, and reduces the need for a dedicated configuration manager role or team, and places the responsibility for ensuring that the packages are complete and deployable onto the agile team.

Continuous integration refers to the process of systematically and regularly both building and deploying the code to a test server [7]. This process opens a feedback channel that allows the team to determine if a change has been completed incorrectly. It provides feedback above and beyond the process of automated builds, as it ensures not only that the code runs on a developer's machine, but that all of the configuration changes necessary have been committed to the code repository.

Coding Standards. Coding standards refers to the group's established norms as to code-naming and consistency [3]. This practice vertically loads the job of an agile team member, as it is an established set of rules by which the team will develop software. These standards are not imposed on the team, but rather are a form of self-management. Further, each member of the team is empowered to suggest changes to the standards either at a retrospective, or during a work cycle.

Refactoring. Refactoring refers to any number of practices that lead to the removal of redundancy, elimination of unused functionality, and refresh obsolete designs [8]. The practice of refactoring both combines tasks and vertically loads. Refactoring refers to the commitment of the team to improve the structure and reduce the complexity of the code whenever necessary.

Importantly, refactoring is performed by the entire team, whenever necessary, rather than by a particular group, or in a process requiring architectural oversight.

Pair Programming. Pair programming refers to the practice of two developers working together to develop a portion of code [6]. Pair programming combines tasks, vertically loads the job, and opens feedback channels.

When pair programming, it is assumed that the pair work together to first design the software, and then build it. In some cases, writing the tests is considered a light implementation of a design specification. In these cases, some advocate that the first programmer write the tests for the code, while the second provides feedback. When programming the actual system code, the pair reverses roles. Assigning authority for design, development, and testing to the pair combines tasks, and vertically loads.

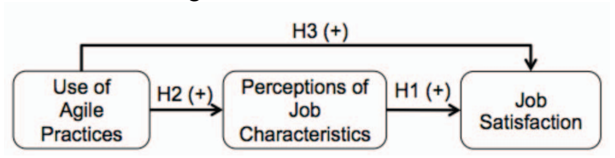
In addition, the act of working in pairs allows for mistakes to be identified in the act of coding. When the observing coder notices a mistake in progress, immediate feedback can be given to correct it. Further, as pairs are intended to be fluid structures, recombining regularly, feedback channels are opened across the team to ensure that coding standards are being implemented consistently across the team.

In this section of the paper, we described Hackman & Oldham's job characteristics model and job design principles. Further, we illustrated the numerous ways in which agile methods' practices implement the job design principled. In the next section of the paper we present our research model and hypotheses.

3. Hypothesis Development

Our research model is presented in Figure 1.

Figure 1: Research Model



Our research model builds on previous research. Our first hypothesis is well tested in the literature [e.g., 1,15]. As our research model is incremental in nature, we include this hypothesis as part of the established nomological network. Each of the construct definitions is included in Table 2.

H1. Higher perceptions of job characteristics are positively related to job satisfaction.

As described in the previous section of the paper, the use of the agile practices is highly congruent with the job design principles proposed by Hackman & Oldham. Because the use of the job design principles is posited to increase the positive perceptions of work characteristics, we propose:

H2: The level of agile method use will positively impact job characteristic perceptions.

Additionally, we propose direct effects of agile use on the outcome variables. Agile methodologists report and some research has supported that agile teams have higher engagement with the client, higher developer motivation, and deliver software of higher quality and with shorter project timelines. Because agile teams experience higher project success rates

H3: The extent of agile use will positively impact job satisfaction.

In the next section we present a research-in-progress study that tests the hypotheses noted above.

4. Methodology

In order to test this preliminary model, we collected survey data in June 2013, consisting of 104 respondents who were software development professionals. We utilized the Empanel, Inc. software developer panel to obtain our respondents. Respondents were screened to ensure that they were part of a software development team, and played a non-management, non-customer role on the team. Further, we screened for developers with more than 1 year of total experience, and at least six months at their current organization.

In order to assure quality responses, we positioned "quality assurance (QA)" questions such as "if you are still paying attention, select 'strongly disagree'" at several points in the survey. If these QA questions were not properly answered, the respondent was removed from the sample. Finally, as the questionnaire was long, and to provide additional assurance that respondents were paying attention, we dropped any respondents who completed the questionnaire in less than 10 minutes.

Table 2: Construct Definitions

Construct Name	Definition
Outcome Variables:	
Job Satisfaction	The extent of positive emotional response to the job resulting from an employee's appraisal of the job as fulfilling or congruent with the individual's values. (Morris & Venkatesh 2010)
Perceptions of Job Characteristics:	
Skill Variety	The extent to which a job requires the use of different talents. (Hackman & Oldham 1980; Morris & Venkatesh 2012)
Task Identity	The extent to which a job involves completing a whole identifiable outcome. (Hackman & Oldham 1980; Morris & Venkatesh 2012)
Task Significance	The extent to which a job has impact on the lives of people in an organization or society in general. (Hackman & Oldham 1980; Morris & Venkatesh 2012)
Autonomy	The extent to which a job provides the employee with discretion to choose how the work is done and to set the schedule for completing the work activities. (Hackman & Oldham 1980; Morris & Venkatesh 2012)
Feedback	The extent to which carrying out the work activities provides the employee with clear information about his or her performance. (Hackman & Oldham 1980; Morris & Venkatesh 2012)
Independent Variable	
Use of Agile Practices	The extent to which the respondent's team utilizes the 12 practices defined in this study.

362 respondents began the survey. Of these, 159 were screened out due to the initial screen questions regarding team role and tenure. 125 more were disqualified based upon the quality assurance tests described above. The remaining 104 respondents completed the entire questionnaire. Respondents who completed the entire survey were compensated by Empanel with points redeemable for cash, merchandise or services. A breakdown of the sample is presented in Table 3.

Table 3. Sample Breakdown

Team Role	N	Avg. Dev. Exp.	Avg. Org. Tenure
Team Lead	27	8.35	6.22
Architect	6	12.25	11.05
Developer	57	8.47	6.11
PM/Scrum Master	6	9.93	9.30
QA/Testing	5	5.98	9.28
Business Analyst	3	4.4	5.9

We adapted some scales from previous research: job satisfaction and job characteristics were measured using scales slightly modified from the Hackman & Oldham scales as used by Morris and Venkatesh [15].

To measure agile method use, we developed a new set of items. Practitioners may use similar terms to describe their practices in use, even as those practices are not executed in the same manner. Because of this, we attempted to develop scales that reflected practices as described by the agile literature [e.g., 3,19]. In order to develop this scale, one of the authors used agile development publications and the input of an external agile researcher, and an agile practitioner to develop an initial set of items for each of the 12 agile practices of interest. Once these items were developed, a sorting exercise was performed using two additional agile practitioners. Initial agreement on the scales was approximately 60%. The items that did not perform well were discussed with the practitioners who did the sort. Based upon this discussion, the items were modified or replaced, and a second round of sorting was performed using three additional agile developers. The agreement in this sorting exercise was over 85%. These items were used for the study. Representative sample questions from the questionnaire are shown in Appendix B.

Further we measured negative affectivity as a control [14]. Negative affectivity is a state factor; individuals who measure high in negative affectivity are more likely to experience dissatisfaction with themselves and their lives than those who measure lower [22]. Finally, we control for six demographic variables: age, organizational tenure, gender, education, organizational tenure and total work experience.

All statistical analysis for this study was performed using STATA Version 12. Initially, we performed an exploratory factor analysis using principal components factors for only the items that were developed for the agile practices. Because we did not expect the factors to be orthogonal, we used oblique oblimin rotation. We performed item culling (**), and due to the fact that the items are new, we retained items that loaded at .6 or higher, and had no cross loadings over .4. While we developed measures for 12 agile practices, the items loaded onto seven factors: Coding Standards, Daily Standup, Refactoring, Pair Programming, Unit Testing, Iterative Planning, and Automated Builds. All of the factors retained mapped to the initial sorting exercise except iterative planning. As indicated in our discussion in section three, we grouped release planning, iteration planning and velocity into a meta-category called "work planning". As such, it is not surprising that the constructs were very similar and did not discriminate. Iterative planning consists of two of the items initially conceptualized as iteration planning, and two of the items conceptualized into velocity. The remaining items for all agile practices loaded on multiple factors or failed to load and were dropped.

We then performed a full factor analysis using the retained agile practices as well as the job characteristics model and job satisfaction. When performing the full factor analysis, two more agile factors (coding standards and unit testing) failed to discriminate and were dropped. Surprisingly, job autonomy also failed to load independently from task identity. We propose that this may be due to the incongruity between Hackman & Oldham's concept of job autonomy and agile method philosophy. Hackman & Oldham propose that job autonomy is the ability to act alone without management interference. In contrast, agile teams take a strongly team-oriented rather than individually autonomous approach to action. Because of this, we dropped job autonomy from the analysis. The final EFA factor loading along with the Cronbach's alpha for each construct is presented in Appendix A. Each of the Cronbach's alphas was greater than the recommended .70.

Finally, we performed a confirmatory factor analysis on the retained measurement model. The CFA indicated an acceptable fit (CFI = .933, RMSEA=.056, SRMR=.064).

We assessed common method variance by using Harman's single-factor test [16], finding that the first factor did not account for a majority of the explained variance explained (only 32% of the 84% explained). Based on this, common method variance was not identified. Further, we took several steps to prevent common method bias. We used different scale headers for different constructs, and we grouped items by construct so as not to disrupt the instrument's logical flow [16].

We tested for variance inflation factors (VIF) of the agile factors, and the perceptions of job characteristics factors. The highest VIF was 1.77, indicating that there was not an issue with multicollinearity.

For this preliminary analysis, we created an index for the use of agile methods by averaging the factor scores for the five retained agile factors. We also created an index for the JCM by averaging the four retained JCM factors.

We performed path regression to test our model with results presented in table 4.

We tested the moderating effect of JCM on AGILE using the Sobel-Goodman test with bootstrapping [17]. This test confirmed the results of the path regression, indicating that approximately 19% of the effect of AGILE on JOBSAT is mediated by JCM.

These preliminary results of our study show that there is a significant, partially mediated relationship between the use of agile methods and job satisfaction.

6. Discussion and Next Steps

We developed an initial model of the impacts of agile method use on job satisfaction. We found preliminary support for our hypotheses of both the direct effect of agile method use and the mediating effect of the job characteristics perceptions on job satisfaction. These results are summarized in Table 5.

In a test of 104 software professionals, the model explains 39% of the variation in job satisfaction, and 30% of the variation on the perceptions of job characteristics. Further, the model supports the partial mediation of the impact of agile method use on job satisfaction. The model contributes to the agile software development and job satisfaction literature by providing a theoretical lens through which to view the impact of agile method use on job satisfaction, as well as to provide evidence of the job design principles effect on the perceptions of job characteristics. We discuss next steps below.

One of the next steps in this research project is to look more deeply into the relationships between the individual agile method constructs and the job characteristics constructs. We intend to collect approximately 200 additional responses during the summer of 2013 in order to perform more robust analyses utilizing SEM techniques. Further, we intend to investigate several moderation effects in order to attempt to establish boundaries for the relevance of the theory.

Table 4: Regression Results

	Path I	Path II
DV: Job Sat		
JCM (H1)	.227**	
AGILE (H2)	.299***	
Negative Affect.	-.253***	
Age	.012	
Education	.100	
Gender	.106	
Ethnicity	-.029	
Org Tenure	-.068	
Experience	-.071	
DV: JCM		
AGILE (H3)		**.446*** **
Negative Affect.		-.134
Age		-.092
Education		-.034
Gender		.156
Ethnicity		.071
Org Tenure		.091
Experience		.148
N	104	104
F	(9, 94) = 6.74	(8, 95) = 5.15
Prob > F	0.000	0.000
R^2	0.3922	0.3027
**p<.01, ***p<.001		

Table 5: Hypothesis Results

Construct	Coeff.	Support?
H1: JCM → JOBSAT(+)	**.227** **	**Yes**
H2: AGILE → JCM (+)	**.446*** **	**Yes**
H3: AGILE → JOBSAT(+)	**.299*** **	**Yes**

In addition to furthering our knowledge of the impact of agile method use on job satisfaction, future research could also explore how the use of agile methods impacts other relevant job perceptions such as perceived work overload and/or turnover intention.

Further, the fact that the classic Hackman & Oldham construct of Job Autonomy did not emerge as a discriminant construct in this context is an unexpected finding. As posited above, this may be potentially due to the heavy team-orientation of agile teams. Future research should investigate whether constructs such as team autonomy may better reflect the perception of agile teams.

From a practice perspective, the findings also provide some initial evidence that may be utilized in the field. Whether or not an organization chooses to implement agile development methods, the mechanisms of the practices investigated in this study may still be applied. Providing teams with greater control over the way that they perform work, greater transparency regarding the project goals and objectives via more frequent feedback from the customer, and investing in automated testing can be achieved whether an organization uses agile methods or not.

7. Limitations

As with all studies, there are a few limitations of this study. The first limitation is with regard to the generalizability of our findings. The sample drawn for this study consisted of IS professionals working in various organizations and industries throughout the United States and should be fairly generalizable. However, there are potential cultural and other differences that may make this study less generalizable across non-US contexts.

Further, while we conceptualized this study at the individual level, there is some evidence from our findings that this concept may be appropriately studied at the team level, or with a multi-level model. As we do not have multiple respondents from teams, we cannot pursue a multi-level analysis with this data. However, future research should include multiple respondents from the same team.

8. Conclusion

We have described the preliminary findings of an exploratory study on the use of agile information systems development methods and job satisfaction. Using a sample of 104 software development professionals surveyed, our study finds evidence of the positive impact of agile method use on perceptions of job characteristics, and job satisfaction. We conclude that there are complex relations still to be discovered regarding the impact of agile method use on job perceptions, and that the use of agile methods has the potential to make non-trivial impacts on the well-being of software professionals.

9. References

[1] Ahuja, M.K., Chudoba, K.M., Kacmar, C.J., McKnight, D.H., and George, J.F. IT road warriors: Balancing work-family conflict, job autonomy, and work overload to mitigate turnover intentions. *MIS Quarterly 31*, 1 (2007), 1–17.

[2] Ang, S. and Slaughter, S.A. Work outcomes and job design for contract versus permanent information systems professionals on software development teams. *MIS Quarterly 25*, 3 (2001), 321–350.

[3] Beck, K. *Extreme programming explained: embrace change*. Addison-Wesley Professional, 2000.

[4] Beecham, S., Baddoo, N., Hall, T., Robinson, H., and Sharp, H. Motivation in Software Engineering: A systematic literature review. *Information and Software Technology 50*, 9 (2008), 860–878.

[5] Burke, R.J. and Greenglass, E. A longitudinal study of psychological burnout in teachers. *Human Relations 48*, 2 (1995), 187–202.

[6] Cockburn, A. and Williams, L. The costs and benefits of pair programming. (2001), 223–248.

[7] Duvall, P.M., Matyas, S., and Glover, A. *Continuous integration: improving software quality and reducing risk*. Addison-Wesley Professional, 2007.

[8] Fowler, M. *Refactoring. Improving the design of Exiting Code*. Addison Wesley Longman, Reading, MA, 1999.

[9] Hackman, J.R., Brousseau, K.R., and Weiss, J.A. The interaction of task design and group performance strategies in determining group effectiveness. *Organizational Behavior and Human Performance 16*, 2 (1976), 350–365.

[10] Hackman, J. and Oldham, G. *Work Redesign*. Addison Wesley, 1980.

[11] Highsmith, J. *Agile Software Development Ecosystems*. Addison Wesley, Boston, 2002.

[12] Küster, J., Gschwind, T., and Zimmermann, O. Incremental development of model transformation chains using automated testing. *Model Driven Engineering Languages and Systems*, (2009), 733–747.

[13] McHugh, O., Conboy, K., and Lang, M. Using agile practices to influence motivation within it project teams. *Scandinavian Journal of Information Systems 23*, 2 (2011), 85–110.

[14] Moore, J.E. One road to turnover: An examination of work exhaustion in technology professionals. *Management Information Systems Quarterly 24*, 1 (2000), 141–168.

[15] Morris, M.G. and Venkatesh, V. Job characteristics and job satisfaction: Understanding the role of enterprise resource planning system implementation. *MIS Quarterly 34*, 1 (2010), 143.

[16] Podsakoff, P.M., MacKenzie, S.B., Lee, J.Y., and Podsakoff, N.P. Common method biases in behavioral research: A critical review of the literature and recommended remedies. *Journal of Applied Psychology 88*, 5 (2003), 879.

[17] Preacher, K.J. and Hayes, A.F. Asymptotic and resampling strategies for assessing and comparing indirect effects in multiple mediator models. *Behavior Research Methods 40*, 3, 879–891.

[18] Rutner, P.S., Hardgrave, B.C., and McKnight, D.H. Emotional dissonance and the information technology professional. *MIS Quarterly 32*, 3 (2008), 635–652.

[19] Schwaber, K. and Beedle, M. *Agile software development with Scrum*. Prentice Hall Upper Saddle River, NJ, 2002.

[20] Sutherland, J. Agile can scale: Inventing and reinventing scrum in five companies. *Cutter IT Journal 14*, 12 (2001), 5–11.

[21] Tessem, B. and Maurer, F. Job satisfaction and motivation in a large agile team. In *Agile Processes in Software Engineering and Extreme Programming*. Springer, 2007, 54–61.

[22] Watson, D. and Clark, L.A. Negative affectivity: the disposition to experience aversive emotional states. *Psychological bulletin 96*, 3 (1984), 465.

[23] Weiss, H.M. and Cropanzano, R. Affective Events Theory: A theoretical discussion of the structure, causes and consequences of affective experiences at work. (1996).

[24] West, D., Grant, T., Gerush, M., and D'Silva, D. Agile development: Mainstream adoption has changed agility. *Forrester Research*, (2010).

Appendix A. Factor Loading Table

	PAIR	STAND-UP	REFAC	ITER PLAN	AUTO BUILD	FEED-BACK	TASK SIG	TASK ID	SKILL VAR	JOB SAT
Pair1	**0.8120**	0.3017								
Pair2	**0.8173**									
Pair3	**0.7993**									
StandUp1		**0.8609**								
StandUp2		**0.8328**								
StandUp3		**0.8634**								
Refac1			**0.7374**							
Refac2			**0.8171**							
Refac3			**0.7902**							
IterPlan1				**0.7480**						
IterPlan2				**0.7178**						
IterPlan3				**0.6800**						
AutoBuild1	0.3435				**0.7423**					
AutoBuild2					**0.8043**					
AutoBuild3					**0.8511**					
Feedback1						**0.7613**				
Feedback2						**0.7615**				
Feedback3						**0.7533**				
TaskSig1							**0.7147**			
TaskSig2							**0.8734**			
TaskSig3							**0.6951**			
TaskID1								**0.8896**		
TaskID2								**0.8754**		
TaskID3								**0.8064**		
Skillvar1									**0.8102**	
Skillvar2									**0.8377**	
Skillvar3					.3041				**0.6471**	
JobSat1										**0.8421**
JobSat2										**0.8495**
JobSat3										**0.7656**
Cronbach's Alpha	0.8552	0.9086	0.8344	0.7963	0.8161	0.8115	0.7623	0.8950	0.7741	0.8830
Exploratory factor analysis results, oblique oblimin rotation. Cross loadings over .3 included.										

Appendix B. Sample Agile Method Use Questions

Agile Method Use

(7-point scale anchored at (1) Strongly Disagree, (4) Could Agree or Disagree, (7) Strongly Agree)

Agile Method Use: Daily Stand Up

1. The team has a short meeting every day to discuss what is going on with the project.

Agile Method Use: Pair Programming

1. Our code is created by two people working together at a single computer.

Agile Method Use: Refactoring

1. Whenever we see the need, we improve the design of the code we have written previously.